Google Stock Price Prediction

Shivangi Patel
The University of Adelaide

Abstract

Stock market prediction is considered to be one of the most difficult tasks in Machine learning, yet widely sought after. The huge potential of financial returns makes it lucrative for analysts and investors alike. In this study, we attempt Google stock price prediction using historical data with the aid of Gated Recurrent Units (GRU) neural network. We perform an extensive tuning of model hyperparameters and also evaluate the effect of wavelet denoising to forecast stock closing prices from one to ten days in future. The performance of our optimized model in stock price prediction of the test dataset is slightly worse, when compared to a naive (baseline) model.

1. Introduction

The challenges of stock price predictions can be attributed to their dependency on a multitude of factors, loosely categorized as fundamental, technical and market sentiment [1]. Fundamental factors drive stock prices based on a company's earnings and profitability from producing and selling goods and services.Technical factors relate to historical stock prices, supply *vs* demand and behavioural factors of traders and investors. Market sentiment is reflective of individual and crowd psychology driven by among several other factors - incidental, political and health news. Using text mining and NLP techniques, researchers have attempted to analyse sentiments from unstructured data like news reports and tweets [2] [3]. Many researchers believe that it is possible to obtain a reasonable indication of stock price movement for a short time frame by studying only historical trends. Because of their ability to process elements of previous data to forecast future data, Recurrent Neural Networks and its variant Gated Recurrent Unit (GRU) are widely employed in forecasting of time-series data. Figure 1 is a representation of basic architectures of vanilla RNN and GRU cell. The activation function in a hidden recurrent layer is typically a *tanh* function, while the activation for the output layer is selected based on the application. The feed forward operation of a vanilla RNN can be expressed mathematically as follows,

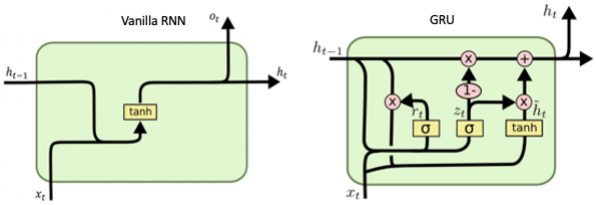

Figure 1. Basic architectures of Vanilla RNN and GRU cells (*Image courtesy [4]*)

$$h_t = \sigma_h(U_h x_t + V_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_h)$$

Where:
x_t : input vector $(m \times 1)$
h_t : hidden layer vector $(n \times 1)$
o_t : output vector $(n \times 1)$
b_h : bias vector $(n \times 1)$
U,W : parameter matrices $(n \times m)$
V : parameter matrix $(n \times n)$
σ_h, σ_y : activation functions

GRU is a specialized type of RNN which aims to solve the vanishing gradient problem observed with vanilla RNN, by using the *update gate* and *reset gate* vectors. Intuitively, the *update gate* helps the model to decide how much of the past information needs to be passed along to he future; while the *reset gate* decides on how much of the past information to forget. Mathematically, the feed forward operations of a GRU cell can be expressed as follows,

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$
$$r_t : \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$
$$\widetilde{h_t} = tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$$
$$h_t = (1 - z_t) \odot h_t t - 1 + z_t \odot \widetilde{h_t}$$

Where:
x_t : input vector $(m \times 1)$
h_t : hidden layer vector $(n \times 1)$
b_z, b_r, b_h : bias vectors $(n \times 1)$
W_z, W_r, W_h : parameter matrices

σ, *tanh* : activation functions

Financial markets are prone to volatility and therefore, the analysing historical trends is difficult due to daily fluctuations. Fourier analysis has achieved remarkable success in images and signal processing, however it can only decompose frequency components irrespective of time. In contrast, Wavelet based transformation method is able to decompose and reconstruct time series data from different time and frequency domains [5, 6]. The basic mechanism of wavelet based denoising is to wavelet transform a signal, where the coefficients of the noise generated by wavelet decomposition are smaller than that of the signal. A threshold is selected to eliminate the noise and then the signal is reconstructed. In this study, we optimize a GRU model to predict closing prices using Google stock price data. We also examine the effect of applying wavelet transformation methods to denoise the input data and compare its performance with unprocessed inputs.

2. Background

With the rise in data mining techniques and machine learning algorithms in the past decade, the potential of deep learning; especially Recurrent Neural Networks in stock price prediction has attracted wide attention from analysts and investors. Zhao *et al* [7] predicted daily stock price direction movement using historical data of 180 representative Shanghai stocks by employing an attention based mechanism with RNN, LSTM and GRU models. The authors note that using deeper neural networks did not necessarily improve performance. The attention based GRU and LSTM models performed significantly better than the RNN model in predicting the movement of stock prices. Qiu *et al* [6] proposed a framework to predict the opening prices of stocks, wherein the data is processed through a wavelet transform and then used in an attention based LSTM model. The researchers used roughly 17 years of data; the S&P 500 Index, Dow Jones Industrial Average (DJIA) and Hang Seng Index (HSI) and evaluated GRU, LSTM, WLSTM (Wavelet denoised + LSTM) and WLSTM + Attention. Lowest RMSE was obtained with the WLSTM + attention model for all the three stock index datasets. Althelaya *et al* [8] evaluate a Multi Layer Perceptron (MLP), LSTM, bidirectional LSTM and stacked LSTM architectures for short (1 day ahead) and long term (32 day ahead) stock market prediction using 7 years of S&P Index data. The authors observed that both bidirectional and stacked LSTM networks produce better performance for prediction of short term prices as opposed to long term prediction results. Bao *et al* [5] proposed a three stage forecasting framework to predict one step ahead closing prices. In the first stage, a denoised time series is generated via discrete wavelet transformation. The authors used *Haar* wavelet basis function. in the second stage, daily features are extracted via Stacked Autoencoders (SAEs) in an unsupervised manner. Third stage comprises of feeding the extracted features in a LSTM model to generate a one-step ahead closing price output. The authors observe that using their framework outperforms the predictive performance of LSTM models in isolation.

3. Methodology

3.1. Data Preprocessing

Google stock price dataset from Kaggle [9] is used in this study. The train dataset comprises of records of *Open, High, Low, Close* prices and *Volume* of stocks from 3^{rd} Jan, 2012 upto 30^{th} Dec, 2016. We further split this dataset in a $80 : 20$ ratio, such that the last 20% of records from the original train dataset can be used as validation set. The test data comprises of 20 records 3^{rd} Jan, 2017 to 30^{th} Jan 2017.

3.1.1 Google stock split correction

It is observed that the *Close* prices are not adjusted for Google stock split in 2014 in the original data set. Therefore, a correction is applied by scaling the *Close* prices by a factor of 0.499 for records from 3^{rd} Jan, 2012 to 26^{th} Mar, 2014.

3.1.2 Scaling

The features in the train set are scaled from 0 to 1 by applying *MinMaxScaler()* function of *Scikit-learn*. The scaling operations learned from train set are applied to the validation and test sets.

3.1.3 Wavelet transformation

Wavelet transformation can be used to denoise financial time series data, as it has the ability to analyse frequency components simultaneously with time. Consequently, compared to Fourier transform which decomposes the frequency components irrespective of time, wavelet applications are preferred method of choice for time series data [5, 6] To denoise input data by wavelet transformation, the methodology used by Bao *et al* [5] is applied. Briefly, a multilevel decomposition is performed for the dataset, the approximation coefficients and details coefficients are extracteed. We evaluate *Haar* and *Daubechies* as wavelet basis functions. For each of the extracted details coefficients arrays, a non-negative garrote threshold is applied. The standard deviation of the detail coefficients is used as threshold. Finally, a multilevel reconstruction is applied to get the denoised features. As suggested by the authors, the wavelet transformation is applied twice to reduce the risk of overfitting.

3.1.4 Preparing input data

After scaling, the input data is prepared for a multistep, univariate prediction. All features are used in the X matrix of input data, while considering that traders would be interested in the prediction of *Close* price, only *Close* price is used as input y. We figured most traders would be interested in bulk buying or selling at the end of a week. Therefore, the input data are prepared with a *timestep (lookback)* period of 22 (working days in a month) and *lookahead* period of 5 (working days of a week). A lookahead period of 10 days is also evaluated. The input data is constructed with a sliding window by a *stepsize* of 1.

3.2. Naive Model

To evaluate the performance of neural networks, a naive (baseline) model is designed. A naive model prediction for a particular day is the *Close* price equivalent to the last seen closing price. Therefore, for one-day ahead forecasts, the closing price of the previous day, *i.e.* day_{t-1} is output. Similarly, for a ten-day forecast corresponding to day_t, the model outputs the closing price of day_{t-10}.

3.3. GRU architectures

Keras library is used for implementation of GRU neural networks. Table 1 summarizes the architectures of the three models used in this study.

GRU Model	# Hidden Layers	# Hidden units per layer	# Total Parameters
Model1	1	50	23,901
Model2	1	100	92,801
Model3	2	50	39,201

Table 1. Summary of network architectures

Tuning of hyperparameters like number of layers, hidden units, learning rate, choice of activation function and batch size is performed and evaluated by *Root Mean Squared Error* (RMSE) of predictions for one-day ahead forecasts. All models are allowed to train for 50 epochs with implementation of early stopping callback and a patience setting of 5.

3.4. Model output processing

The models used in this study output a multistep, univariate prediction corresponding to each set of inputs. Since we are using *lookahead* periods of 5 and 10, the corresponding outputs are obtained in blocks of 5 and 10 respectively. The model outputs are rearranged to compare all forecasts for each day against the actual closing price for that day. For example, for the period of 5 days, the model output is rearranged such that for each date, the corresponding forecasts *i.e.* day-1, day-2, day-3, day-4 and day-5 can be compared against the actual closing price for that date. Since the *lookback* (timestep) is 22, the models output predictions for day 23rd onwards. It can be noted that for the 23rd day, the model can only output day-1 forecast. Similarly for 24th day, only day-1 and day-2 forecasts are obtained, and so on. From the 27th day onwards, all five forecasts are obtained for each day.

4. Results and Discussion

In this study, we aim to predict Google stock closing prices using historical data with features *Open, High, Low, Close* and *Volume*.

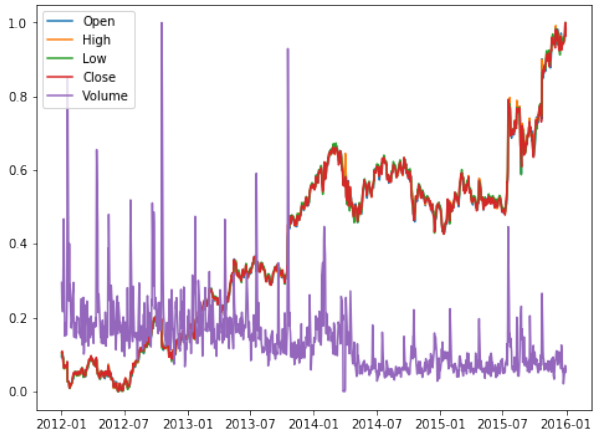

Figure 2. Trend of features over the period from 2012-2016

Figure 2 shows the trend of all features over the period from Jan, 2012 to Jan, 2016, with the features being scaled from 0 to 1. The price features have a considerable daily fluctuation, but have gradually risen over the period of four years. The trend of *Volume* is slightly in the downward direction, with the daily numbers fluctuating wildly. Model1 (1 hidden layer, 50 units per layer) is first evaluated for its prediction performance, using *Adam* optimizer, default learning rate (0.001), *tanh* activation function (*Keras* default for GRU models) batch size of 32, 50 epochs with early stopping callback (patience parameter of 5).

Forecast day	Validation set RMSE	
	Baseline Model	Model1
day-1	8.24	15.44
day-2	11.92	13.2
day-3	15.49	14.76
day-4	17.88	16.4
day-5	19.78	18.4

Table 2. RMSE obtained with baseline *vs* Model1

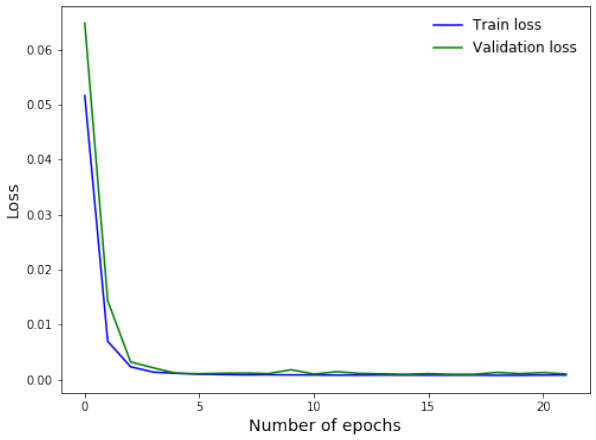

Figure 3. Loss curves during training of Model1

The train and validation loss curves in Figure 3 show a sharp drop with no indication of overfitting during model training. Table 2 compares the RMSE for day-1 to day-5 forecasts of validation set, with corresponding RMSE obtained using baseline model. The performance of Model1 for day-1 and day-2 forecasts is worse than the baseline model. There is a marginal, although insignificant reduction in error for day-3 to day-5 forecasts relative to baseline model. The poor performance of the model despite the sharp drop of loss curves in Figure 3 indicate that there is perhaps little room for improvement in model training. Nevertheless, we monitor the effect of hyperparameter tuning. We first optimize the learning rate, by ranging from multiples of 0.1 to 15 with the default learning rate of 0.001 (Table 3).

Learning Rate	Validation set RMSE for day-1 forecasts
0.0001	17.05
0.0005	11.44
0.001	11.52
0.005	9.05
0.0075	9.15
0.01	9.08
0.0125	**8.98**
0.015	9.39

Table 3. Optimization of learning rate

Lowest error is obtained with learning rate of 0.0125. Increasing the depth of models or number of hidden units produce no improvement in performance (Table 4). Table 5 shows the evaluation of different activation functions, with the *elu* activation function giving lowest error.

Lastly, batch sizes are tested (Table 6), with batch size of 128 showing optimal performance. We also evaluate the effect of denoising of the train set using wavelet transformation method. Figure 4 shows the denoised closing price plots of *Daubechies* and *Haar* wavelet basis functions, overlaid with the plot of actual closing price. As expected, the denoised plots show reduced daily fluctuation, with the overall trend remaining similar to the actual closing price trend. However, no improvement in model performance is observed upon training with denoised data (Table 7). Table 8 compares the RMSE obtained for day-1 to day-5 forecasts of validation set using the optimized model and the baseline model. The optimized model shows improvement in performance for day-1 and day-2 forecasts when compared to Model1 in Table 2, however, it still has little to no benefit over the baseline model for corresponding days.

Model architecture	Validation set RMSE for day-1 forecasts
Model1 (1 hidden layer, 50 units per layer)	**8.98**
Model2 (1 hidden layer, 100 units per layer)	9.91
Model3 (2 hidden layers, 50 units per layer)	9.62

Table 4. Optimization of model architecture

Activation function	Validation set RMSE for day-1 forecasts
tanh	8.94
sigmoid	9.90
relu	9.71
elu	**8.57**
selu	8.01

Table 5. Optimization of activation function

Batch size	Validation set RMSE for day-1 forecasts
16	9.38
32	9.12
64	8.93
128	**8.65**

Table 6. Optimization of batch size

Wavelet basis function	Validation set RMSE for day-1 forecasts
No denoising	8.98
Daubechies	13.78
Haar	22.52

Table 7. Prediction performance with Wavelet denoising

A marginal improvement is seen for day-3 to day-5 fore-

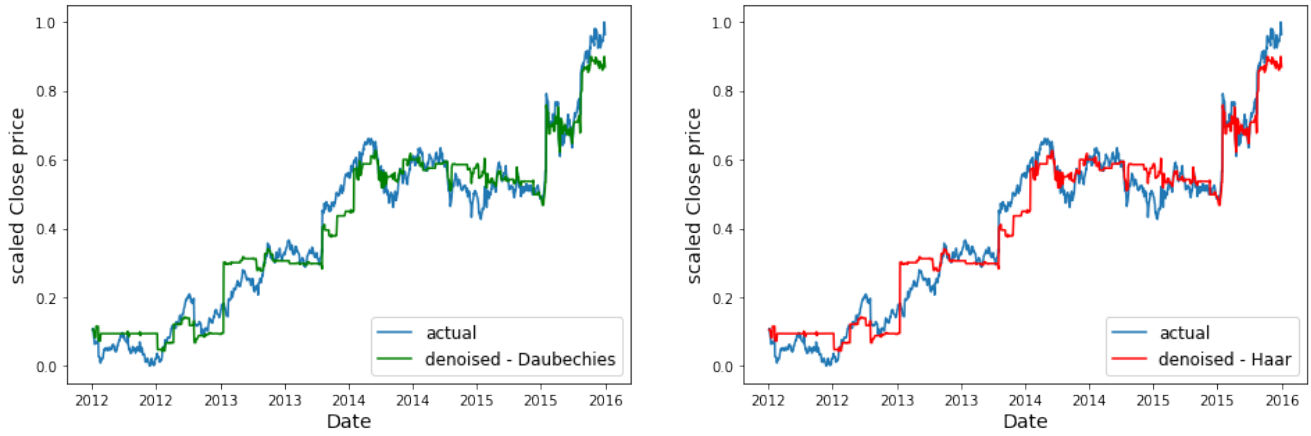

Figure 4. Wavelet denoised plots of Close price

Forecast day	Validation set RMSE	
	Baseline Model	Optimized model
day-1	8.24	8.65
day-2	11.92	11.85
day-3	15.49	14.56
day-4	17.88	16.36
day-5	19.78	17.84

Table 8. Baseline *vs* optimized model (lookahead=5)

Forecast day	Validation set RMSE	
	Baseline Model	Optimized model
day-1	8.33	12.81
day-2	12.04	14.82
day-3	15.66	15.11
day-4	18.08	16.81
day-5	20.00	18.29
day-6	21.39	19.64
day-7	22.83	20.91
day-8	23.97	22.03
day-9	24.83	23.13
day-10	25.51	24.00

Table 9. Baseline *vs* optimized model (lookahead=10)

casts. We speculate that the model may perform slightly better relative to the baseline, for farther ahead forecasts and therefore retrain the optimized model using *lookahead* period of 10.

A similar trend in prediction performance is observed with the retrained model (Table 9). Figure 5 shows a daily snapshot of error comparisons between the optimized and baseline models for day-1 and day-10 forecasts. The errors obtained with the optimized model are higher for day-1 forecasts, but consistently lower than baseline for day-10

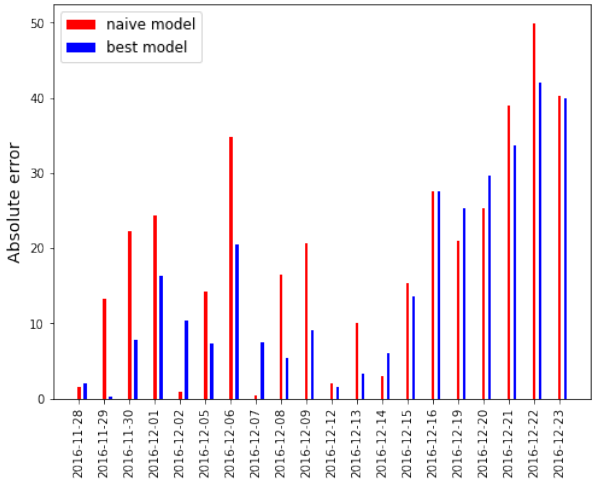

Figure 5. Daily snapshot of errors for day-10 forecasts

forecasts.

We finally evaluate the test set using optimized model forecasting 10 days ahead. Contrary to our expectations, the optimized neural network model shows a worse performance compared to the baseline model, even for day-10 forecasts.

5. Conclusion

In this study we evaluate GRU neural networks for Google stock price prediction using historical data from 2012 to 2016. We use all features *Open, High, Low, Close* and *Volume* of stocks with a timestep (*lookback*) period of 22 days to forecast closing prices for *lookahead* periods of 5 and 10 days. Even with an extensive hyperparameter tuning and evaluation of wavelet denoising for all features, we observe no significant benefit of prediction performance or sometimes even worse compared to a baseline model. A

Forecast day	Test set RMSE	
	Baseline Model	Optimized model
day-1	6.28	10.99
day-2	8.75	6.97
day-3	11.45	15.08
day-4	14.38	17.48
day-5	16.03	18.61
day-6	17.03	20.53
day-7	17.92	23.11
day-8	18.61	24.82
day-9	18.88	26.14
day-10	17.83	26.19

Table 10. Baseline *vs* optimized model - test set

steep drop in train and validation loss curves approaching towards a 0 loss is observed during model training. The poor performance of the model despite almost a 0 loss indicates that there is insufficient information in the dataset for the model to learn. We conclude that studying only historical trends over a short time frame is not beneficial for stock price prediction. Perhaps a more nuanced approach would be to collect fundamental data on market sentiments and use in combination with historical trends to achieve a better performing model.

6. Github

For code, please refer to Github.

References

[1] "Forces that move stock prices." [Online]. Available: https://www.investopedia.com/articles/basics/04/100804.asp

[2] F. Xu and V. Keelj, "Collective sentiment mining of microblogs in 24-hour stock price movement prediction," in *2014 IEEE 16th Conference on Business Informatics*, vol. 2. IEEE, 2014, pp. 60–67.

[3] A. Porshnev, I. Redkin, and A. Shevchenko, "Machine learning in prediction of stock market indicators based on historical data and data from twitter sentiment analysis," in *2013 IEEE 13th International Conference on Data Mining Workshops*. IEEE, 2013, pp. 440–444.

[4] "Vanilla rnn and gru basic architectures." [Online]. Available: http://dprogrammer.org/rnn-lstm-gru

[5] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PloS one*, vol. 12, no. 7, p. e0180944, 2017.

[6] J. Qiu, B. Wang, and C. Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism," *PloS one*, vol. 15, no. 1, p. e0227222, 2020.

[7] J. Zhao, D. Zeng, S. Liang, H. Kang, and Q. Liu, "Prediction model for stock price trend based on recurrent neural network," *JOURNAL OF AMBIENT INTELLIGENCE AND HUMANIZED COMPUTING*, 2020.

[8] K. A. Althelaya, E.-S. M. El-Alfy, and S. Mohammed, "Evaluation of bidirectional lstm for short-and long-term stock market prediction," in *2018 9th international conference on information and communication systems (ICICS)*. IEEE, 2018, pp. 151–156.

[9] "Google stock price dataset." [Online]. Available: https://www.kaggle.com/rahulsah06/gooogle-stock-price