Predict stock price with RNNs

Melinda Dong
University of Adelaide
North Terrance, SA 5000
a1870910@adelaide.edu.au

Abstract

This study implemented the classic time series RNNs (Recurrent neural network) models on S&P 500 ETF dataset. There are 3 round of study in total. First round experiments applied SMA(simple moving average) as baseline model and compared with classic RNN, LSTM(long short-term memory networks) and GRU(Gate Recurrent Unit) models and the results of different models were compared. The second round experiments based on GRU model, by tuning different hyperparameters, tries to find the effect on different sliding window, and different hidden layers. The third round of experiments is to test the adaptability of the model(using GRU model as sample) to different data. Study found out the RNN series model is not good for prediction stock market. Because the stock market is complex and the simple using price to predict price is not enough.

1. Introduction

The purpose of RNNs is to process sequence data. In the traditional neural network model, from the input layer to the hidden layer to the output layer, the layers are fully connected, and the nodes between each layer are unconnected. But this kind of neural network cannot handle sequence problems very well. However, the RNNs model will remember the previous information and apply it to the calculation of the current output, which makes the RNNs model better at dealing with sequence problems.

Since the 1980s, people have been trying to use machine learning to find some changing patterns in the stock market. Because the stock market is very complicated, Stock prices has a lot to do with people's fluctuating emotions. news, public opinions, politics, etc. will all affect stock prices. And there are often some black swan events, such as the outbreak of the epidemic [3].

for the purpose of learning, This study apply RNNs models to predict stock price. The main dataset is S&P 500 ETF. In the first round of study, The baseline model, SMA (simple moving average) with sliding window 20 gives the best result, with MSE(mean square error) 0.00054(All numbers in this article are kept to 5 decimal places). The RNN, LSTM and GRU models all can't beat the baseline model. With same hyperparameter setting, RNN has MSE 0.00185 on test data. GRU has MSE 0.00192 on test data and LSTM has 0.10295 on test data.

In second round of study, in order to better compare the influence of different hyperparameters on model performance, the experiment uses the GRU model as base model. On this basis, different sliding window and hidden layers were applied. In this case 20 sliding window with hidden layer 1 perform the best.

In the third round of study, the best GRU model from previous study were applied on other stock dataset to test the model adaptability. On OXY(western petroleum) the MSE on test data is 0.04368, On KO(Coca Cola) the MSE on test data is 0.09302, on TSLA(Tesla) the MSE on test data is 0.08637. compared with the preform on original SPY (MSE 0.00192 on test data), the result is not good. In general, RNNs model are not good enough for stock prediction and this needs more research.

2. Method description

This section will introduce the dataset's structure and the mathematics behind SMA. Then introduce the conception of autoregressive model, which was the inspiration for the RNN family of networks. Finally, the network architecture of classic RNN, LSTM and GRU will be clearly explained.

The loss function this study applied is the average mean square error, for convenience, the abbreviation MSE was used in following. The formula of MSE is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y_i}\right)^2$$

MinMax scaler was applied on data before model training. Because there are different evaluation indicators among the data, such as different units or different orders of magnitude. MinMax scaler can make all indicators in the same order of magnitude, which is convenient for comprehensive

comparison [9]. At the same time, MinMax scaler can make the process of finding the optimal solution smoother, make it easier to converge correctly, and improve the calculation accuracy. The formula of MinMax scaler is:

$$X_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2.1. Dataset

The data is gotten from Yahoo websitehttps://au.finance.yahoo.com/quote/SPY?p=SPY.

SPDR S&P 500 ETF Trust (SPY) is an ETF of S&P 500. S&P 500 (The Standard and Poors 500) is a stock market index tracking the stock performance of 500 large companies listed on stock exchanges in the United States. It can roughly represent the overall stock market trend in the US(see figure 1.).

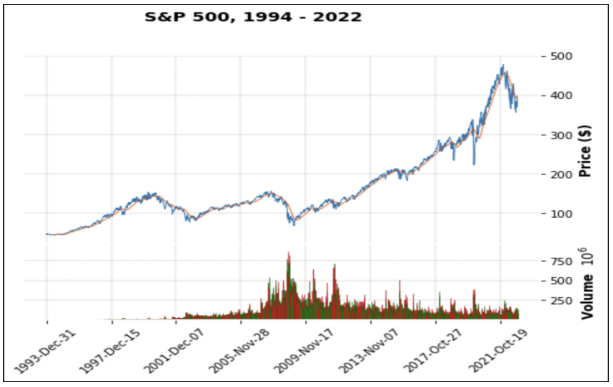

Figure 1. SPY dataset

The total dataset has 7270 samples, including the stock price information from 1993/12/31 to 2022/11/11. The red line in figure 1 is the SMA with silding window 100. Each sample has 7 features, this study only used 5 of them, they are date, open, high, low, close. The train-test split ratio is 8:2, the first 5800 samples are train dataset, the last 1450 samples are test dataset.

2.2. SMA

Moving average is a common tool for analyzing time series in technical analysis and is often applied to stock price series [8]. The moving average can filter high-frequency noise, reflect the low-frequency trend in the medium and long term, and assist investors in making investment judgments.

According to different calculation methods, popular moving averages include simple moving average, weighted moving average, exponential moving average, etc. Here the simple moving average is used as the baseline, and its calculation formula is as follows:

$$SMA = \frac{C_1 + C_2 + C_3 + ... + C_n}{n}$$

where:

C_n is the closing price on day n.

n is moving average period.

2.3. RNNs theory

In real life, many data are continuous sequences, for example, music, speech, text and video. If their sequence is rearranged, then the original meaning will be lost. For this kind of sequence data, Current data and previous observations are correlated.

Let the event x that be observed on time T as x_t, according to Bayesian rules, the probability of event x is :

$$P(x) = P(x_1) \cdot P(x_2|x_1) \cdot P(x_3|x_1, x_2) ... P(x_T|x_1, ..., x_{T-1})$$

To order to know x_t, we need information from x_1 to x_{t-1}. This model of modeling data that has already been seen is called auto-regression [1]. However, to calculate x_t to calculate all the data before time t is not very feasible in reality, and a lot of calculation will be wasted. To address this problem, autoregression models have been proposed. According to the Markov assumption, the current data is only related to the past n data. When predict x_t, instead of considering the data from x_1 to x_{t-1}, we can just consider the data from x_{t-n} to x_{t-1}.

$$P(x_t|x_1, x_2 ... x_{t-1}) = P(x_t|x_{t-n} ... x_{t-1})$$

Another solution is latent autoregression model, the key idea of latent autoregression models is that it introduces a latent variable h_t to represent past information.

$$h_t = f(x_1, x_2 ... x_{t-1}) \Rightarrow x_t = P(x_t|h_t)$$

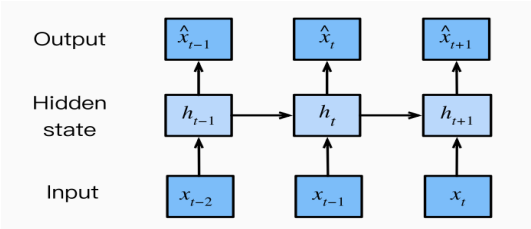

Figure 2. latent autoregressive model

And latent autoregression models are the prototype that originally inspired the RNN models.

2.3.1 The structure of basic RNN

A recurrent neural network (RNN) is a type of artificial neural network designed to recognize data's sequential patterns to predict the following scenarios. This architecture can simulate temporal dynamic behavior and use of feedback loops to process a sequence, which makes RNNs great for time series problems. The structure of basic RNN looks like this:

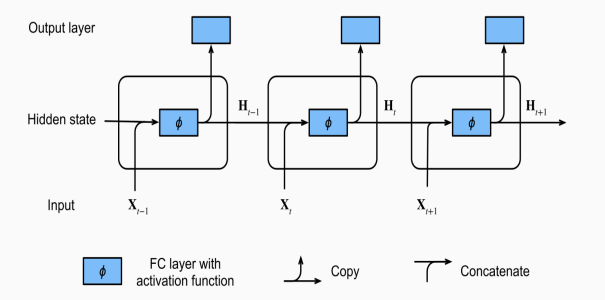

Figure 3. Simple RNN model

The general calculation process is to splice the input x_t of the current time step t and the hidden variable H_{t-1} of the previous time point and put the result into the activation function. Put this output to the fully connected layer to give the final output. Compared with the traditional CNN, it adds $H_{t-1}xW_{hh}$ to its hidden layer.

This variable represents all the historical information before the current step, also known as the memory of the current step.

Hidden layer:

$$h_t = \phi(W_{hh} \cdot h_{t-1} + W_{hx} \cdot X_{t-1} + b_h)$$

Output layer:

$$O_t = W_{ho} \cdot h_t + b_o$$

where: ϕ is the active function.

Based on the basic RNN model, deep RNN model add multiple hidden layers, each hidden layer adds more nonlinearity to model, which allows models can adapt more complex situation.

2.3.2 LSTM

LSTM (long short-term memory networks) [4] is a variety of recurrent neural networks (RNNs). The biggest improvement of LSTM is that it is more capable to learn long-term dependencies, especially in sequence prediction problems. The main purpose is to solve the problem of gradient disappearance and gradient explosion during long sequence training; hence, it can process the entire sequence of data [5]. There is the structure of SLTM:

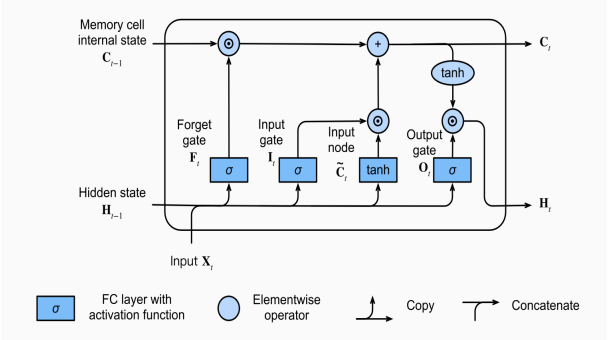

Figure 4. LSTM model

In SLTM, there are 3 kinds of gate: Input gate: Decide whether to ignore input data. Forget gate: Decrease the data towards 0. Output gate: Decide whether to use hidden state.

Input gate:

$$I_t = \sigma(X_t \cdot W_{xi} + H_{t-1} \cdot W_{hi} + b_i)$$

Forget gate:

$$F_t = \sigma(X_t \cdot W_{xf} + H_{t-1} \cdot W_{hf} + b_f)$$

Output gate:

$$O_t = \sigma(X_t \cdot W_{xo} + H_{t-1} \cdot W_{ho} + b_o)$$

Input node:

$$\widetilde{C_t} = tanh(X_t \cdot W_{xc} + H_{t-1} \cdot W_{hc} + b_c)$$

Its computation of input node is similar to the three gates, but using a tanh function with a value range for (-1,1) as the activation function. It can be regarded as an auxiliary memory unit that has not been normalized and has a relatively large value.

memory cell internal state:

$$C_t = F_t \odot C_{t-1} + I_t \odot \widetilde{C_t}$$

Hidden state:

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

Where:$\odot$ is Hadamard Product, an elementwise product operator

2.3.3 GRU

GRU (Gate Recurrent Unit) [2] is a type of RNN as well. Like LSTM, it was also proposed to solve problems such as long-term memory and gradients in backpropagation. The core idea of GRU is when observing a sequence, not every data is equally important, it's better when model pay more

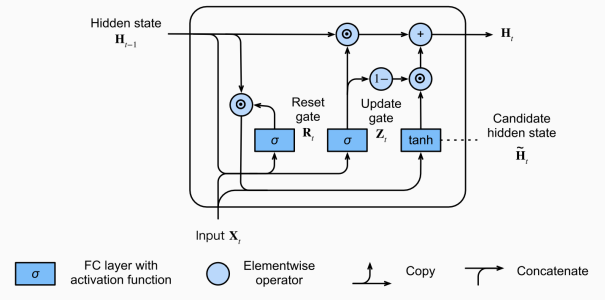

Figure 5. GRU model

attention to the more important information. The biggest improvement of LSTM is that its performance is similar to SLTM but it's computationally cheaper. This due to its' simplified structure.

In GRU, there are 2 kinds of gate: Reset gate: control reset Update gate: control update: decide whether memories more or forget more.

Reset gate:

$$R_t = \sigma(X_t \cdot W_{xr} + H_{t-1} \cdot W_{hr} + b_r)$$

Update gate:

$$Z_t = \sigma(X_t \cdot W_{xz} + H_{t-1} \cdot W_{hz} + b_z)$$

Candidate Hidden State:

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

Hidden state:

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \widetilde{H_t}$$

3. Experiments and analysis

The first round study uses SMA with 20 days sliding window as baseline, compared with RNNs models. There are 3 RNNs, they are the simple RNN, SLTM and GRU. Those models are trained and tested on S&P 500 ETF (SPY)dataset. In order to make a better comparison and after experimentation, the following hyperparameters will be used in all the RNNs models in the first round study:

- Sliding window: 20
- Number of layers: 1
- Hidden dimensions: 32
- Input feature: 4
- Learning rate: 0.01
- Optimizer: Adam

3.1. Model comparison

The baseline model, SMA with sliding window 20 days has the best performance, The MSE on overall data is 0.00054. Since the whole dataset plot is too crowded, the last 100 days prediction is showned as follow:

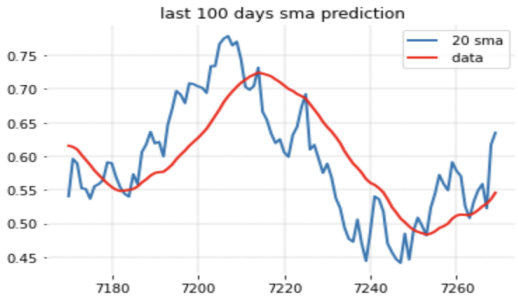

Figure 6. SMA prediction on last 100 days

from the plot, the prediction line can matching the real data good. It can be clearly shown in the figure that the SMA prediction line is very smooth, but the overall prediction lags behind than the real data.

As for RNN, GRU and LSTM models, they are performed good on training data, the loss droped very fast the first 10 epoch training, after 20 epochs training, the loss doesn't change too much(see figure 7.)

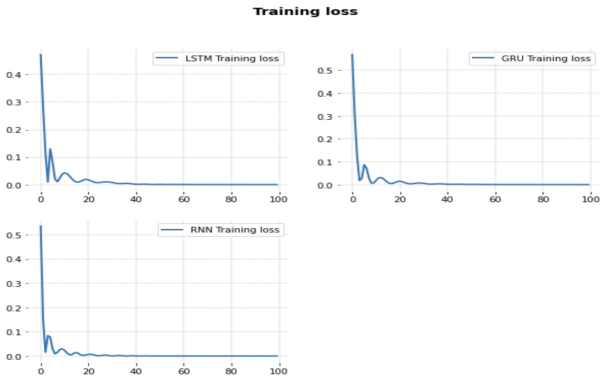

Figure 7. Training Loss on LSTM,GRU,RNN

After 100 epochs training, The LSTM model gives the MSE 0.00012 on train data. For GRU, after 70 epochs training, the MSE loss drops less than 0.0001. For RNN, after 90 epochs training, the MSE loss drops less than 0.0001. As for the training speed, RNN is the fastest, GRU is the second fastest and LSTM is the most slow model. In order in feel the difference clearly, last 100 days plot was zoomed as follow(see figure 8.):

As for the performance on test data, LSTM gives MSE loss as 0.10295, GRU gives MSE loss as 0.00192 and RNN gives MSE loss as 0.00185. All these 3 models are show-

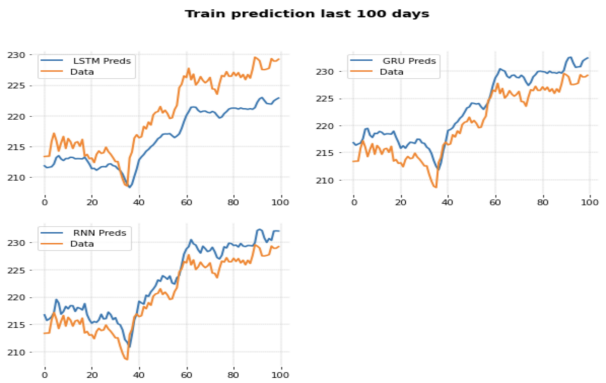

Figure 8. Training prediction on LSTM,GRU,RNN(last 100 training days)

ing overfitting compared with training loss. LSTM perform the worst, GRU and RNN perform similar and RNN is slightly better, but it may be due to randomness in the experiment(see figure 9).

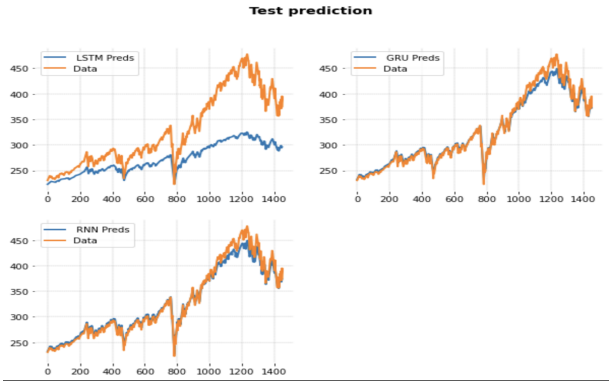

Figure 9. Test prediction on LSTM,GRU,RNN

From this study(see table 1.), RNN series models(LSTM, GRU, RNN)cannot beat the basic SMA model, GRU and RNN, wich the model strucure are simpler than LSTM, perform batter than LSTM. As for SMA, even the prediction performed nice, it has a strong lag.

3.2. Hyperparameter Tuning on GRU

Although the RNN series models perform poorly in stock price prediction, this experiment tested several sets of hyperparameters, aiming to find out the impact of different hyperparameter settings on the performance of the model. Both RNN and GRU performed well in the first round of experiments. In order to better compare the influence of hyperparameters on the model, the second round experiment only choose to train on GRU model.

3.2.1 Test on different sliding window

Different sliding windows were applied to test how it effect model's performance. In the case of keeping other variables the same, this experiment tested 4 groups of sliding windows: 10 days, 20 days, 50 days and 100 days. Since this experiment is designed in a many-to-one mode, using n days sliding window means using the first $n-1$ days to predict n_{th} day's stock price. The result was shown as follows(see figure10 and Table 2).

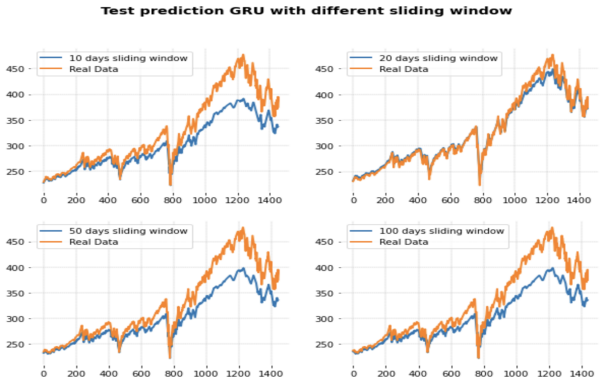

Figure 10. Test prediction on different sliding windows(GRU)

The relationship between the test MSE loss and the length of the sliding window is that as the length of the sliding window becomes longer, the test MSE loss first decreases and then rises. Too long or too short a sliding window are not conducive to the performance of the model. In this experiment, a time window length of 20 days performed best, and 10 days, 50 days and 100 days sliding windows have similar performance with MSE loss around 0.026.

3.2.2 Test on different hidden layers

Different hidden layers were applied to test how it effect model's performance. Increasing the number of hidden lays is increasing the complexity of model. In the case of keeping other variables the same(there are 32 neurals in each hidden layer), this experiment tested 4 different hidden layers: 1 layer, 2 layers, 3 layers and 5 layers. Since the experimental data is not large, this experiment designed not to use too many hidden layers. The result was shown as follows(see figure11).

The relationship between the test MSE loss and the numbers of hidden layers in this case is that as the the more the hidden layers, the more the test MSE loss(see table 3). With single hidden layer, the model perform the best. The possible reason is that the amount of data is too small, and the more the complexity of the model is increased, the more likely it is to overfitting, hence, in this case, single hidden layer performs the best.

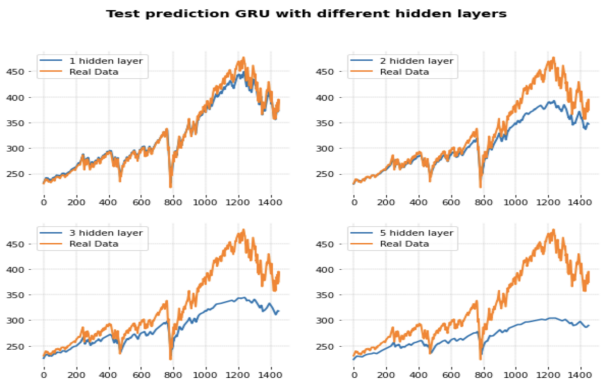

Figure 11. Test prediction on different hidden layers(GRU)

3.3. Applicability of the model

After the above experiment, GRU model with single hidden layer and using 20 sliding window perform the best. But there is a hypothesis: the performance of the model is closely related to the shape and trend of individual stock fluctuations. The model may only memorize the stock sequence and cannot predict the stock price trend very well. To test this hypothesis, the best GRU model from previous experiment was applied to test 4 different set of stocks which have different lengths, shapes and properties. The 4 datasets are(SPY is the original dataset):

- SPY: S&P 500 ETF (1993/12/31 - 2022/11/11)
- OXY: Western petroleum (2012/12/31 - 2022/11/11)
- KO: Cola Cola (1964/12/31 - 2022/11/11)
- TSLA: Tesla (1982/12/31 - 2022/11/11)

This is the training performance(see figure 12):

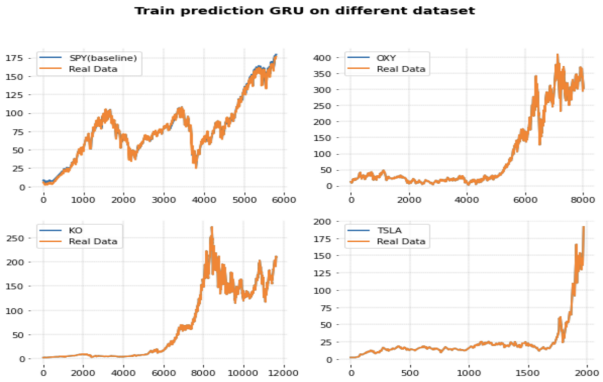

Figure 12. Training prediction on different dataset(GRU)

From the plot, the training performance are good on all these dataset, the prediction line is almost overlapped with the real data. Besides SPY, other stocks all have long tails at left and TSLA has the most sharp shape. There are the test result of these dataset(see figure 13 and table 4).

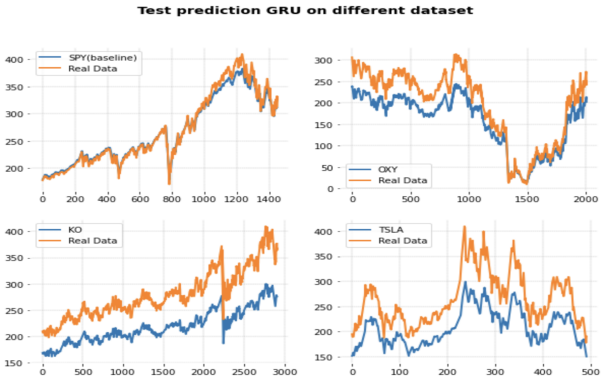

Figure 13. Test prediction on different dataset(GRU)

From the test result, the previous hypothesis is true that the model's applicability is not good. The model that can perform good on original data cannot perform good on other dataset. The shape of stock also matters, stocks with long smooth tails are trend to perform worse, for example, TSlA.

4. Conclusion

In this project, There are 3 rounds of implementations of stock prediction. The first round, 3 classic RNN models were implemented, LSTM, GRU and RNN, these 3 models were trained on S&P ETF dataset (7240 sample with 4 features). SMA with 20 sliding window were implemented as baseline model. SMA perform the best with test MSE 0.00054, GRU(0.00192) and RNN(0.00185) have similar performance, but LSTM didn't perform well, with test MSE as 0.10295. All the 3 RNN series models kind of shows overfitting(see table 1).

In the second round, 2 hyperparametr tuning were implemented on GRU to test how sliding window and the number of hidden layers affect model's performance. With the increase of sliding window, The MSE loss decrese first and after reaching some point start increasing. As for the number of hidden layers, the less the better. In this case GRU with single hidden layer and 20 days sliding window perform the best(see table 2).

In the last round, the best GRU model from second round were implemented on different dataset to test if the model really can predict well on stock price or just simply remember the original dataset. Also to test how different shape of stocks can impact model performance. The result shows that RNN series models are not perform very good on stock prediction because stock market is very complex, using only price to predict price is not enough. The model perform worse on the stock with long and smooth tail(see table 3).

There are some limitation of this study:

1st round study: Baseline model VS RNNs models

	SMA	LSTM	GRU	RNN
Test MSE	0.00054	0.10295	0.00192	0.00185

Table 1. MSE on testing data(models comparison)

2ed round study: Hyperparameter tuning on GRU

	10 days	20 days	50 days	100 days
Test MSE	0.02617	0.00192	0.02605	0.02617

Table 2. Test MSE with different sliding windows(GRU)

	1 layer	2 layers	3 layers	5 layers
Test MSE	0.00192	0.02108	0.06387	0.12418

Table 3. Test MSE with different hidden layers(GRU)

3rd round study: GRU with different dataset

	SPY	OXY	KO	TSLA
Test MSE	0.00192	0.04368	0.09303	0.08637

Table 4. Test MSE with different dataset(GRU)

There are just 2 hyperparameters being tested, there are many other hyperparameters should be tuned as well;

All the RNNs models kind of shows overfitting, maybe some noise can be added during data processing or model training to reduce overfitting;

Using only price to predict price is not good enough and the stock market is very complex, future study can add more features, like the news, public opinions and macroeconomic and political factors, etc [6].

This study is more focused on RNNs models, but there are many other new models, like transformer, Efficient Adaptive Ensamble method [7]. In feature's study these methods can be tested.

5. Code

Here is the git hub link: https://github.com/MelindaDong/RNNs_StockPrediction.

References

[1] https://d2l.ai 2

[2] Cho, Kyunghyun, et al. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. 2014. 3

[3] Gao, Y, Wang, R & Zhou, E 2021, 'Stock Prediction Based on Optimized LSTM and GRU Models', Scientific Programming, vol. 2021, pp. 1–8. 1

[4] Hochreiter, S & Schmidhuber, J 1997, 'Long Short-Term Memory', Neural Computation, vol. 9, no. 8, pp. 1735–1780. 3

[5] Sherstinsky, A 2020, 'Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network', Physica. D, vol. 404, p. 132306–. 3

[6] P. Khandelwal, J. Konar and B. Brahma, "Training RNN and it's Variants Using Sliding Window Technique," 2020 IEEE International Students' Conference on Electrical,Electronics and Com- puter Science (SCEECS), 2020, pp. 1-5, doi: 10.1109/SCEECS48394.2020.93. 7

[7] Malkiel BG, Fama EF. Efficient capital markets: a review of theory and empirical work. J Finance. 1970;25(2):383–417. 7

[8] https://www.investopedia.com/terms/m/movingaverage.asp. 2

[9] https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/ 2