Using neural network models to predict stock prices

-CISC452 Group 26

Date: Dec.12th.2021

Group Members

Melissa Sun	(17ys65@queensu.ca)	20093324
Xiangyu Shi	(17xs2@queensu.ca)	20071873
Sifan Zhu	(17sz36@queensu.ca)	20091314
Chen Dan	(18cd11@queensu.ca)	20133399

1 Problem description, motivation, and contribution of each member

The financial market plays an important role in the world. With the development of science and technology, it is possible to predict the stock price. However, the financial market and stock rates are affected by national sentiments, policies, economics, and multiple psychological, emotional, and human factors, these predictions often fail to achieve the desired output (Aadesh Mallya, 2021).

The stock prediction has been a hot topic for a long time, and due to many reasons, such as the limitations of the device, only tiny people could do some research and have some solutions for this study (Chen, S., & He, H.,2018). Fortunately, the rapid rise of technology makes this topic easier for people to analyze the previous data and predict the future trend of stock prices by using different ways now, including using artificial neural networks (ANN).

To deal with this stock prediction problem, different models are built and analyzed to get the conclusion of which model could forecast the stock price more accurately. Compared with existing learning-based methods, the accuracy of our models is demonstrated by using the New York Stock Exchange dataset using the Mean Square Error (MSE) and Mean Absolute Error (MAE) (Z. Wang, S. Ho and Z. Lin, 2018). The reduction of MSE and MAE shows that there is an improvement in model performance.

Both Chen Dan and Xiangyu Shi implement LSTM but with different layers. While Sifan Zhu is using GRU and Melissa Sun builds her model with RNN. All of us implement different models and help each other to understand theirs.

2 Data description and preparation

Since there are approximately 8,51,266 records consisting of stock price records of 501 companies in this dataset, we chose “Netflix” to predict the stock price. We focus on the Close price as the value of prediction and the final output. Date, open price, volume, low, high are the input features for the model. After 4 CSV files “fundamentals.csv”, “prices-split-adjusted.csv”, “prices.csv”, and “securities.csv” are downloaded and unzipped, we import “prices-split-adjusted.csv” to implement the model, and the date is used for the index. Then move the close price to the last column and drop the null values in the dataset to make it readable and ready for future processes. Using sklearn preprocessing method, min_max_scaler that normalizes all the data. Finally split the dataset to 80% for training and 20% for testing.

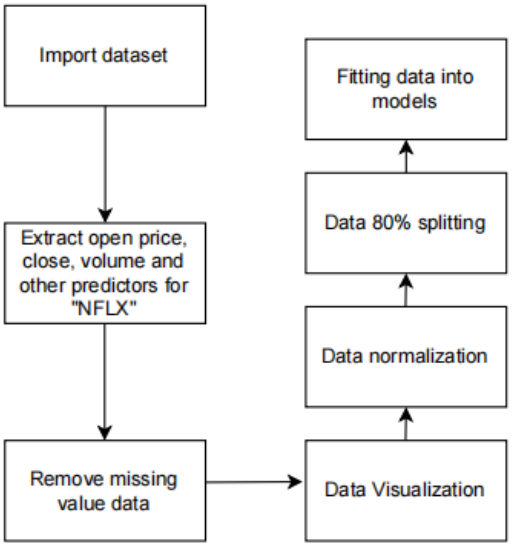

Fig 1. FlowChart of Data Prepossessing

We used the same code for data preprocessing so that we could use the same data and compare them at the end (ani1cr7, 2017). The visualized procedure is in the above figure 1.

3 Model Implementation

3.1 LSTM I

Xiangyu Shi(20071873/ 17xs2) implemented Stock Prediction with the LSTM(or long-term short-term memory network) model, a neural network that is responsible for finding the dependence between observations in a time series. Therefore, it is used for forecasting purposes especially for unstable time series that have chaotic factors. The stock market is essentially a chaotic and highly unstable time series(Aadesh Mallya, 2021). LSTM is a variant of RNN that could deal with vanishing gradient problems. The difference between the LSTM model and the typical RNN model is LSTM is able to store information over a period of time, in other words, they have a memory capacity. This characteristic is extremely useful in Time-Series or Sequential Data. We are free and able to decide what information will be stored and what discarded by using LSTM.

Creating an LSTM model using the built-in LSTM in TensorFlow looks like figure 1. It is a four-layer neural network that has two LSTM layers and two Dense layers. The first two layers have 256 neurons and for the rest two dense layers, there are 32 and 1 unit respectively. The dimension for inputs is five which is based on the features of stock and the close price is the forecasting data. Based on the hyperparameter testing in BenjiKCF(BenjiKCF, 2018) that gives out the suitable parameters for the stock prediction

problem. By testing different unit sizes from 512, 256, 128, 64, 32, I figured out LSTM with 256 units has the best performance. The memory days for this model is five which is a period of stock market opening in a week. The training data will be fitted in the model and trained in batches of 512 units, over 100 epochs that have a good convergence speed avoiding the underfitting. The dropout with the value 0.2 happens in the model after each LSTM layer. The dropout prevents the overfitting in this kind of large-scale neural network and makes it more efficient and less time-consuming. The first dense layer with the rectified linear unit as activation function is commonly used for a regression problem. The 'Adam' optimizer is chosen for compilation and a linear activation in the last layer as it is the fully connected layer that outputs the predicted stock price.

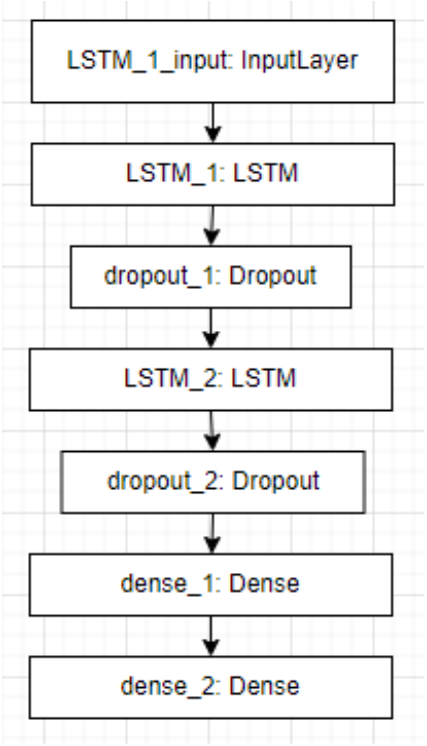

Fig 2. The Architecture of LSTM model

The NYSE dataset downloaded from Kaggle and adjusts the data first so that NFLX is chosen to be used. The dataset is then split into a training and testing set that 80% of data will be used for training the model and 20% will be used for testing/validation.

3.2 LSTM II

ChenDan (20133399/ 18cd11) implements a Long Short -Term Memory (LSTM) model with different layers for forecasting stock prices by utilizing the Keras package. Recurrent Neural Network is a powerful tool for tackling time series problems (Lipton, Z. C., Berkowitz, J., & Elkan, C., 2015). As a variant of the RNN model, LSTM leverages memory cells rather than a conventional artificial neuron. These special memory neurons are able to effectively

associate memories and input remotely in time (Chen et al., 2015). In the following models, “NFLX” is selected as our target company. We could consider data from NFLX from 2010 to 2016 as an order of sequence. The model uses previous sequence length days’ information to predict the (sequence length + 1)th day’s close price.

For the initial version, we chose 7 as sequence length, which means the model will use the previous 7 days’ information to predict the 8th day’s close price. The model uses two LSTM layers with 256 and 128 units respectively and a linear activation function. The output layer has one prediction value.

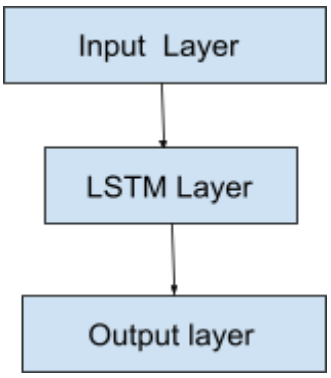

Fig 3. The Architecture of the LSTM model

After splitting the processed data into 80% training data set and 20% test data set, we fit the data into the model. Mean absolute value is applied as our evaluator. After 90 iterations, our MAE for this Long Short Term Model (LSTM) is 0.0112.

In the initial version, we randomly select the number of hidden layers, number of hidden layer units, and days that the model needs to remember. In order to improve our model, an optimizer is added to try different combinations of sequence length, number of units for different layers followed by relu activation function.

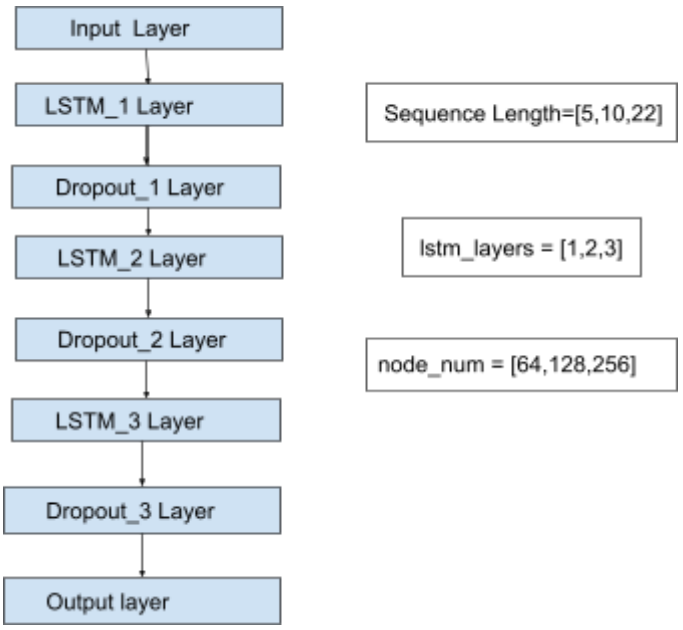

Fig 4. The Architecture of the LSTM model for stock prediction

The optimizer allows us to input a list of various numbers for different features. For example, we choose sequence_length = [5, 10, 22], as the stock market opens 5 days a week, and approximately 22 days per month. LSTM _ layers = [1,2,3] means the model contains at least1 LSTM layer and at most 3 LSTM layers. The figure on the left shows a model with 3 LSTM layers. For each LSTM layer, the optimizer will substitute 64 units, 128 units, and 256 units for each iteration.

3.3 GRU

Sifan(20091314/ 17sz36) implemented Stock Prediction with GRU(Gated Recurrent Unit) model. GRU is a recurrent neural network, improving the memory capacity of a recurrent neural network. GRU is quite similar to LSTM, but with fewer parameters. GRU does not have an output gate. GRU uses fewer training parameters and less memory, which executes more faster and efficiently than LSTM. GRU has better performance with shorter sequences (Phi, 2020).

GRU contains tanh activation and sigmoid activation. Tanh activation always has the value from -1 and 1, helping to regulate and control the values that flow through the network, otherwise, the value may turn extremely large after a couple of math operations. Sigmoid activation controls the value between 0 and 1, which helps determine if values are forgotten or kept. Since any number multiplied by 0 is 0, which means the number is “removed”, no more need for the future. As each value times, 1 is still the number itself, which represents the number kept. The reset gate decides how much input and previous output to be passed to the next cell, and the update gate is used to determine how much of the past information to

forget. The current memory content ensures that only the relevant information needs to be passed to the next iteration, which is determined by the weight W(Phi, 2020).

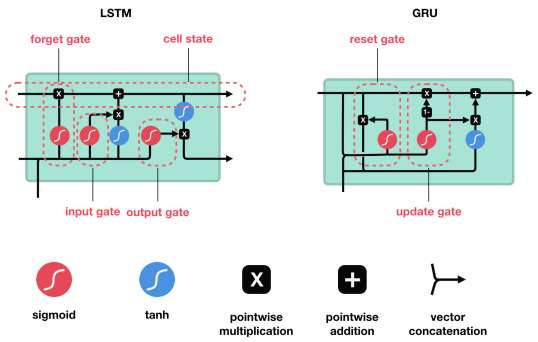

Fig 5. Model Structure of GRU compared with LSTM

We first obtain the data from Kaggle and unzip the values. “NFLX” is the company we use to predict the price, focusing on the close price to predict future prices. To complete the task, start with data preprocessing. The length=50, which is the 50th day of close price value, is selected to predict the 51st day's price. Also, normalize the data to the value between 0-1. For data split, 1150 pieces of records are selected for training data and 512 pieces of records as testing data.

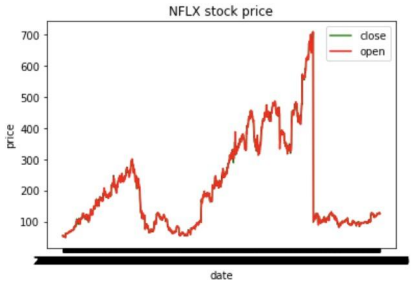

Fig 6. Plot NFLX stock price with the open and close price of NFLX stock price

As building up a GRU network, three layers with 50 cells each are applied, with the tanh activation function, and sigmoid recurrent_activation: model.add(GRU(50, activation='tanh', recurrent_activation='sigmoid')). The network has 40,215 parameters involved. Then validate the data by inputting the x_test and y_test. To demonstrate the result, display loss for training data, and val loss for validation. Finally, as there is one predicted data as output, test the data with model.add(Dense(1,kernel_initializer='uniform', activation = 'linear')). The ‘Adam’ optimizer is selected.

3.4 RNN

Melissa Sun (20093324/17ys65) implemented a simple RNN model, which is the recurrent neural network. It contains connections from output nodes to the hidden layer and/or input layer nodes. It allows interconnections between nodes of the same layer. Output from the previous step is fed as input to the current step. RNN has a “memory” which remembers all information about what has been calculated. Thus, it is useful in time series prediction due to its feature of remembering previous inputs.

The data of the company NFLX in the New York Stock Exchange are used to train and test the model. 80% of data is used as training data and the rest is testing data.

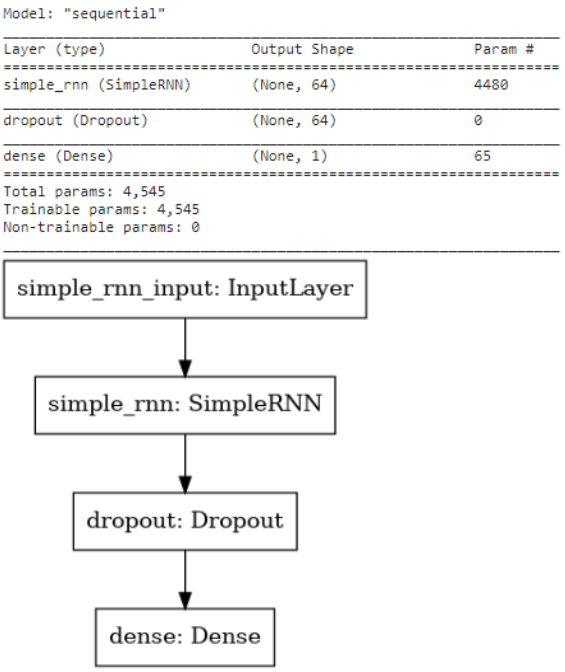

Fig 7. The architecture and structure of the RNN model

The model is built with the architecture in figure 6, which contains one simple RNN layer with 64 neurons, a dropout with 0.2 and a dense layer with linear activation function and random weights. I tried for different numbers of neurons, and it turns out that 64 has a good performance. I also tried to add some more layers, the result shows that there wasn’t much difference between one layer and two layers. Using Adam optimizer and adjusting it with a learning rate of 0.0005. After a few times of trying, 0.0005 has a better performance than 0.01 and 0.001. Mean absolute error is used to compile the model.

Building the model with input 5 and sequence length of 22. The reason is that the stock

market opens for 5 days a week and approximately 22 days a month. Fit the model the training data, a batch size of 128 and epoch of 20. The batch size and epoch number are both adjusted a few times, 128 batch sizes and 20 epochs return the best result on the mean squared error and mean absolute error.

4 Results

4.1 LSTM Model 1

In this report, a commonly used measure, mean squared error (MSE), Mean absolute error (MAE) and Root-mean-square error(RMSE) are provided to evaluate the accuracy and convergence time of the models. Dealing with this kind of large dataset, it is hard to check every value to see if there are any outliers. Therefore looking at the ratio of MAE to RMSE can help us to find out if there are large but uncommon errors.

Through the model score function in the model that found out the MSE and RMSE for this LSTM are 0.00027 and 0.02 respectively. The testing model has 0.00189 MSE and 0.04 RMSE.

The resultant graph of predicted results vs actual results is as follows:

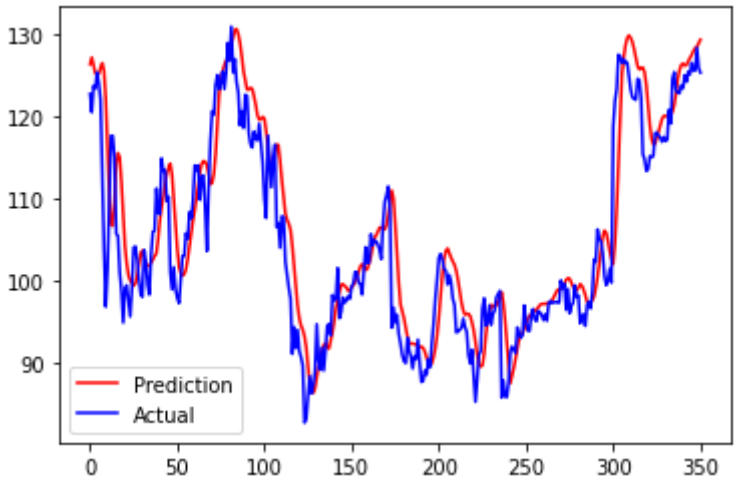

Fig 8. Plotted graph for Predict and Actual stock price for LSTM model 1

The graph shows us predictions of the last 350 days.

Figure 8 for the predicted values is almost exactly similar to that of the actual graph. In the stock market, more than the price of the stock, it is important to know if the prices are going to rise or fall and this graph gives a close approximation of that knowledge.

4.2 LSTM Model 2

The baseline model is 2 LSTM layers with 256, 128 units respectively, and followed by a linear activation function. Mean absolute value is applied(MAE) as our evaluator. After 90 iterations, the MAE for this Long Short Term Model (LSTM) is 0.0112.

After optimization, the best model is a 3-layers LSTM model with 256 units, 128 units, and 128 units respectively. The optimal days that the model needs to remember are 5. The Mean Square Error(MSE) and Root Mean Square Error(RMSE) of training data for optimized LSTM are 0.00033 and 0.02 respectively. The testing model has 0.00224 and 0.05MSE. The MAE for optimized models is 0.0110.

Model	MAE
LSTM model (baseline)	0.0112
LSTM model(with optimizer)	0.0110

The following graph of predicted results vs actual results illustrates our prediction of last 350 days:

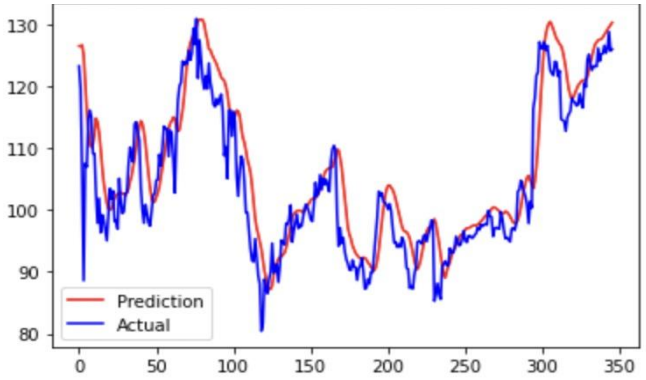

Fig 9. Plotted graph for Predict and Actual stock price for LSTM model 2

4.3 GRU

MAE and R2 are used to evaluate the outcome of the network. For the evaluator, we all applied Mean absolute value. After 100 epochs of training, and with the batch size of 64, with the built-in method, Mean absolute error (MAE) was calculated as 0.017, and Mean Square Error (MSE) was also used to evaluate as 0.00405. As the first epoch, the MAE is 0.286, but MAE decreases to 0.017 after 100 epoch training. With a decrease of MAE score, it represents the network showing positive performance. Both of the scores are matched with other members and the published research. Finally, visualize the plotted graph of its actual

and predicted stock price, the plotted predicted curve is pretty much matched with the actual price. We can conclude the network obtains precise accuracy with a low MAE score.

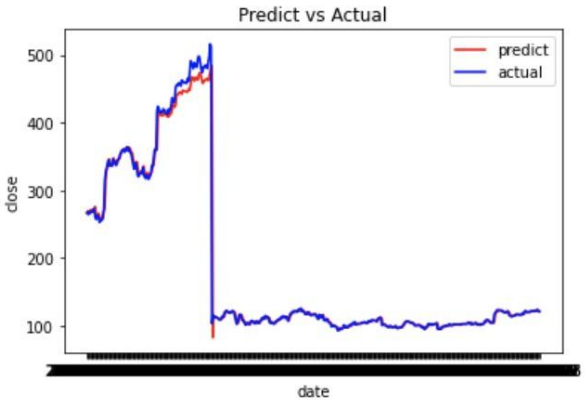

Fig 10. Plotted graph for Predict and Actual stock price for GRU

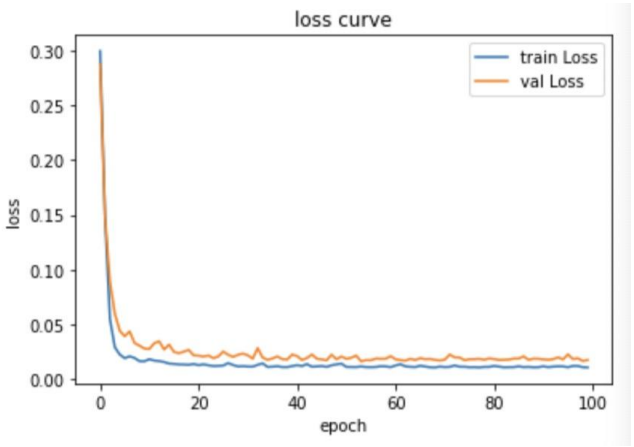

Fig 11. Plotted graph for training loss and val loss

4.4 RNN

After running 20 epochs, the result shows that the mean squared error and mean absolute error decreases as the epoch number increases. The MSE decreases from 0.0443 to 0.0015, and MAE decreases from 0.1386 to 0.0261. The reduction means that there is an improvement in the model.

Use predict function to get predicted value y_pred for testing dataset. Then use the built-in function to get the MSE and MAE for testing data, which MSE is 0.00266 and MAE is 0.0393.

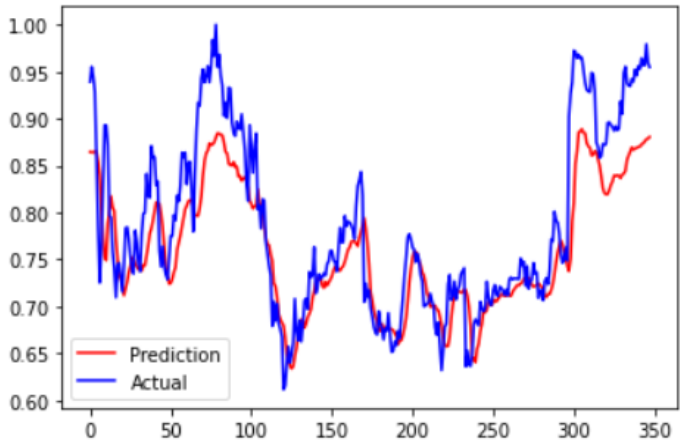

Fig.11. Plotted graph for Predict and Actual stock price for RNN model

Figure 11 is the plot of the predicted value and actual value. The red line is the predicted value, and the blue line is the actual value. From this figure, we could see that the simple RNN model could predict the value so that the predicted value plot could match the actual value well.

5 Discussion and open problems

Neural networks have developed from models mainly used in cognitive modelling and computational neuroscience to powerful and practical tools. Recurrent neural networks overcome many limitations of traditional machine learning methods on input and output data (Lipton, Z. C., Berkowitz, J., & Elkan, C., 2015). The models we build are all based on recurrent neural networks. LSTM and GRU are the variants of recurrent neural networks, and RNN is the direct application of recurrent neural networks.

When compared to some of the previous work in the literature, all our models manifest good results in predicting the future stock price based on past days' information. The best model on Kaggle uses the LSTM model with 6 layers (deadskull, 2017). Its mean square error of training data is 0.00019, and the mean square error of the test score is 0.00033.

Table I compares the results of our models and the best result on Kaggle. For comparison, the metric of mean absolute error is leveraged. Table II summarizes the results of the MAE for each model.

Model	Train MSE	Test MSE
Long Short - Term Memory (Model 1)	0.00027	0.00189
Long Short - Term Memory (Model 2)	0.00033	0.00224
Gated Recurrent Units	0.00016	0.00405
Simple Recurrent Neural Network	0.00040	0.00266
Best Model On Kaggle	0.00019	0.00033

TABLE I. COMPARISON BETWEEN LSTM, GRU, SIMPLE RNN AND PREVIOUS RESEARCH

Model	Mean Absolute Value
Long Short - Term Memory (Model 1)	0.0313
Long Short - Term Memory (Model 2)	0.0110
Gated Recurrent Units	0.0170
Simple Recurrent Neural Network	0.0393

TABLE II. COMPARISON OF MAE BETWEEN LSTM, DRU, SIMPLE RNN

For future modification to obtain a better performance, several points can be further explored:

1. Enrich the data: use more companies' data to make predictions
2. Exploring different combinations of input features, such as different sequence lengths, numbers of neurons for different layers
3. Exploring different combinations of activation functions

Lastly, the rapid success of recurrent neural networks on natural language tasks leads us to believe that extensions of this work to longer texts would be fruitful (Lipton, Z. C., Berkowitz, J., & Elkan, C., 2015). We hope what we got in this paper could help people who want to explore this area in the future to get some ideas and inspiration.

Reference

Aadesh Mallya(2021), “Echo State Networks and Existing Paradigms for Stock Market Prediction”, International Conference on Emerging Smart Computing and Informatics (ESCI).

ani1cr7. (2017). ANI1CR7/stock-price-prediction-using-LSTM-neural-networks. GitHub. Retrieved December 12, 2021, from https://github.com/ani1cr7/Stock-Price-Prediction-using-LSTM-Neural-Networks.

BenjiKCF. (2017). BenjiKCF/Neural-Net-with-Financial-Time-Series-Data: This solution presents an accessible, non-trivial example of machine learning (Deep learning) with financial time series using TensorFlow. Retrieved from https://github.com/BenjiKCF/Neural-Net-with-Financial-Time-Series-Data

Chen, S., & He, H. (2018). Stock prediction using Convolutional Neural Network.IOP Conference Series: Materials Science and Engineering, 435, 012026.

Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of china stock market. 2015 IEEE International Conference on Big Data (Big Data). https://doi.org/10.1109/bigdata.2015.7364089

Deadskull7. (2017). Deadskull7/New-York-stock-exchange-predictions-RNN-LSTM: Best score on Kaggle so far. mean square error after repeated tuning 0.00032. used stacked GRU + LSTM layers with optimized architecture, learning rate and batch size for best model performance. the graphs are self explanatory once you click and go inside !!! GitHub. Retrieved December 12, 2021, from https://github.com/deadskull7/New-York-Stock-Exchange-Predictions-RNN-LSTM

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019.

Phi, M. (2020, June 28). Illustrated guide to LSTM's and GRU's: A step by step explanation. Medium. Retrieved December 12, 2021, from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Wang, Z., Ho, S.-B., & Lin, Z. (2018). Stock market prediction analysis by incorporating social and news opinion and sentiment. 2018 IEEE International Conference on Data Mining Workshops (ICDMW). https://doi.org/10.1109/icdmw.2018.00195