RNN's for Stock Price Prediction

Vineeth Marikuntematha Ravisharadhya
University of Adelaide
Adelaide, SA 5000
Email id: a1896845@adelaide.edu.au

Abstract

This study aims to explore the application of RNN-LSTM in forecasting stock market fluctuations, aiding investors in comprehending stock prices. This predictive analysis enables investors to utilize diverse machine learning methods for optimal stock selection, thereby augmenting their investment returns. LSTM and GRU , are deep learning methods, facilitates the analysis and prediction of stock market data, considering challenges, profits, investments, and future market performance. The research delves into employing neural networks to grasp alterations in stock prices, interest trends, and trading activities within the stock market domain. Data from companies like Apple, Google, Microsoft, and Amazon were used to perform the task using RNN-LSTM model. Finally, The GRU model showed the best result giving less error of 4.61 than LSTM and simple RNN models.

1. Introduction

As the market economy grows, the financial services system has improved. People are increasingly investing in the financial market and seeking more financial information. The stock market is now an essential part of the broader financial industry, especially the securities sector. Market fluctuations, both upward and downward, significantly impact economic health and shareholders' interests, posing challenges to the market economy. With the modernization of society and the internet, computer systems and information technology have advanced, offering powerful tools to handle financial information. Predicting stock price trends accurately has become a primary focus for many researchers due to these advancements [1,3].

From the outset of the stock market, investors have sought ways to anticipate stock prices. Technical analysts study stock charts to recognize patterns and trends, aiding in the prediction of future stock values. Analyzing vast amounts of historical market data poses a challenge for financial analysts, prompting the development of various Artificial Intelligence (AI) techniques for automated stock price prediction [3]. The exploration of automated stock price prediction traces back to around 1994, with early research focusing on machine learning regression models. Since then, continual efforts have been made to create automatic strategies for forecasting stock prices. The advancement of AI, big data, and improved computational capabilities has made automated stock price forecasting increasingly feasible.

Using machine learning involves utilizing algorithms to analyze data, enabling predictions regarding stock prices.

Machine learning also faces a big challenge as its success depends a lot on how data is shown [2,3] . Stock market data over time behaves unpredictably, which makes it tough to create useful features from it. This complexity makes it harder to use machine learning for predicting stock prices. However, deep learning models are favored as they don't need separate feature building, making them popular for forecasting stock prices or trends using large amounts of past data.

This study compared different recurrent neural network (RNN) models like Long Short-term Memory (LSTM), and Gated Recurrent Unit (GRU)—based on how well they predict outcomes [2]. These RNN types were chosen because they're known and widely used for predicting sequences . The LSTM Model's hidden and cell states assist in extracting relevant and accurate information from processed data.

1.1. Related Work

In 2012, Anshul Mittal [4] applied sentiment analysis and machine learning principles to investigate the relationship between "public sentiment" and "market sentiment." They introduced a novel cross-validation method for financial data and achieved 75.56% accuracy using Self Organizing Fuzzy Neural Networks (SOFNN) by analyzing Twitter feeds alongside DJIA values from June 2009 to December 2009.

In 2013, Jianfeng Si [5] proposed a technique to use topic-based sentiments from Twitter to predict the stock market. Their approach involved learning daily topic sets using a continuous Dirichlet Process Mixture model. For each topic, they developed a sentiment time series and correlated these with the stock index, aiming to predict market trends.

In 2014, Ayodele A. Adebiyi [6] detailed the process of constructing a predictive model for stock prices using the ARIMA model. The research highlighted the ARIMA model's potential for short-term prediction, showing competitive results compared to existing methods for stock price prediction.

1.2.Dataset Description

Dataset comprises a comprehensive collection of companies data like Apple ,Google , Microsoft and Amazon. The dataset is imported from the yfinance python library using datareader so the data will always be dependent on the regular stock price . The data mainly exclaims on the highs and lows of the price of each company on a regular basis. To encourage the usability of the dataset we split the data to train and test them.

Furthermore , this paper contains the section starting with method description where the methods used during the process have been explained , experimental analysis explains about results obtained during the process, conclusion and reference.

2. Description of the method

2.1. Data Preprocess

The primary goal of this research is to identify or predict the data as per the data we have gathered from the yahoo finance library (yfinance) . Initially the dataset undergoes data pre-processing where the left out data will be monitored and fill it up with necessary action. The data which is cleaned and sorted will be further trained and tested with RNN models. Finally the output of the data helps us to predict and find out the error.

Some of the functions and optimizers used during the model training are listed below:

Tanh Function

The tanh function, also known as the hyperbolic tangent function, is a mathematical operation frequently applied as an activation function in Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. It is highly advantageous in these frameworks as it introduces non-linear characteristics, aiding the network in understanding intricate patterns and connections within sequential data.

$$tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Here, x represents the input to the function.

Adam Optimizer

This type of optimizer is extensively used in training neural networks such as RNNs and LSTMs. It combines the advantages of AdaGrad and RMSProp by utilizing adaptive learning rates and momentum. In essence, Adam adjusts the learning rates for each parameter within the neural network based on historical information about the gradients' first and second moments. This adaptive strategy significantly enhances optimization efficiency during the training phase.

Dense Layer

The dense layer, also referred to as the fully connected layer, serves as a key element in neural networks. It establishes connections between each neuron in one layer to every neuron in the subsequent layer, facilitating the network's capability to comprehend intricate patterns from the input data. This layer performs a linear operation on the input, succeeded by an activation function, enabling the network to model intricate non-linear relationships between the input and output.

Loss Function

Loss functions are fundamental elements in machine learning. Mean Squared Error (MSE) calculates the average squared difference between predicted and actual values in regression tasks, placing higher emphasis on larger errors by squaring the deviations. Conversely, Cross Entropy serves as an evaluation metric in classification scenarios, measuring the distinction between predicted probabilities and true class labels, often employed in tandem with softmax activation for classifying outputs.

Memory Cells

Memory cells are also known as Long Short-Term Memory networks, differ from traditional RNNs due to their special memory cells. These cells act as storage units, letting the network retain and selectively remove information over long sequences. This unique setup prevents a common problem seen in regular RNNs called vanishing gradients. Moreover, these cells have gates controlling data flow, allowing the network to decide what to remember, discard, or update. This ability is crucial for understanding long-term patterns in sequences. LSTMs perform well in tasks like language modeling, time series prediction, and natural language processing by effectively capturing complex patterns in sequential data.

Update Gate and Reset Gate

The update gate in a Gated Recurrent Unit (GRU) decides how much old information to keep and how much new data to use from the current input. It ranges between 0 (completely forgetting old memory) and 1 (keeping it entirely), based on the current input and previous hidden state. Similarly, the reset gate controls how much old information to ignore when calculating the new candidate state. Like the update gate, it considers the current input and previous hidden state. Both gates are crucial in deciding how the network blends past and present information to update its memory for the next step in the sequence.

Minmaxscaler

MinMaxScaler is a method in machine learning used to change features within a set range, often from 0 to 1. It adjusts data by using the smallest and largest values in the dataset. By applying a formula that involves subtracting the smallest value and dividing it by the difference between the largest and smallest values, MinMaxScaler changes the data so that the smallest becomes 0 and the largest becomes 1. This normalization helps when features have different scales or units, ensuring a balanced effect when training models. It's commonly used in various machine learning methods like neural networks, support vector machines, and clustering algorithms, making features similar for better model performance.

Rooted Mean Square Error and Mean Square Error

The Mean Squared Error (MSE) evaluates the average of the squared disparities between predicted and observed values within a dataset. It quantifies the model's performance by computing the average of the squared deviations between predicted and actual values.

represented by the formula

$$\text{MSE} = \frac{1}{n}\sum_{i=1} + \frac{c}{d} = (Y_i - Y'_i)\text{^}2$$

, normalizes the squared differences by the number of samples (n).

RMSE is the square root of MSE. It shows the average size of errors in a way that's easy to understand. RMSE helps gauge how closely the model's predictions match the actual data, using the same units as the original data.

RMSE = sqrt(MSE)

2.2 Neural Network

A neural network is a standard multilayer structure where neurons in different layers are connected. This connection involves each neuron in a lower layer linked to neurons in upper layers to establish a connection weight. Notably, neurons within the same layer don't have direct connections or weights between them. The

structure of this neural network is illustrated in Figure 1 .

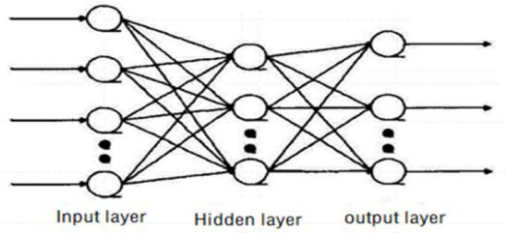

Figure 1. Structure of neural network

2.3 Recurrent Neural Networks

In a regular neural network, connections span from the input layer to the hidden layer and then to the output layer, lacking direct links between nodes within layers. This setup slows down training and testing, leading to problems like overfitting and local minimum, limiting its usefulness for various tasks. Recurrent Neural Networks (RNN) are specialized for handling sequences, widely applied in speech recognition, language translation, and image descriptions [6]. They particularly shine in tasks related to natural language processing (NLP).

RNNs establish connections over different time steps, using the same weight and creating cyclic links across time. This weight sharing reduces the number of temporal parameters in RNN systems. To simplify complexity, the current state is often connected to only a few past states. Figure 2 displays the basic structure of RNN loops and their modules.

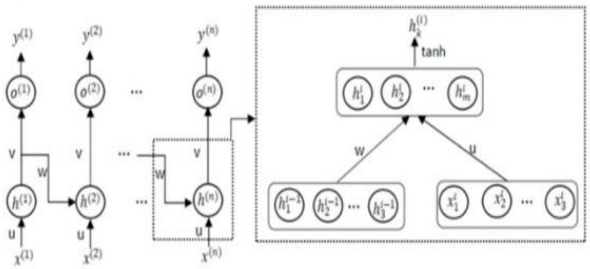

Figure 2. Basic structure of Recurrent Neural Network

2.3.1 Long Short-term Memory Network

LSTM is a special kind of RNN that tackles issues in traditional RNNs like losing important information or getting too much data. LSTMs are great at handling sequences of data by remembering important things for a long time. They do this using a memory cell and gates that control how data moves in and out, helping them remember or forget things when needed. People use LSTMs a lot for things like working with time-based data, understanding languages, recognizing speech, and analyzing sequences of information because they're good at remembering important details for a long time. In addition, every LSTM cell computes new values of hidden state and cell sate.A common architecture is composed of a memory cell, an input gate, an output gate and a forget gate ,The mathematical formulation and architecture of the LSTM cell is given below along with Figure 3.

$$f_t = \sigma\,(x_t W_f + H_{t-1} U_f) \qquad i_t = \sigma\,(x_t W_i + H_{t-1} U_i)$$
$$o_t = \sigma\,(x_t W_o + H_{t-1} U_o) \qquad H' = \tanh(x\,W + H\;\; U)$$

$$S_t = \sigma\,(S_{t-1} \times f_t + i_t \times H') \qquad H_t = \tanh(S_t) \times o_t$$

where, i, f , and o are input, forget, and output gates respectively , H and S are hidden state and memory state and W represents the weight matrices.

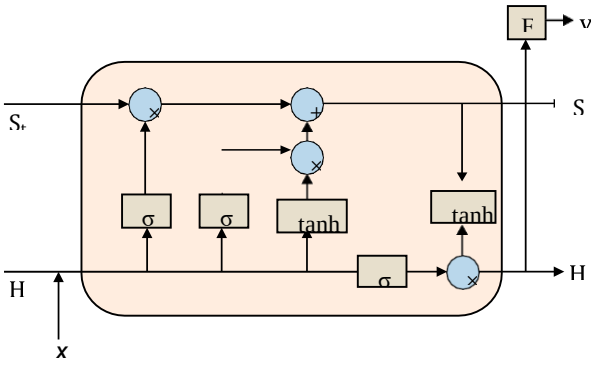

Figure 3. Architecture of LSTM

2.3.2 Gated Recurrent Unit Network

GRU is like LSTM but simpler. It's an RNN made to handle sequences and remember important details. It uses gates to manage information flow, but it has fewer gates than LSTM, which makes it simpler to use and train. GRUs solve the vanishing gradient issue and help learn distant connections in sequences, akin to LSTMs. However, they have a simpler setup by merging input, forget, and output gates into two gates (update and reset). GRUs are used in tasks such as language processing, speech recognition, and predicting sequences of events. GRUs are often used in tasks dealing with sequences, like understanding languages, translating text, and predicting time-based data.

$$Z_t = \sigma\,(W_z x_t + U_z H_{t-1}) \qquad R_t = \sigma\,(W_r x_t + U_r H_{t-1})$$
$$H = \tanh(W_h x_t + (R_t \times H_{t-1})\, U_h) \qquad H = (Z_t \text{ x } H_t) + ((1-Z_t) \text{ x } H_{t-1})$$

where, Z and R are update and reset gates

H_t' and H_t are c and i are hidden states

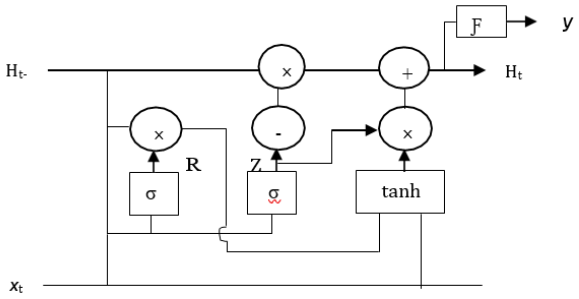

Figure 4. Architecture of GRU

3. Experimental Analysis

The experimentation were conducted on the basis of regular stock price as per the imported yahoo finance data. Each process involved in this experimentation leads to the outcome of the prediction of the stock price which were trained and tested between three RNN models namely simple rnn , LSTM and GRU. The evaluation of the model performance was based on volume of sales the companies have had and their start price at the initial stock sale, further we take the main consideration of the price of the stock when it was at the closing period , which helps us to know how the prediction of a stock price can be expected.

According the initial process I have imported the data from yahoo finance through the library -yfinance which the data is read from the datareader function. Then as said earlier the data have been thoroughly segregated , so as to plot the insights through a graphical representation of closing price.

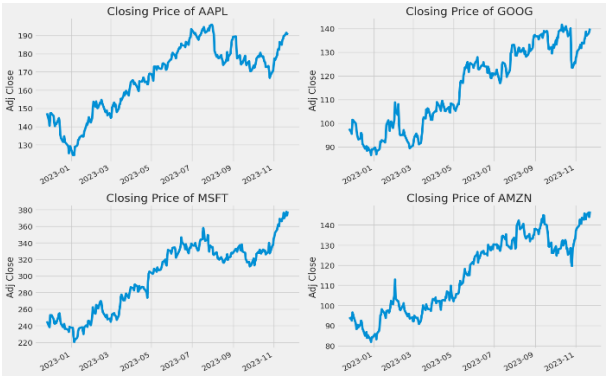

Figure 5: Data visualization for Closing price

During this process I have also considered the average moving stocks to consideration as it suggests the idea of how much stocks the companies have been selling for the amount of stock price on the particular day where this also helps us to find out the stocks sold on an average on a regular basis of each company.

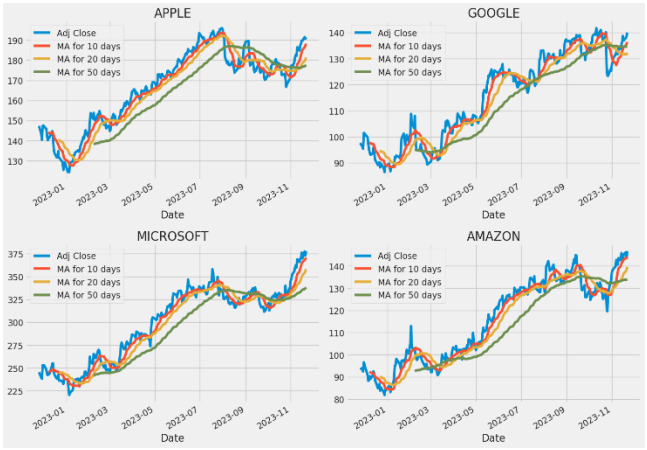

Figure 6: Data visualization for average moving stocks

Now the further I have intrigued the process of correlation so as to study the intensity and direction of the linear connection between two variables, presenting values within the range of -1 to 1. A coefficient of 1 signifies an absolute positive correlation, indicating that as one variable increases, the other also increases in a perfectly linear manner. Conversely, a coefficient of -1 denotes an absolute negative correlation, demonstrating that as one variable rises, the other decreases in a perfectly linear fashion.

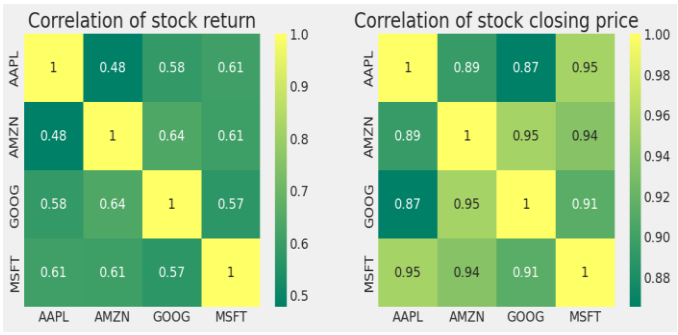

Figure 7. Data visualization of correlation between stock closing and stock return

Now the architecture of RNN model building helps to predict the stock price on a daily basis. In this project I have used three architectures of RNN namely simple RNN , LSTM and GRU. Moving on to one by one results or outcome of each model architecture of RNN as follows

Simple Recurrent Neural Networks (RNN's)

The Keras Sequential model includes two SimpleRNN layers: one with 128 units returning sequences and another with 64 units giving only the last output, both using Tanh activation. It's followed by two Dense layers: one with 25 units and the other with a single unit for continuous regression output.

The model learns patterns in sequences by reducing errors between predictions and actual values through 'adam' optimizer and 'mean_squared_error' loss function across 5 epochs with a batch size of 32. Its goal is to predict continuous values in sequences by adjusting its settings to make more accurate predictions and minimize errors.

Table 1. Represents the RMSE and MSE of the simple RNN model

.

Rooted mean square error (RMSE)	Mean square error (MSE)
8.992649	80.86775119

As model predicts by using it to guess outcomes for test data. Then, it checks how close these guesses are to the real results by calculating RMSE, which shows the average difference between the guesses and the actual values as shown in Table 1. It also computes MSE using the sklearn library, measuring the squared difference between the actual and predicted closing prices for validation. Both RMSE and MSE help understand how accurately the model predicts stock prices or similar continuous data, gauging its overall performance.

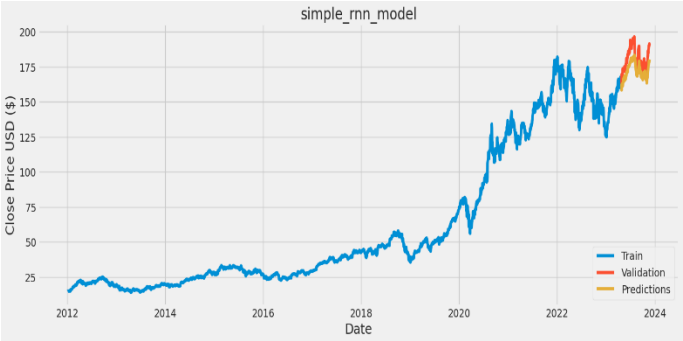

Figure 8. Representation of train , validation and prediction over time using simple RNN model

Then further a large plot shows how well the 'simple_rnn_model' predicts stock prices over time as shown in Figure 8. It compares the actual prices during training and validation periods ('Train' and 'Validation') with the model's predicted prices ('Predictions'). This visual comparison helps in understanding how closely the model's guesses match the real prices at different times. The figure 9 represents the difference between predicted price and stock close price.

Date	Close	Predictions
2023-04-24	165.330002	160.733292
2023-04-25	163.770004	161.896851
2023-04-26	163.759995	160.358414
2023-04-27	168.410004	158.366928
2023-04-28	169.679993	162.305374
...	...	...
2023-11-16	189.710007	177.349579
2023-11-17	189.690002	179.248322
2023-11-20	191.449997	178.672989
2023-11-21	190.639999	179.043594
2023-11-22	191.309998	180.161926

[149 rows x 2 columns]

Figure 9. Representation of predicted price vs close stock price of simple RNN model

Long Short-term Memory Network (LSTM)

The LSTM neural network in Keras, designed for handling sequential data such as time-series information. This model consists of two LSTM layers—one with 128 memory units processing sequences and another with 64 units producing a single output, both utilizing the Tanh function. Additionally, there are two Dense layers: one with 25 units employing Tanh and another with a single unit for the final output. The model is prepared for training using the 'adam' optimizer and 'mean_squared_error' as a measure of errors. Subsequently, it undergoes training on 'x_train' and 'y_train' data for 5 epochs, analyzing 32 data points in each iteration.

Table 2. Represents the RMSE and MSE of the LSTM model

Rooted mean square error (RMSE)	Mean square error (MSE)
5.5834700	31.1751380

The model predicts by comparing its guesses on test data with the real values. As shown in Table 2 ,it calculates RMSE and MSE to measure the average differences between the model's predictions and the actual stock prices as shown. These metrics help understand how accurate the model is in predicting continuous values like stock prices.

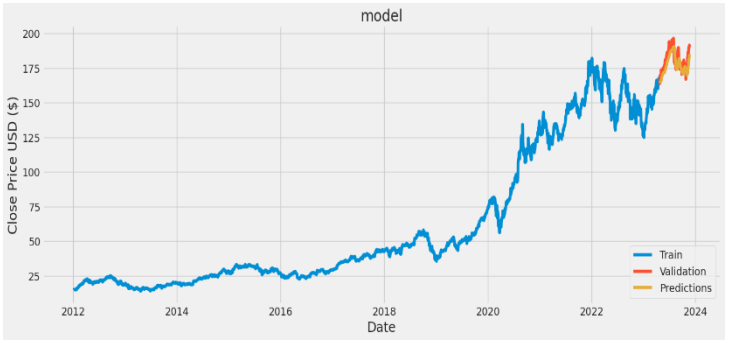

Figure 10: Representation of train , validation and prediction over time using LSTM model

The above figure 10, having graph showing historical stock prices used for training, actual prices from validation, and the model's predicted prices. It helps in comparing how well the model's predictions match the real prices over time. The figure 11 represents the difference between predicted price and stock close price.

Date	Close	Predic_tions
2023-04-24	165.330002	164.260864
2023-04-25	163.770004	164.449890
2023-04-26	163.759995	164.466248
2023-04-27	168.410004	164.370361
2023-04-28	169.679993	164.506302
...	...	...
2023-11-16	189.710007	180.283783
2023-11-17	189.690002	181.588242
2023-11-20	191.449997	182.780777
2023-11-21	190.639999	183.931778
2023-11-22	191.309998	184.916519

[149 rows x 2 columns]

Figure 11. Representation of predicted price vs close stock price of LSTM model

Gated Recurrent Unit Network

The GRU model used during the experiment contains two layers: the first processes sequences with 64 units, and the second has 64 units, producing a single output. Both GRU layers use the tanh activation function. Additionally, the model has two Dense layers: one with 32 units and another single unit for the final output. It's trained using the 'adam' optimizer and 'mean_squared_error' to measure errors over5 epochs, handling 32 data points at a time from 'x_train' and 'y_train' datasets.

Table 3. Represents the RMSE and MSE of the GRU model

Rooted mean square error (RMSE)	Mean square error (MSE)
4.61391359	21.288198

The GRU model's accuracy is done by comparing its predictions on test data with the actual values, computing RMSE for average prediction differences and MSE for average squared differences in closing prices shown in Table 3 . These metrics aid in understanding the model's performance in predicting continuous data such as stock prices.

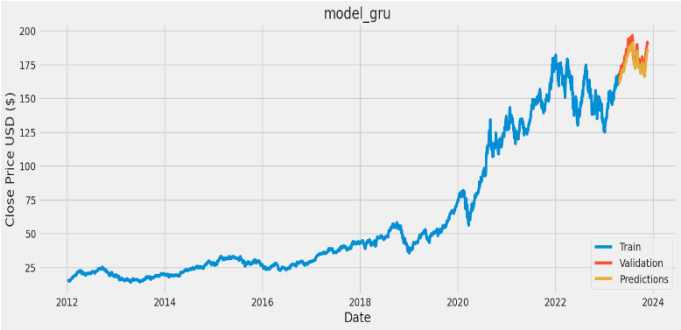

Figure 12: Representation of train , validation and prediction over time using GRU model

The above figure 12 , has the graph (size: 16x6) showing how well 'model_gru' predicts stock prices over time. It displays the 'Close' prices during training ('Train'), the real prices from validation ('Validation'), and the predicted values ('Predictions') by the model. This visualization helps in understanding how accurately the model predicts actual prices across various time periods. The figure 13 represents the difference between predicted price and stock close price of GRU model.

Date	Close	Predic__tions
2023-04-24	165.330002	163.595978
2023-04-25	163.770004	163.196747
2023-04-26	163.759995	162.396271
2023-04-27	168.410004	161.857651
2023-04-28	169.679993	163.406250
...	...	...
2023-11-16	189.710007	183.193344
2023-11-17	189.690002	184.412216
2023-11-20	191.449997	185.129105
2023-11-21	190.639999	186.161652
2023-11-22	191.309998	186.470261

[149 rows x 2 columns]

Figure 13. Representation of predicted price vs close stock price of GRU model

Comparison Between The Three Models

The table 3 clearly shows the difference between the models namely simple RNN , LSTM and GRU having their RMSE's and MSE's compared among them.

Table 4. Represents comparison between three models

	simple RNN	LSTM	GRU
RMSE	8.992649	5.5834700	4.61391359
MSE	80.86775119	31.1751380	21.288198

Implementation

The code has run successfully giving a significant improvisation between the three models and the source code for the proposed model for butterfly image classification can be found at:
https://github.com/vin66788/Deeplearninga1896845final.git

4. Conclusion

The study concentrated on the prediction analysis through three model found to be improved and the code also runs successfully giving better prediction outcomes along with minimum rooted mean square error. The predictions show a good result and when compared with each model the LSTM and GRU model showed best results than the simple RNN model giving maximum RSME value, and prior to this research the data keeps on changing on a regular basis , this helps to analyze and predict as most of the data can be compared to the previous data.

In the future, the research could explore different techniques for adjusting hyperparameters to determine the ideal number of hidden layer units and stacked layers. This exploration aims to better capture fluctuations in past data. Additionally, using time intervals shorter than 7 days might lead to improved outcomes.

References

1. Saud, AS & Shakya, S 2020, 'Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE', *Procedia Computer Science*, vol. 167, pp. 788–798
2. Zhu, Y 2020, 'Stock price prediction using the RNN model', *Journal of Physics: Conference Series*, vol. 1650, no. 3, pp. 32103-.
3. Sequeira, S & Banu, PKN 2021, 'Comparisons of Stock Price Predictions Using Stacked RNN-LSTM', in *Communications in Computer and Information Science*, vol. 1483, Springer International Publishing, Cham, pp. 380–390.
4. Mittal A, Goel A. Stock prediction using twitter sentiment analysis[J]. Standford University, CS229 (2011 http://cs229.stanford.edu/proj2011/GoelMittalStockMarketPredictionUsingTwitterSentimentAnalysis. pdf), 2012, 15.
5. Si J, Mukherjee A, Liu B, et al. Exploiting topic based twitter sentiment for stock prediction [C]//Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2013: 24-29.
6. Ariyo A A, Adewumi A O, Ayo C K. Stock price prediction using the ARIMA model [C]//2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation. IEEE, 2014: 106-112.