# Explore the impact of hyper-parameters on the performance of RNN and LSTM models in predicting stock trading volume

Hengyi Ma
a1875198
The University Of Adelaide

## Abstract

*Stock prediction, especially direct stock price prediction, has complex influencing factors and intuitive value returns, and has always been one of the hot topics in modern society. In the field of artificial intelligence, researchers have also tried to apply various existing models to stock predictions to assist decision-making. This report will try to introduce the classic neural network RNN and its improved model LSTM, which deal with time series related problems, into stock prediction for regression prediction. Different from the common stock price prediction, this report will try to predict the more challenging goal of daily trading volume, and study the performance of RNN and LSTM on trading volume prediction under different hyper-parameters by adjusting the hyper-parameters.*

*The code of this report can be accessed at:* `https://github.com/HengyiMa/deeplearning-demo/tree/main/lstm%2Crnn%20%E8%B6%85%E5%8F%82%E6%95%B0%E6%8E%A2%E7%A9%B6`

*Keywords: stock prediction, trading volume, RNN, LSTM*

## 1. Introduction

The stock market is an integral part of the global economy, and successful stock market investments will bring substantial returns, which makes stock market predictions a hot topic. However, the fluctuations of the stock market are affected by a large number of factors such as policy orientation and market sentiment. Its complexity and uncertainty make predicting stock prices and trading volumes a subject of great concern.

Traditional stock prediction methods usually use linear regression, SVM, EMA [8] and other means. However, these methods either cause information loss during the prediction process, or are not perfect when dealing with nonlinear and dynamic relationships. The rise of deep learning models provides new possibilities for solving this problem. Recurrent neural networks such as RNN and LSTM have excellent performance in processing sequence data and are therefore widely used in stock market prediction tasks.

Although related research has flourished in recent years, stock prediction is still a challenging task with high uncertainty. There are even theories such as random walk theory [5] that state that changes in asset prices are random. This means that stock price movements are unpredictable, so past prices cannot be used to accurately predict future prices. In fact, today's stock price forecasts are mostly based on short-term forecasts and do not perform well on long-term forecasts.

Combining hierarchical learning theory [2] and multimodal learning theory, people can make more accurate predictions of targets by combining multiple levels of prior information. Therefore, an improved method for stock price prediction that can be considered is By first predicting the relevant quantities of stock predictions, and then combining these prediction results to predict stock prices. This article will focus on one of the characteristics - the prediction of daily trading volume. It predicts daily trading volume by applying RNN and LSTM, two neural networks with time series characteristics, and modifies some hyper-parameters to observe different hyper-parameters. Let's look at the performance of the model to make a preliminary exploration of RNN and LSTM prediction of daily trading volume.

## 2. Background knowledge

RNN: Traditional deep neural networks perform well in processing problems such as multi-classification of images. However, in addition to data like images, there is also a large amount of data with temporal characteristics, which we generally call time series, such as speech and price fluctuations. This type of data has a temporal or logical order. Therefore, a neural network variant RNN that can remember this order was developed.

A recurrent neural network is a neural network that is specialized for processing a sequence of data x(t)= x(1), . . . , x(t) with the time step index t ranging from 1 to t[3]. For

tasks such as speech and language that involve continuous input, it is often better to use an RNN. In an NLP problem, if you want to predict the next word in a sentence, you have to know the words that precede it. RNNs are called recurrent networks because they perform the same task on every element in the sequence, with the output depending on previous computations. Another way to think of RNNs is that they have a "memory" that captures the information that has been computed so far. The operation method of RNN is shown in the figure below:

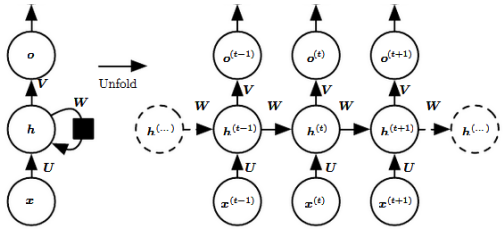


Figure 1: A typical structure of RNN model

LSTM: LSTM[6] is a long short-term memory network and a special RNN. Compared with traditional RNN, LSTM is more suitable for processing and predicting important events with long intervals in time series.

The traditional RNN structure can be seen as a "circuit" composed of multiple repeated neurons. Each neuron receives input information and produces an output, and then uses the output again as the input of the next neuron and passes it on in sequence. This structure can learn short-term dependencies on sequence data, but due to gradient disappearance and gradient explosion problems, RNN is difficult to achieve good performance when processing long sequences.

LSTM can effectively solve long sequence problems by introducing the concepts of memory cells, input gates, output gates and forgetting gates. The memory cell is responsible for storing important information. The input gate decides whether to write the current input information into the memory cell. The forgetting gate decides whether to forget the information in the memory cell. The output gate decides whether to use the memory cell information as the current output. Control of these gates can effectively capture important long-term dependencies in sequences and can resolve gradient problems. The internal structure of a typical LSTM network is as shown in the figure 2:

## 3. Related work

MSE loss function: In this experiment, regression prediction was performed, and MSE loss was used as the loss function. MSE is a measure of estimator quality [7]. MSE measures the average squared difference between model predictions and actual values. The calculation formula is

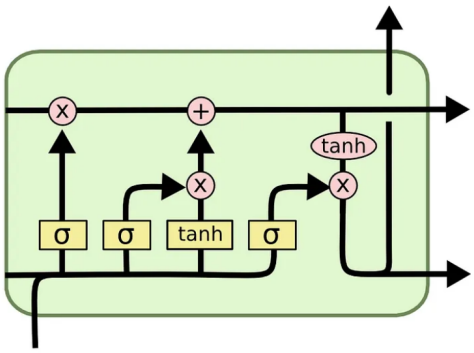


Figure 2: A typical structure of LSTM model

as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Among them, n is the number of samples, $y_i$ is the actual value, and $\hat{y}_i$ is the predicted value. Since it is derived from the square of the Euclidean distance, it is always a positive value and decreases as the error approaches zero. For MSE, the smaller the value, the closer the model's prediction is to the actual value, so the smaller the MSE, the better.

Adam optimizer: The experiment in this report uses the Adam optimizer [4]. Adam (Adaptive Moment Estimation) is an adaptive learning rate optimization algorithm that is widely used for model training in deep learning. Compared with the traditional stochastic gradient descent (SGD) algorithm, Adam is more efficient in practical applications, especially when processing large-scale data and complex models. Adam is described as combining the advantages of two other extensions of stochastic gradient descent. Specifically:

The Adaptive Gradient Algorithm (AdaGrad) maintains the learning rate of each parameter, thereby improving performance on sparse gradient problems such as natural language and computer vision problems.

Root mean square propagation (RMSProp) also maintains learning rates for each parameter, which are adjusted based on the average of the recent magnitudes of the weight gradient (i.e., how fast it changes). This means that the algorithm performs well on online and non-stationary problems such as noise.

Gradient vanishing/exploding: Gradient vanishing and gradient explosion problems are common problems in deep learning. Since neural networks require backpropagation to update weights, and the process of backpropagation follows the chain rule, the deeper the neural network is, the smaller or larger the gradient value will be. Vanishing gradient means that during the backpropagation process, the gradient of the network gradually decreases and eventually becomes very close to zero. This will cause the underlying

weights to be almost never updated, making it difficult for the underlying network to learn useful features. Gradient explosion means that during the backpropagation process, the gradient becomes very large, causing the weight updates to be too large and the model parameters to diverge. This can lead to numerical instability and training failure [1].

## 4. Proposed method

This report uses stock market data collected from Microsoft's qlib for experiments, and selects the SH60000 data for training and testing.

We hope to predict the trading volume of the next day using features such as trading volume and the closing price of the day. But unlike stock market price prediction, trading volume usually has more severe volatility and frequent changes, which also makes trading volume forecasting a more challenging task.

The comparison of stock market prices and trading volumes in the training set used is as follows: (data is normalized)

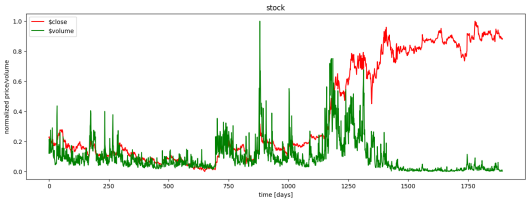


Figure 3: Regularized trading volume and stock close price trend charts

There are two hyper-parameters studied in this article:

hidden_size:It is the number of feature dimensions in the hidden layer, which determines the complexity of the hidden layer data during model training. It can be approximately understood as the number of neurons in each layer of CNN.

num_layers:It determines how many layers of network stacks there are in the same network module, which can be approximately understood as the network depth in CNN.

We choose the network structure of RNN and LSTM under hidden_size=5, num_layers=1, as the baseline model, and set the sliding window size to 20.

The purpose of the experiment is to compare horizontally and vertically the performance of RNN and LSTM models in transaction volume prediction regression under different hyper-parameters. Therefore, for the neural networks of RNN and LSTM, we adjust the parameters from the two perspectives of hidden_size and num_layers respectively, and observe the performance of the regression model in predicting transaction volume as the hyper-parameters change from the two perspectives of the model itself and between the models.

Regarding hidden_size, in addition to the hidden_size=5 of the baseline, a total of three different parameters of hidden_size =25, 125, and 250 are set for the two models.

On num_layers, in addition to baseline's num_layers=1, a total of three different parameters are set for the two models: num_layers=5, 15, and 25.

Based on this, we designed two groups of large experiments based on the number of models. Each group of large experiments was grouped according to two variables. Each group tested four models including baseline, resulting in a total of 16 groups of experiments.

This report is to establish a regression model, so it no longer uses accuracy as an evaluation criterion, which is commonly used in classification problems. Instead, it uses visualization methods as indicators, that is, whether the trend is consistent with the real results and the gap between the trends and the real results is used as the evaluation criterion. At the same time, we use the loss curve as an auxiliary reference to determine the quality of training and whether it is overfitting/underfitting. At the same time, in order to ensure fair results, the optimizers in all experiments used the Adam optimizer and set the initial learning rate to 0.005, and all ran for 100 epochs.

## 5. Experiment results

We first analyze the experiments of the two models separately, and then compare the performance between the two models. For the RNN model to predict transaction volume, the prediction results after changing the hidden_size parameter are as follows: The output of hidden_size=5 model:

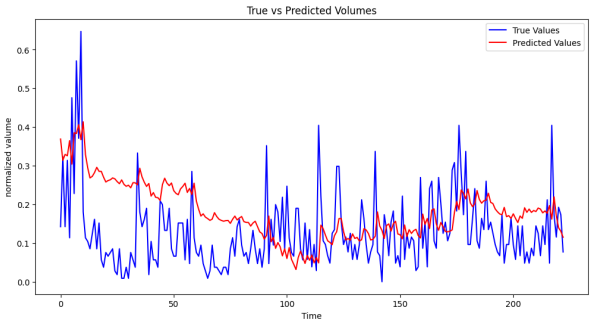


Figure 4: Result for hidden_size=5 RNN model

The output of hidden_size=25 model:

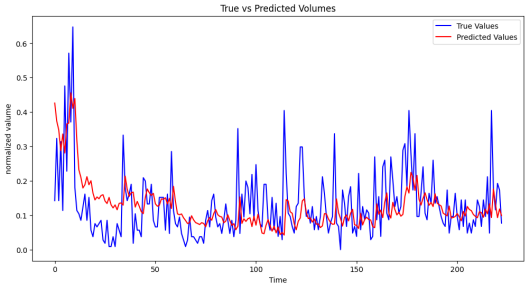


Figure 5: Result for hidden_size=25 RNN model

The output of hidden_size=125 model:

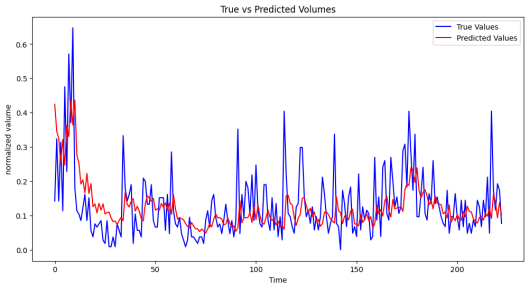


Figure 6: Result for hidden_size=125 RNN model

The output of hidden_size=250 model:

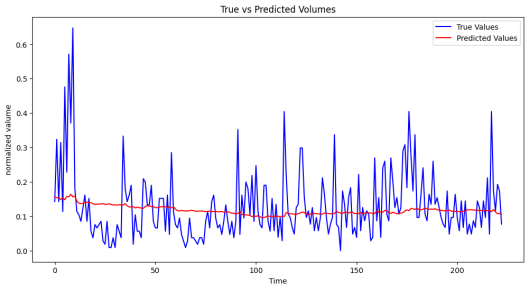


Figure 7: Result for hidden_size=250 RNN model

Analyzing the process of gradual increase of hidden_size, it can be found that compared with the real results, as the hidden_size increases to 5, 25, and 125, the predicted results are more accurate than the actual results. This accuracy is reflected in two aspects: more accurate prediction trends and smaller differences between predicted values and actual values. It shows that when the RNN model is making predictions, appropriately improving the feature dimensions of the hidden layer of the model can help improve the learning ability of the model. But when hidden_size=250, the prediction curve quickly attenuates to close to the horizontal line and gives relatively poor prediction results. However, observing the peaks and troughs of the prediction curve, we can see that the timing of the peaks and troughs is still consistent with the actual model, indicating that hidden_size is higher , the model's learning ability is greatly reduced, but the model still has a good perception of the peaks and troughs in the timing diagram.

The prediction results after changing the size of the num_layers parameter are as follows:

The output of num_layers=1 model:

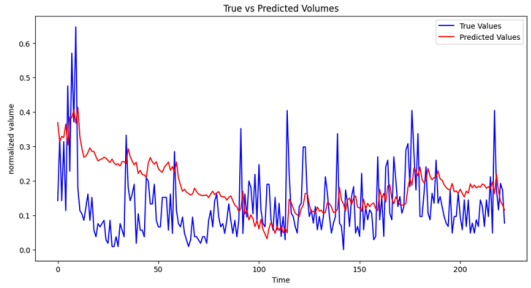


Figure 8: Result for num_layers=1 RNN model

The output of num_layers=5 model:

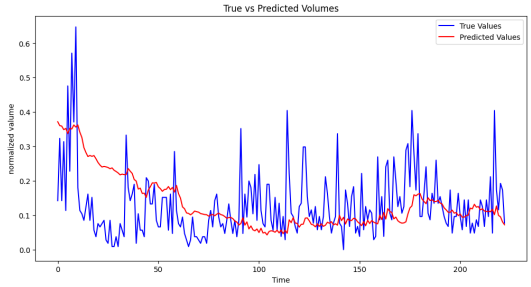


Figure 9: Result for num_layers=5 RNN model

The output of num_layers=15 model:

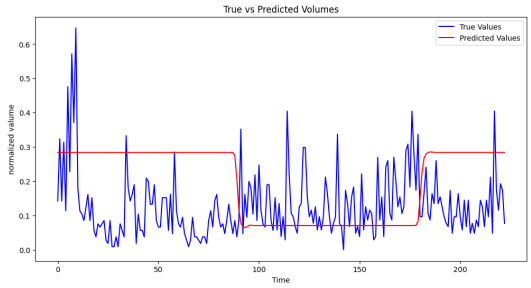


Figure 10: Result for num_layers=15 RNN model

The output of num_layers=25 model:

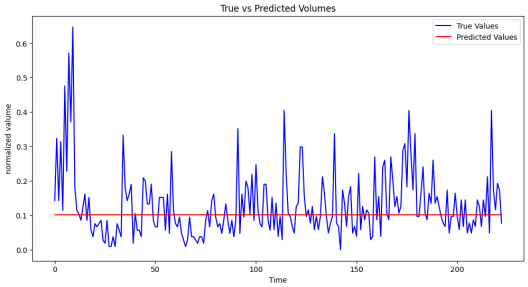


Figure 11: Result for num_layers=25 RNN model

It can be observed that as the network depth of RNN stack increases, the expressive ability of the model gradually degrades. Degradation is manifested as a gradual decline in the nonlinear ability of the prediction regression curve, from the curve degenerating to a segmented pattern similar to a three-segment horizontal line, to completely degenerating into a horizontal line.

It shows that under the current complexity of data training, using a deeper RNN model results in a greatly weakened expression ability of the model.

For the LSTM model to predict transaction volume, the prediction results after changing the hidden_size parameter are as follows:

The output of hidden_size=5 model:

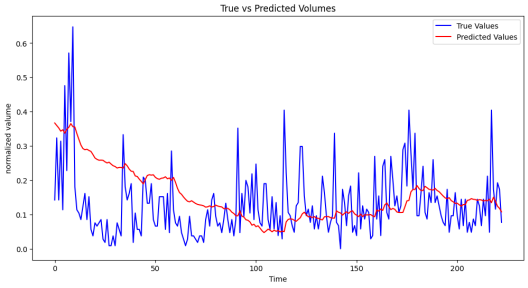


Figure 12: Result for hidden_size=5 LSTM model

The output of hidden_size=25 model:

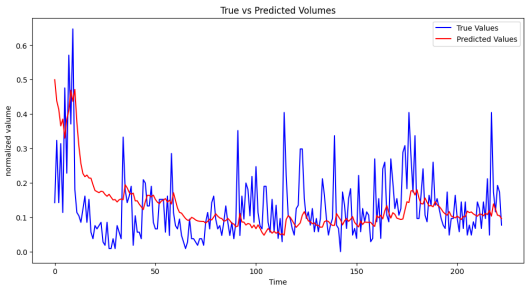


Figure 13: Result for hidden_size=25 LSTM model

The output of hidden_size=125 model:

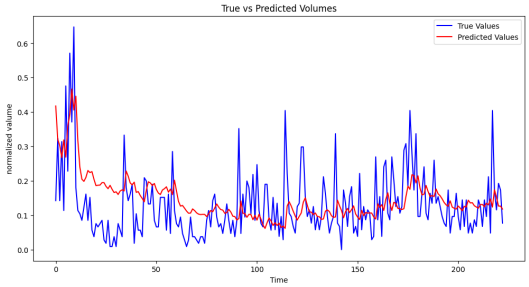


Figure 14: Result for hidden_size=125 LSTM model

The output of hidden_size=250 model:

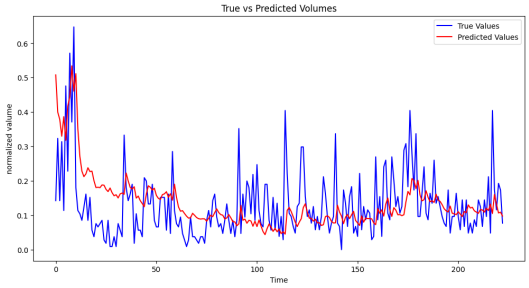


Figure 15: Result for hidden_size=250 LSTM model

It can be observed that as hidden_size increases from 5 to 25 and 125, the expressive ability of the predicted regression curve gradually increases, and when hidden_size=250, the performance of the regression curve is relatively stable. During the whole process, the expressive ability of the LSTM model first increases as hidden_size increases, and then stabilizes. This shows that a certain increase in hidden_size will help improve the learning ability of LSTM, and this expressive ability will not decrease rapidly as hidden_size continues to increase.

Next, the prediction results after changing the size of the num_layers parameter are as follows:

The output of num_layers=1 model:

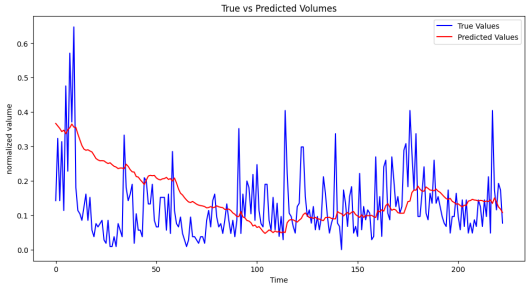


Figure 16: Result for num_layers=1 LSTM model

The output of num_layers=5 model:

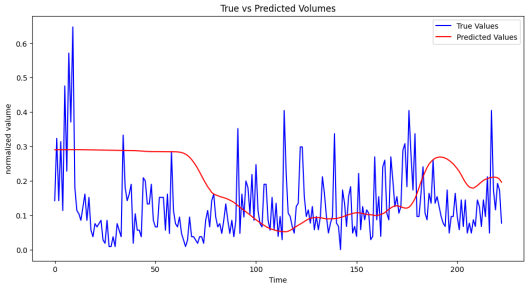


Figure 17: Result for num_layers=5 LSTM model

The output of num_layers=15 model:

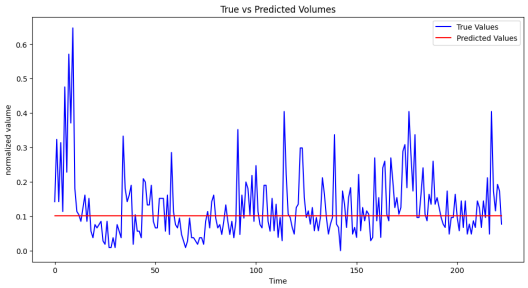


Figure 18: Result for num_layers=15 LSTM model

The output of num_layers=25 model:

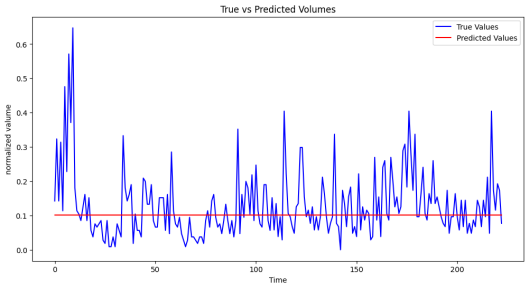


Figure 19: Result for num_layers=25 LSTM model

It can be observed that as the network depth num_layers of each layer stack increases, the expressive ability of the model decreases rapidly, so that by the time num_layers=15, it has degenerated to a situation close to a horizontal line. This shows that when the current task uses LSTM and related parameters as depth, the increase in depth will quickly reduce the expressive ability of the model.

## 6. Result analysis

We select LSTM and RNN that perform better in the entire hyper-parameter analysis for comparative analysis of model observations and real values. Taking the baseline model of RNN as our reference model, the comparison of the model's performance in trading volume prediction and the performance of a simple stock price prediction model is as follows:

The model used in this report for trading volume forecasting:

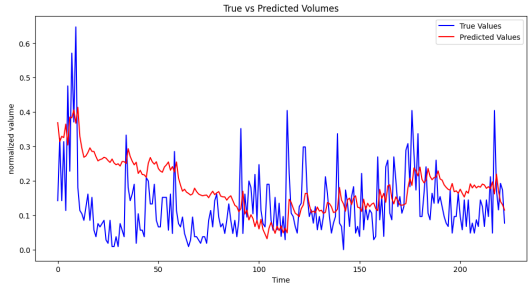


Figure 20: Our RNN baseline model for volume prediction

A simple stock price prediction model:

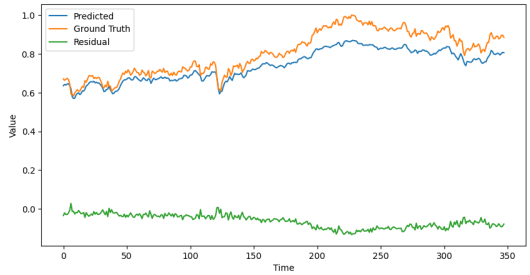


Figure 21: A simple stock price prediction model

It can be seen that the stock price prediction model performs far better than the trading volume prediction. There are many possible reasons for this situation. On the surface, the variance of trading volume is much larger than the variance of stock price, which means that the volatility of trading volume is far greater than that of stock price, and it is even more irregular. In addition, the trading volume data shows a large number of peaks and troughs with large gaps, while the continuity of stock price fluctuations is much stronger than the trading volume data. From the perspective of the internal reasons of the model itself, based on the characteristics of the neural network fitting image under under-fitting, over-fitting, and normal fitting, It is speculated that the neural network itself is not as good at fitting this kind of data with larger variance, higher frequency, and stronger volatility than more stable data. The fitting ability of the data is not as good as that of more stable data. A typical image of under-fitting, normal fitting, and over-fitting expressing the model fitting situation is as follows:

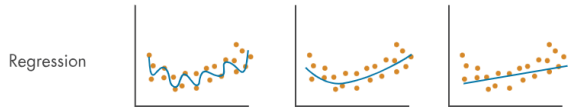


Figure 22: over-fitting,normal fitting and under-fitting in a regression model

From the perspective of hyper-parameter adjustment, adjusting hidden_size and num_layers respectively for the RNN model and LSTM model also show different results. In the process of increasing hidden_size to the RNN model, the performance of the fitted regression curve first becomes better and then becomes worse. However, in the process of increasing hidden_size to the LSTM model, the performance of the fitted regression curve first becomes better and then stabilizes at a relatively good level. Therefore, LSTM has a better effect on hidden_size than the RNN model. This may be inspired by comparing the loss curves of the two models under the same hidden_size=50. The loss curves of the RNN and LSTM models under hidden_size=50 are as shown in the figure:

RNN loss curve:


Figure 23: The loss curves of the RNN under hidden_size=50

LSTM loss curve:

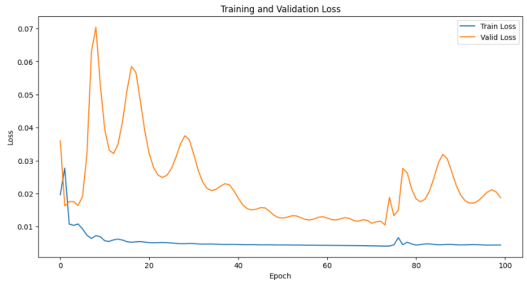


Figure 24: The loss curves of the LSTM under hidden_size=50

It can be found that the loss curve of RNN is "steeper" than that of LSTM, that is, the gradient changes more drastically. The reason may be that LSTM benefits from the existence of memory gates so that the model is not as sensitive to short-term changes as RNN, which can produce a smoother loss curve and further affect the model performance.

Increasing num_layers, or model depth, to RNN models and LSTM will lead to degradation in model expression ability, which is different from our general understanding of deep neural networks. It is speculated that this is because the feature space generated by the model's features is too simple, allowing the model to learn features well at the depth of the baseline.

Both models show the characteristic that the regression curve approaches a horizontal line when the depth is too deep, for which we put forward two hypotheses. One is that the gradient disappears in the model, which is very common in deep neural networks. So we tested the model performance by adding a batch normalization layer to the RNN model. Experiments show that with the batch normalization layer, the performance of the model is improved. Specifically, when the complexity of the original model degenerates into a horizontal line, adding batch normalization can change this phenomenon. Unfortunately, if we continue to increase the depth of the model on the basis of adding the batch normalization model, we find that the model will still degenerate into a horizontal line. The regression curve of the RNN model with batch normalization added under num_layers=15 and the regression curve under num_layers=25 are shown in the figure:

Model output of num_layers=15 with batch normalization:

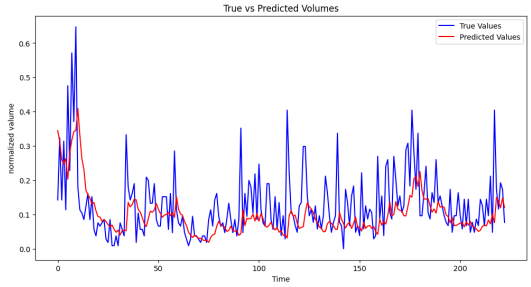


Figure 25: The loss curves of the RNN under num_layers=15 with batch normalization

Model output of num_layers=25 with batch normalization:

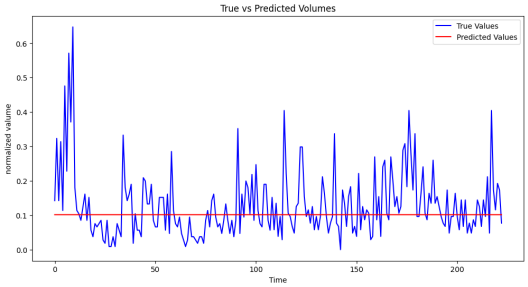


Figure 26: The loss curves of the RNN under num_layers=25 with batch normalization

The second is that the model increases network complexity but does not optimize other parameters, or the feature space is too simple and the model decision boundary is too complex, causing the model to finally converge at the local minimum. This local minimum is a special case where the model believes that only a horizontal line is needed to achieve the lowest MSE loss. This is an interesting phenomenon and can be a way of cheating the model. In fact, this phenomenon is not only observed in regression models, but also in some cases where RNN and LSTM are used for classification tasks, the model only predicts one result to achieve a better accuracy.

At the same time, in the process of degradation of model expression ability, it is observed that LSTM has completely degraded to close to the horizontal line when num_layers=15, while RNN degrades to the horizontal line when num_layers=25. One thing that needs to be clarified here is that although the LSTM model introduces a long-term memory mechanism based on the RNN model, the long-term memory mechanism is more to solve the forgetting problem caused by long sequence input. Regarding the number of network layers stacked in the model, in the experiments of this report, LSTM was more sensitive to the depth of the model than RNN, which is a point that needs to be correctly distinguished.

## 7. Conclusion

Based on the prediction of trading volume in the stock market, this report designed hyper-parameter experiments on the two parameters of model depth and model breadth for the two models of LSTM and RNN, and attempted to compare the performance within the parameters of the model and between models. Combining the regression curve, visualization results and loss function, this report proposes the following interesting conclusions, which also have the potential for subsequent in-depth research:

1. For data with larger variance, more violent fluctuations, and greater differences between high and low frequencies, the neural network's performance on such data is not as good as more stable data due to its own limitations.

2. To a certain extent, the increase in model breadth of RNN and LSTM helps to strengthen the learning ability of the model, and LSTM is more receptive to such hyper-parameter adjustments than RNN, although RNN's learning ability will decrease when the model breadth is large, RNN still has good recognition ability for data peaks and troughs with obvious fluctuations.

3. Blindly increasing the depth of the model will not help the model learn more complex patterns, but may lead to a significant degradation of the model's learning ability. This degradation is manifested in the model using "cheating" means to make predictions. In the classification task, this "cheating" is reflected in the model selecting the label result with the highest frequency as the prediction result, and in the regression task, the model selects a horizontal line with a certain intercept as the prediction result. There are some ways to slow down this cheating such as adding a batch normalization layer, but it cannot be effectively avoided completely.

## References

[1] Yash Bohra. The challenge of vanishing/exploding gradients in deep neural networks. *Data Science Blogathon*, 2021.

[2] Minshuo Chen, Yu Bai, Jason D Lee, Tuo Zhao, Huan Wang, Caiming Xiong, and Richard Socher. Towards understanding hierarchical learning: Benefits of neural representations. *Advances in Neural Information Processing Systems*, 33:22134–22145, 2020.

[3] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] T Smith. Random walk theory: Definition, how it's used, and example. *Tilgjengelig fra: https://www. investopedia. com/terms/r/randomwalktheory. asp. Hentet*, 24:2023, 2023.

[6] Ralf C Staudemeyer and Eric Rothstein Morris. Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

[7] Wikipedia contributors. Mean squared error — Wikipedia, the free encyclopedia, 2023. [Online; accessed 16-November-2023].

[8] Wikipedia contributors. Stock market prediction — Wikipedia, the free encyclopedia, 2023. [Online; accessed 16-November-2023].