Assignment 3 Stock Price Prediction with RNNs

Yuzhe Yan
December 6, 2024
University of Adelaide

Abstract

This report uses an RNN-based model to predict the stock prices of the CSI-300 index. The target is to check whether deep learning methods can predict time series data well. For this analysis, we use historical data from the China Financial Futures Exchange (CFFEX) covering nine years, from April 2010 to April 2019.

This report also compares three models: a standard RNN, an RNN with LSTM, and an RNN with GRU. The primary purpose is to find which model works best for this dataset. It is important to note that the results and conclusions here are only for academic and testing purposes. They do not represent making real stock market investment decisions.

Because these models have different essential characteristics, we expect the RNN with GRU to perform better than the others. The coming sections will provide more details on the methods, the comparison steps, and the analysis results.

1. Introduction

Many factors affect stock prices in the stock market, and those factors also affect different types of stocks. For a simple and initial purpose, this report focuses on data from the CSI-300 index over the past nine years, using data from CFFEX. Predicting stock prices can help improve returns, manage risks, and guide investors in making better choices. Using historical data supports this goal. Since index and stock trends depend strongly on time, I used Recurrent Neural Networks (RNNs) as a deep learning model in this report. [1]

RNNs are a good model that fits for time series analysis because they can analyze with time sequences that each point depends on earlier points. At any given time, current data is influenced by past data, and RNNs are designed to use this stored information to predict future outcomes. Also, compared to CNNs or other models, RNNs can efficiently find patterns in time-series data. However, basic RNNs often need to work with long-term dependencies. To address this problem, we can use advanced variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks. [4] [2] These reasons make RNNs an appropriate choice for this report and support the idea of exploring and comparing vanilla RNNs, LSTMs, and GRUs.

2. Background

2.1. Theories

The fundamental reason for choosing models like RNNs, LSTMs, and GRUs is their ability to capture temporal dependencies in sequence data by storing information through recurrent connections over time. By using these three models is highly significant for tasks involving time series analysis, natural language processing, and stock price prediction. [3]

This study begins by considering the basic architecture of Vanilla RNNs. A Vanilla RNN processes sequential data using a recursive approach. At each time step, the RNN takes the current input and the hidden state from the previous time step, then generates a new output and an updated hidden state. Weight matrices and activation functions govern this process. In this analysis, I will experiment with different activation functions so that the algorithm can incorporate past information when dealing with new inputs. To evaluate the performance of RNN models, we will use Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). Since the prediction target is the direction of stock movements, we will also compare Mean Directional Accuracy (MDA).

As explained before, the computation of a basic RNN at each time step involves taking the current input. And the previous hidden state to produce the output and new hidden state. However, Vanilla RNNs need to address the issues of vanishing and exploding gradients adequately. While these issues may not significantly impact simplified time series, they become more significant when dealing with long sequences during backpropagation.

For these problems, considering more models is essential. The next model to consider is the Long-Short-Term Memory (LSTM) network. LSTMs introduce gating mechanisms to overcome the shortcomings of Vanilla RNNs.

These gating mechanisms—specifically the Forget Gate, Input Gate, and Output Gate—control the flow of information, allowing the network to remember, update, or discard information as needed. LSTMs theoretically have a stronger capability to prevent gradient-related issues.

Compared to Vanilla RNNs, LSTMs use these gating mechanisms to decide which information to forget, which to add, and how to produce the final output at each time step. As a result, LSTMs typically excel at modeling longer sequences. This advantage is particularly evident in applications like stock price prediction. In fact, LSTM can be seen as a more powerful successor to the basic RNN model and a predecessor to the GRU model.

Gated Recurrent Units (GRUs) represent another type of RNN model. Compared to LSTMs, GRUs are somewhat simpler while still introducing gating mechanisms—namely the Update Gate and the Reset Gate. The Update Gate helps the model determine how much past information should be preserved, while the Reset Gate controls what information should be ignored during the current computation. GRUs eliminate the separate output gate and explicit memory cell found in LSTMs, often leading to more efficient training.

In summary, while Vanilla RNNs, LSTMs, and GRUs all aim to capture temporal dependencies in sequences, they do so at different levels of complexity and efficiency. LSTMs and GRUs address the limitations of Vanilla RNNs, making them strong choices for longer sequences and more challenging prediction tasks such as forecasting stock prices.

2.2. DataSet

The data used in this report comes from Professor Zhen Z.'s GitHub repository (https://github.com/ChZPan/stock-market-prediction-RNN). In his repository, there are four datasets related to the CSI 300 index, from April 2010 to April 2019, with each dataset covering a two-year interval. To work with a longer overall time period, I decided to combine these four datasets into a single dataset. The data includes the date, time, opening price, and closing price. In this report, the analysis and prediction focus on the closing price column.

2.3. Model Implementation

I will make some mathematical additions to RNN, Vanilla RNN, LSTM, and GRU. First of all, for RNN, apart from the above description of RNN, the important point of RNN is that each output from the hidden state at time step t-1 serves as input with the time step n-1. As shown in the figure below, this is a classic RNN structure. As Figure 1 shows,

The computation within an RNN is defined by two main equations:

$$h_t = g(W_{xh}x_t + W_{hh}h_{t-1} + b_h),$$

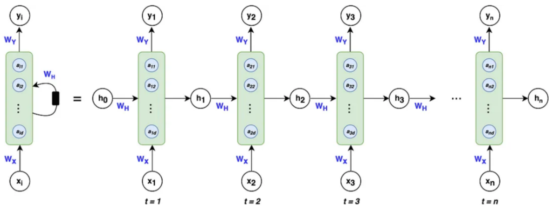


Figure 1. Basic RNN Structure

where:

- h_t is the hidden state at time t,
- g is the activation function (e.g., $\tanh$),
- W_{xh} is the weight matrix for the input to hidden state,
- W_{hh} is the weight matrix for the hidden state recurrence,
- x_t is the input vector at time t,
- h_{t-1} is the hidden state from the previous time step,
- b_h is the bias term.

The output at each time step is computed as:

$$z_t = g_n(W_{hz}h_t + b_z),$$

where:

- z_t is the output vector at time t,
- W_{hz} is the weight matrix connecting the hidden state to the output layer,
- b_z is the output bias term.

The second is vanilla RNN. Compared with the most basic RNN, the main change of vanilla RNN is that its final hidden pass through the fully connected layer and finally gets the output. For vanilla RNN, tanh is used as the activation function. In fact, vanilla RNN and RNN are very similar in many cases.

The third model implementation is LSTM. As Figure 2 shows the structure about LSTM. The specialty about it is LSTM contain the memory block. And the cell state in LSTM allows LSTM to selectively retian or delete information based on the gates' decision. The last one is GRU. As shown in the Figure 3, this is the structure diagram of GRU. Due to the existence of a reset gate and update gate, GRU is more stable than LSTM. Especially for sequential data, GRU should be more efficient in processing. [5]

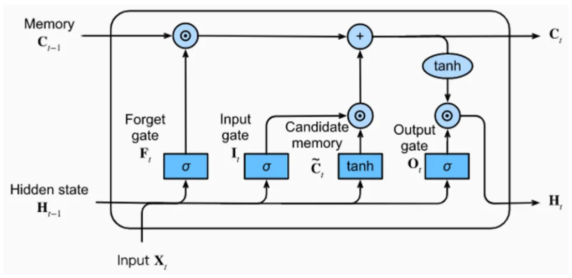


Figure 2. LSTM Structure

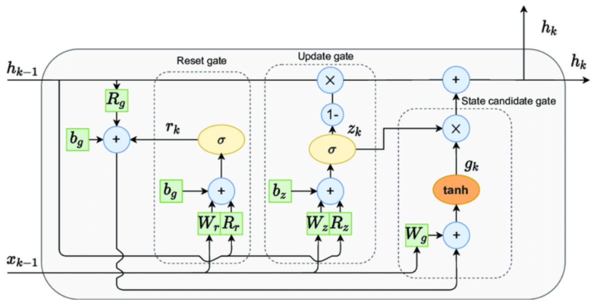


Figure 3. GRU Structure

3. Methodology

3.1. Model Configuration

Distinguishing the characteristics of Vanilla RNNs, LSTMs, and GRUs and selecting the most suitable model is crucial, given that each structure offers its own unique advantages and features. The use of Keras has significantly reduced the workload in this project, allowing for more epochs to be run. Initially, we will train each of the three models for 100 epochs. After identifying the best-performing model, we will then proceed with a series of hyperparameter tuning steps.

In addition to basic parameter adjustments, two primary approaches will be highlighted here. First, we will experiment with different optimizers, likely Adam and SGD, as these two are commonly used in time series analysis. Second, we will vary the activation functions. Specifically, we will consider ReLU and Tanh as two different activation functions and attempt to identify which one yields the best results.

The Rectified Linear Unit (ReLU) is a widely used activation function that remains linear for positive inputs, which often speeds up training and convergence. However, ReLU can also suffer from issues such as the “dying ReLU” problem, potentially affecting model performance. On the other hand, the Tanh (hyperbolic tangent) activation function maps inputs to the range [-1, 1]. Compared to ReLU, Tanh provides a smoother curve and can capture more complex nonlinear relationships, although it may still be influenced by the vanishing gradient problem.

By comparing these hyperparameters—optimizers and activation functions—we aim to determine which combination performs best for stock price prediction. Ultimately, our goal is to identify the model, along with its associated configuration, that can deliver the most accurate and reliable forecasts.

3.2. Data Process

The CSI-300 Index price dataset contains 32208 rows. For testing and analysis, the target is the ClsoePrice column, which represents the market close price at the time period. The preprocess step is necessary for effectively using the dataset. It includes scaling the values to the range of -1 to 1. After running the data splitting process, the CSI-300 dataset is transferred to sequential order. The normalization process is also important after scaling.

3.3. Hyperparameter Set

Distinguishing the characteristics of Vanilla RNNs, LSTMs, and GRUs and selecting the most suitable model is crucial, given that each structure offers its own unique advantages and features. The use of Keras has significantly reduced the workload in this project, allowing for more epochs to be run. Initially, we will train each of the three models for 100 epochs. After identifying the best-performing model, we will then proceed with a series of hyperparameter tuning steps.

In addition to basic parameter adjustments, two primary approaches will be highlighted here. First, we will experiment with different optimizers, likely Adam and SGD, as these two are commonly used in time series analysis. Second, we will vary the activation functions. Specifically, we will consider ReLU and Tanh as two different activation functions and attempt to identify which one yields the best results.

The Rectified Linear Unit (ReLU) is a widely used activation function that remains linear for positive inputs, which often speeds up training and convergence. However, ReLU can also suffer from issues such as the “dying ReLU” problem, potentially affecting model performance. On the other hand, the Tanh (hyperbolic tangent) activation function maps inputs to the range [-1, 1]. Compared to ReLU, Tanh provides a smoother curve and can capture more complex nonlinear relationships, although it may still be influenced by the vanishing gradient problem.

By comparing these hyperparameters—optimizers and activation functions—we aim to determine which combination performs best for stock price prediction. Ultimately, our goal is to identify the model, along with its associated configuration, that can deliver the most accurate and reliable forecasts.

4. Experimental analysis and results

In this report, training and testing are conducted on CIFAR-10. By comparing the base model used for testing with three additional CNN models examined in-depth, several data-driven conclusions are drawn, including training loss, validation loss, and training accuracy. By comparing the outputs of these metrics, a relative assessment of model performance can be made. Furthermore, an in-depth analysis of ResNet-18 is performed. Through tuning several key hyperparameters, attempts are made to identify an optimal model state and determine the best combination of hyperparameters to enhance the performance of ResNet-18.

4.1. Model Comparision

The transparent model architecture used in this project is based on TensorFlow. Such transparent architectures are RNN, LSTM, and GRU. This section will demonstrate how to compare the advantages of such transparent architectures and discover the best-performing models. After setting up the basic code through Keras, I will run 50 epochs for each RNN model. By comparing the training model and the verification model, we can find the group with better performance and then find the more suitable model.

The first is vanilla RNN. After 50 epoch measurements, the Test MSE of vanilla RNN is 43.6937, the Test RMSE is 6.6101, and the Test MDA is 0.5162. MDA represents the accuracy of direction prediction. Since the predicted stock index is CSI-300, this index has certain particularities, and it is not easy to obtain a high MDA value based on time series prediction. However, it is precisely because of this that it is very important to help us find a more suitable model by evaluating it. As shown in Figure 4 and Figure 5, the blue line represents the training loss, and the orange line represents the validation loss. Since 50 epochs of training have been conducted, the training loss continuously decreases, indicating that the model's effect in fitting the training data is becoming smaller and smaller. The same verification loss is high in the initial performance. Then, it drops rapidly and tends to be stable, which proves that Vanilla RNN also performs well on the verification set. Similarly, since the verification loss has not increased, the model is relatively stable, and there is no overfitting. The second one is LSTM. The conclusion drawn by LSTM this time is worse than expected. Based on the same run of 50 epochs, the Test MSE of LSTM can be obtained as 43.7303, the Test RMSE is 6.6129, and the most important Test MDA is 0.5058. It can be clearly seen that each evaluation indicator is slightly worse than Vanilla RNN. However, since the changes in the set basic parameters are normal, even if we see a reasonable loss curve from the Figure 6, it still shows that LSTM is unsuitable for this data set. Finally, there is GRU. The MDA of GRU here is 0.5197, which is better than the previous RNN model. Similarly, getting Test MSE and Test RMSE shows

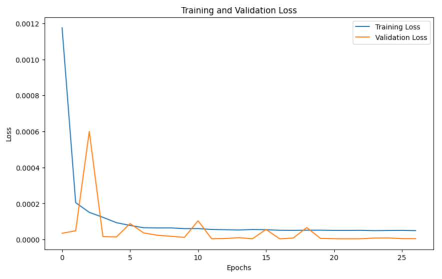


Figure 4. Vanilla RNN Performance

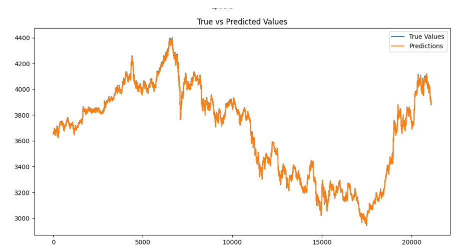


Figure 5. Vanilla RNN Performance

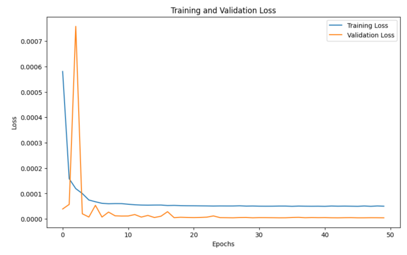


Figure 6. LSTM Performance

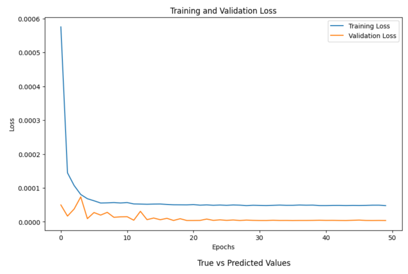


Figure 7. GRU Performance

that both Test MSE: 36.4472 and Test RMSE: 6.0372 are better than the previous two sets of model architectures. As shown in the Figure 7, GRU has a better convergence speed during training and drops to a stable value in a few training cycles. Similarly, if you observe the orange line and verify the loss, you can also find that both the validation and training losses are relatively low. Moreover, it is relatively stable at a gradually decreasing level.

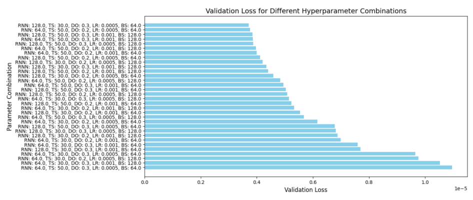


Figure 8. Optimizer-Adam

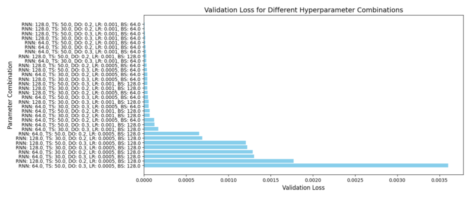


Figure 9. Optimizer-SGD

4.2. Hyper-Paramter Tuning:Optimizer

Based on the above tests, the GRU is the most suitable model for the current situation. This is likely because GRU suppresses some noise by updating the gating mechanism, so GRU has the best performance. I used hyperparameter optimization to get, explore and combine the GRU with the data set. In this part, two optimizers will be used; first, the most widely used Adam and second, SGD will be used for comparison. As shown in Figure 8 and Figure 9, after optimizing through different hyperparameters, the best model, based on Adam, has a dropout rate of 0.3, a learning rate of 0.0005, and a batch size of 64. However, for SGD, the best model parameters are a dropout rate of 0.3, a learning rate of 0.0005, and a batch size of 128. From comparing their best models, we can see that Adam is stronger than SGD. The validation loss of 'tanh' is closer to 0 than SGD's, which is about two decimal points less. Under the same circumstances, the verification loss values of Adam are relatively close, which also shows that Adam is less sensitive to different parameters. So Adam is a more suitable optimizer here.

4.3. Hyper-Parameter Tuning:Activation Function

The purpose of this part is to test different activation functions. Remove the previous tanh as the essential activation function. As shown in the figure 10, I switched to ReLu here and performed 50 rounds of epochs on the GRU using ReLu. Through comparison, it was found that the performance of ReLU is more potent than tanh's. The reason why ReLu is obviously due to tanh may be that ReLu has a stronger ability to deal with nonlinear nature. Although stocks are composed of time series, ReLu has performed better in training and testing the CSI-300 index because the index is not continuous.

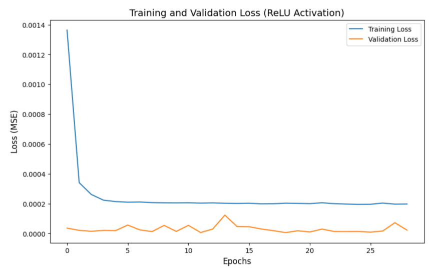
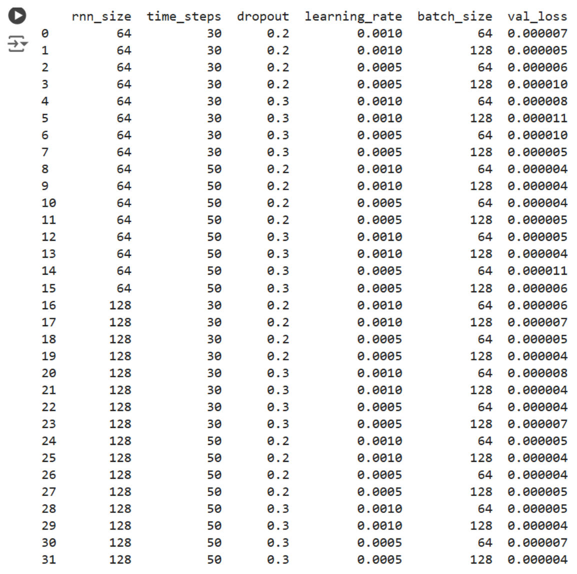


Figure 10. ReLu Activation



```
    rnn_size  time_steps  dropout  learning_rate  batch_size  val_loss
0         64          30      0.2         0.0010          64  0.000007
1         64          30      0.2         0.0010         128  0.000005
2         64          30      0.2         0.0005          64  0.000006
3         64          30      0.2         0.0005         128  0.000010
4         64          30      0.3         0.0010          64  0.000008
5         64          30      0.3         0.0010         128  0.000011
6         64          30      0.3         0.0005          64  0.000010
7         64          30      0.3         0.0005         128  0.000005
8         64          50      0.2         0.0010          64  0.000004
9         64          50      0.2         0.0010         128  0.000004
10        64          50      0.2         0.0005          64  0.000004
11        64          50      0.2         0.0005         128  0.000005
12        64          50      0.3         0.0010          64  0.000005
13        64          50      0.3         0.0010         128  0.000004
14        64          50      0.3         0.0005          64  0.000011
15        64          50      0.3         0.0005         128  0.000006
16       128          30      0.2         0.0010          64  0.000006
17       128          30      0.2         0.0010         128  0.000007
18       128          30      0.2         0.0005          64  0.000005
19       128          30      0.2         0.0005         128  0.000004
20       128          30      0.3         0.0010          64  0.000008
21       128          30      0.3         0.0010         128  0.000004
22       128          30      0.3         0.0005          64  0.000004
23       128          30      0.3         0.0005         128  0.000007
24       128          50      0.2         0.0010          64  0.000005
25       128          50      0.2         0.0010         128  0.000004
26       128          50      0.2         0.0005          64  0.000004
27       128          50      0.2         0.0005         128  0.000005
28       128          50      0.3         0.0010          64  0.000005
29       128          50      0.3         0.0010         128  0.000004
30       128          50      0.3         0.0005          64  0.000007
31       128          50      0.3         0.0005         128  0.000004
```

Figure 11. Hyperparameter tuning for GRU

5. Reflection and Limitation

I used three RNN models to train and analyze CSI-300 in this project. In the meantime, it was found that LSTM performs worst in the context of powerful mechanisms. This warned me that the optimized model could have been more powerful. Similarly, even GRU optimized through hyperparameters has no obvious advantage over Vanilla RNN. Of course, this is related to the relatively simple data structure used. It is actually irresponsible to make simple model predictions in financial markets. There are so many influencing factors that come into play. Time series is only one of the inevitable influencing conditions. There is no reference value if only the past index price is considered. However, through this experiment, I learned a lot about RNN and studied the advantages and disadvantages of different models.

6. Code

The codes for this project is available on Github Repository: DeepLearning-RNN(https://github.com/HowardYan-

PG/DeepLearning-RNN). The repositiry is publicly readable. Appreciate your view.

7. Conclusion

This report explores the application of Recurrent Neural Networks (RNN) and their variants (LSTM and GRU) in stock price prediction based on historical data from the CSI-300 Index. By modeling data from 2010 to 2019, the study aimed to evaluate the ability of deep learning methods to process time series data. To compare model performance, the report uses mean square error (MSE), root mean square error (RMSE), and direction prediction accuracy (MDA) as evaluation indicators.

The results show that GRU performs best among the three models, achieving the lowest test MSE and RMSE values and the highest MDA. Its superiority may be due to its simple and efficient gating mechanism. In addition, the performance of GRU is further improved through hyperparameter adjustments, such as optimizers (Adam and SGD) and activation functions (ReLU and Tanh). Among them, the Adam optimizer and ReLU activation function combination has the best effect, showing faster convergence speed and lower verification loss.

Although the experiment achieved good results, the report also pointed out the model's limitationsmodel's limitations, such as a single data source and the complex and changing influencing factors in financial markets that needed to be fully taken into account. Through this study, I deeply understand the advantages and disadvantages of RNN and its variants in time series analysis, providing a reference for future research and practice.

References

[1] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5457–5466, 2018. 1

[2] Balduíno César Mateus, Mateus Mendes, José Torres Farinha, Rui Assis, and António Marques Cardoso. Comparing lstm and gru models to predict the condition of a pulp paper press. *Energies*, 14(21):6958, 2021. 1

[3] Mahla Nikou, Gholamreza Mansourfar, and Jamshid Bagherzadeh. Stock price prediction using deep learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management*, 26(4):164–174, 2019. 1

[4] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020. 1

[5] Moritz Wolter and Angela Yao. Complex gated recurrent neural networks. *Advances in neural information processing systems*, 31, 2018. 2