Using RNNs in Stock Prediction

Meng-Hsuan Wu

Abstract

The direction of the financial market is always stochastic and volatile and the return of the security return is deemed to be unpredictable. Analysts now are trying to apply the modeling techniques from Natural Language Processing into the field of Finance to the similarity of having the sequential property in the data. In this research, we have constructed and applied the state-of-art deep learning sequential model, namely Long Short Term Memory Model (LSTM) and Gated Recurrent Unit (GRU) and Stacked LSTM/GRU, to the prediction of stock prices on the next day. Our input data contains traditional end-day price and trading volumes, The result has shown that GRU beats LSTM other models in terms of prediction error. Furthermore, we discovered that the stacked-RNN model does not improve the predictive power over single layer, even though it has more complex model structure.

Introduction

In the field of quantitative trading, the pivotal focus revolves around predicting future security returns, serving as the cornerstone of the industry. The formulation of future trading strategies hinges upon the industry's outlook on the financial market. Within the trading domain, fundamental analysis and quantitative trading are employed as two primary methods. The fundamental approach bases trading decisions on subjective views of the industry or company's future, relying on public information such as market news and financial statements. Conversely, quantitative trading leverages mathematical models, mitigating the impact of human subjectivity and emotion. Traditional quantitative strategies utilized methods like linear regressions, ARIMA, and GARCH models, proving effective in earlier financial regimes but diminishing in effectiveness with industry shifts. The quantitative trading industry has transitioned into the 'deep learning era,' with the prevalence of models like LSTM (LongShortTermMemory) in predicting stock returns. This project applies two common models of RNN(Recurrent Neutral Network), LSTM and GRU, to financial time series prediction for future price movements. The output from the deep learning model informs the further development of a quantitative trading strategy, whose returns are then compared with the market.

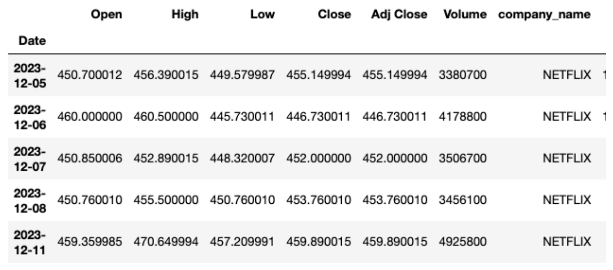
Related Work

The original idea to use LSTM to predict market stock price is inspired by [5]. [6] summarizes the design experience of using LSTM Recurrent Neural Network to predict stock market. [7] provides an architecture reference to build a time series forecasting model.More recently, deep learning methods have demonstrated better performances thanks to improved computational power and the ability to learn non-linear relationships enclosed in various financial features [3] Some previous work ([1-4][8-10]) has already discussed the potential approach to apply reinforcement learning in the equity market in recent years, but several challenges still exist:

- Real-world trading data is limited.
- The reward of reinforcement learning and features can be defined in multiple ways.
- Potential pattern may not be captured since machine training is only based on historical data.

Dataset and Features

We chose 6 stocks with top market capitalization in the S&P500 index from 2012 to 2023 as our dataset. The stocks selected are Microsoft, Amazon, Meta, Netflix, Google and Apple. The main reason of the pick is that these stocks make up over 20% of the S&P 500 — a staggering figure considering the S&P 500 is generally viewed as a proxy for the United States economy as a whole. The original data is fetched from Yahoo Finance API. Each dataset row is comprised of date, open price, high price, low price, close price, volume, and ticker symbol. The dataset has 16375 rows in total. We split the dataset into training and testing on a 80/20 basis. The sampled original dataset is presented in Table 1.



	Open	High	Low	Close	Adj Close	Volume	company_name
Date							
2023-12-05	450.700012	456.390015	449.579987	455.149994	455.149994	3380700	NETFLIX
2023-12-06	460.000000	460.500000	445.730011	446.730011	446.730011	4178800	NETFLIX
2023-12-07	450.850006	452.890015	448.320007	452.000000	452.000000	3506700	NETFLIX
2023-12-08	450.760010	455.500000	450.760010	453.760010	453.760010	3456100	NETFLIX
2023-12-11	459.359985	470.649994	457.209991	459.890015	459.890015	4925800	NETFLIX

Figure 1: Data structure of Netflix stock from 2012 to 2023

Methods

The main idea of RNN is to apply the sequential observations learned from the earlier stages to forecast future trends. Long-Short Term Memory (LSTM) model and Gated recurrent units (GRU) are updated versions of RNN. They can overcome the drawback of RNN in capturing long term influences. In order to examine the impact of predictive powers in different financial Time Series, we built 6 deep learning models:

- RNN with LSTM Model (LSTM)
- RNN with GRU Model (GRU)
- RNN with 2 Stacked-LSTM (Stacked-LSTM)
- RNN with 4 Stacked-LSTM (Stacked-LSTM)
- RNN with 2 Stacked-GRU (Stacked-GRU)
- RNN with 4 Stacked-GRU (Stacked-GRU)

Single/Stacked-LSTM:

The main idea of RNN is to apply the sequential observations learned from the earlier stages to forecast future trends. Long-Short Term Memory (LSTM) model is an updated version of RNN. It can overcome the drawback of RNN in capturing long term influences. LSTM introduces the memory cell that enables long-term dependency between time lags. The memory cells replaces the hidden layer neurons in the RNN and filters the information through the gate structure to maintain and update the state of memory cells. The gate structure includes input gate, forget gate and output gate. The forget gate in the LSTM determines which cell state information is discarded from the model, it accepts the output from the previous time step h_{t-1} and the new input of the current time step. Its main function is to record the amount of information reserved from the previous cell state to the current cell state. It will output a value between 0 and 1 where 0 means complete reservation and 1 means complete abandonment.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

The input gate determines how much the current time network input xt is reserved into the new cell state Ct, it avoids feeding the unimportant information into the current memory cell. It has three different components: 1) Get the state of the cell that must be updated; 2) Create a new cell state; 3) Update the cell state to the current cell state.

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t h_t$$

The output gate controls how much the newly created cell state will be discarded, the output information is firstly determined by a sigmoid layer, then the newly created cell state is processed by tanh, together with the sigmoid output to determine the final output.

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

Single/Stacked-GRU:

The GRU is like a long short-term memory (LSTM) with a gating mechanism to input or forget certain features, but lacks a context vector or output gate, resulting in fewer parameters than LSTM. GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be similar to that of LSTM.
The GRU has two gates and two hidden state:
Candidate Hidden State and Final Hidden State:

$$\hat{h}_t = \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$

Update Gate:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

The update gate helps the model determine how much of the past information (from previous time steps) needs to be passed along to the future. It is crucial for the model to capture long-term dependencies and decide what to retain in the memory.

Reset Gate:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r$$

The reset gate decides how much of the past information to forget. It allows the model to decide how important each input is to the current state and is useful for making predictions.

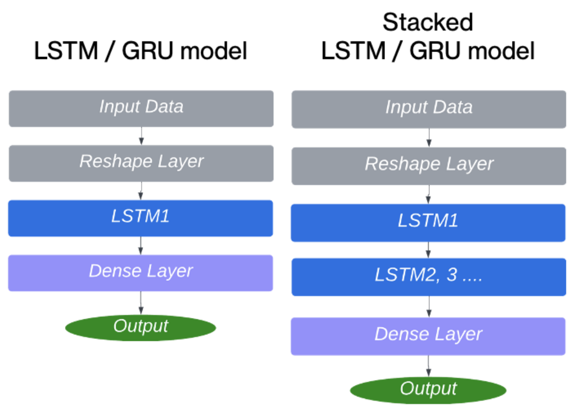


Figure 2: The architecture of deep learning models

Results

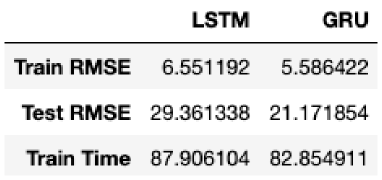
In this project, we will be using data from the past to predict the return on the next trading day. To assess the model, our primary model performance metric is Root Mean Squared Error (RMSE). The RMSE will be performed on the test data, which is totally unseen from the training and development stages. It is calculated between our predicted price and the true price.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

For the models, the tanh activation function was used for the LSTM layers and the sigmoid activation function was used for the output layers, these activation functions have been proved to be showing more superior results than other activation functions. Moreover, the dropout probability of 20% is also applied for each hidden layer as the usual regularization method to reduce the overfitting issue. Finally, the ADAM optimization is used for learning the parameters, and the mean squared error is utilized as the loss function. The lookback day is the numbers of days we use from the past to predict the future, in this research we have experimented a large number of look back days and eventually decided to use look back days of 64 days, meaning that we will be using the data from past 64 days to predict the stock prices in the next day

Each model was trained by 500 epochs. Generally, GRUs performed better than LSTM in regards to RMSE and training time in all layouts. We expected the Stacked-LSTM model can capture more stochasticity within the stock market due to its more complex structure. However, it was interesting that our experiments showed the opposite results. Typically, the deeper neural network is able to explain more complicated problem than single-larger neural network. However, we discovered is the fact that the stacked-LSTM does not significantly outperform the LSTM in the context of stock price prediction. Instead, the performance LSTM even beat the stacked-LSTM in certain instances. It has proved that the more complex representative does not necessarily improve the predictive power. It is possibly due to two reasons:
1. The more complicated neural network representation causes overfitting issue, the greater number of parameters in stacked-LSTM memory model does not generalize well in the unseen data.
2. The stacked-LSTM is more suitable in predicting classification problems rather than continuous time series like stock prices.

Single LSTM and GRU



	LSTM	GRU
Train RMSE	6.551192	5.586422
Test RMSE	29.361338	21.171854
Train Time	87.906104	82.854911

Figure 3: Performance of Single LSTM and GRU

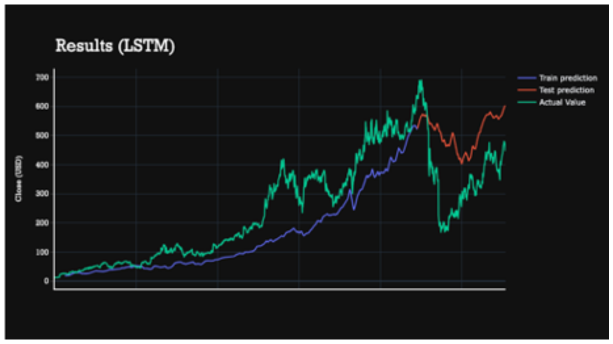


Figure 4: Results of Single LSTM

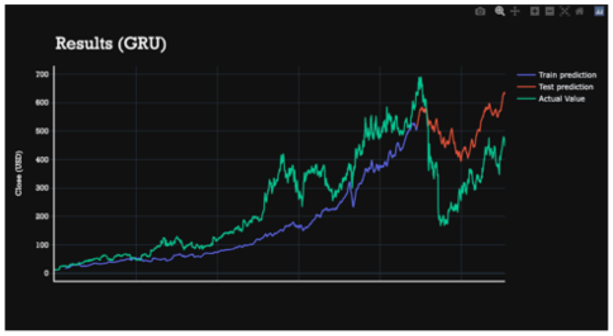
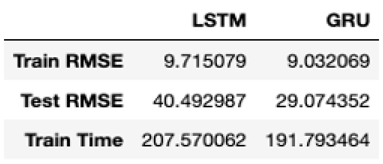


Figure 5: Results of Single GRU

Stacked LSTM and GRU



	LSTM	GRU
Train RMSE	9.715079	9.032069
Test RMSE	40.492987	29.074352
Train Time	207.570062	191.793464

Figure 6: Performance of Stacked LSTM and GRU - 2 Layers

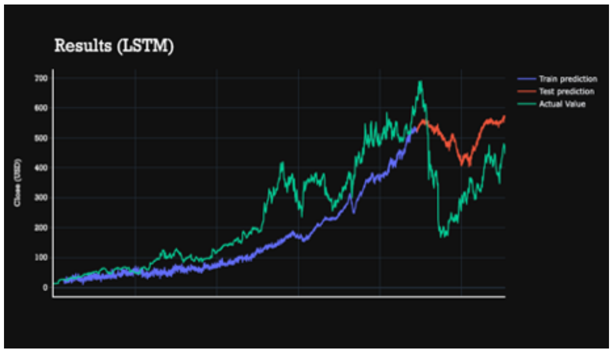


Figure 7: Results of Stacked LSTM - 2 Layers

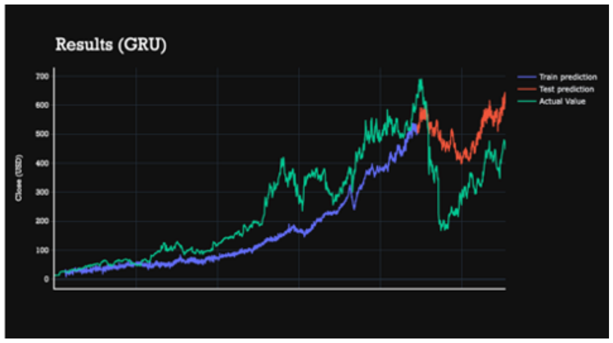
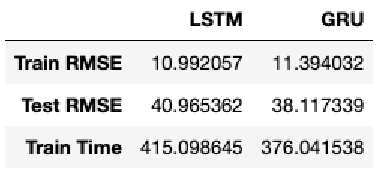


Figure 8: Results of Stacked GRU - 2 Layers



	LSTM	GRU
Train RMSE	10.992057	11.394032
Test RMSE	40.965362	38.117339
Train Time	415.098645	376.041538

Figure 9: Performance of Stacked LSTM and GRU - 4 Layers

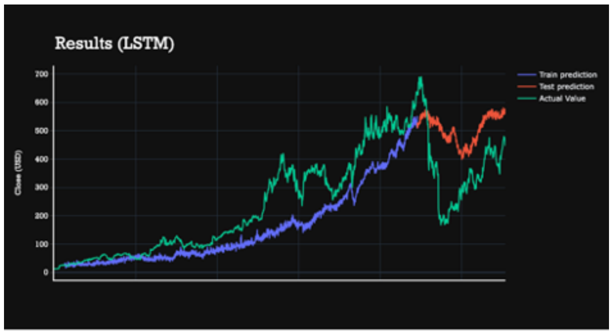


Figure 10: Results of Stacked LSTM - 4 Layers

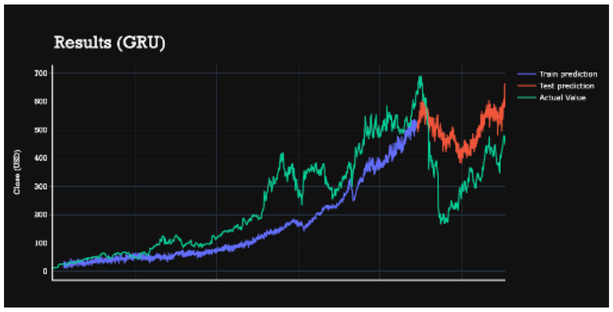


Figure 11: Results of Stacked GRU - 4 Layers

Conclusion and Future Work

This paper establishes a forecasting framework to predict the prices of stocks. We built up several variants of RNN models and compared the results. The results indicate that with fewer parameters and operations, GRU may perform better than LSTM. Furthermore, stacked RNNs don't necessarily link to better prediction in certain use cases due to the potential of overfitting. One direction of future work will be adding more features in consideration for the forecast. Another direction will be the implementation of more models such as CNNs or Attention-LSTM models introduced by google.

References

[1] Fischer, T. G. 1970. Reinforcement learning in Financial Markets - A Survey.

[2] Karpathy, A. 2015. The Unreasonable Effectiveness of Recurrent Neural Networks.

[3] Lee, C.; Jongdae, J.; and Sehwan, Y. 2007. Neural network model vs. Sarima model in forecasting Korean Stock price index (KOSPI). *Issues In Information Systems*.

[4] Liu, X.-Y.; Xiong, Z.; Zhong, S.; Yang, H.; and Walid, A. 2022. Practical deep reinforcement learning approach for stock trading.

[5] Moghar, A.; and Hamiche, M. 2020. Stock market prediction using LSTM recurrent neural network. *Procedia Computer Science*, 170: 1168–1173.

[6] SAYAH, F. 2023. stock market analysis + prediction using LSTM.

[7] TensorFlow. 2023. Time series forecasting.

[8] Théate, T.; and Ernst, D. 2021. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173: 114632.

[9] Wang, J.; Zhang, Y.; Tang, K.; Wu, J.; and Xiong, Z. 2019. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks.

[10] Yang, H.; Liu, X.-Y.; Zhong, S.; and Walid, A. 2020. Deep Reinforcement Learning for Automated Stock Trading: An ensemble strategy. *SSRN Electronic Journal*.

(6) (5) (7) (3) (10) (1) (4) (9) (8) (2)