FINAL REPORT

# Classification and Regression for Stock Price Prediction with LSTM, TCN and Other Models

December 30, 2018

Name: Qingshan Yao
Student ID: 516030910539
IEEE, CS, SJTU

# INTRODUCTION

High-frequency trading is one of the important topics in the field of quantitative trading, among which the prediction of stock price is the most important. The course design content of this course is to predict the mean value of the middle price at the next 20 time points through the study of the data in the order book.

Through the analysis of the problem, it can be found that it is undoubtedly a time series forecasting problem to predict the mean of the median price of the next 20 through the data of the first 10 time nodes.

I think the price forecasts are different from other types of forecasts.We can simply regard it as a regression problem and that's what most people think.But at the same time it is a good idea to regard it as a classification problem, through the classification of rise, fall and maintain the same, we can also achieve good results.

In the process of completing this task, I tried a variety of data processing methods and models, and in this article I will mainly show the LSTM model, TCN(Temporal Convolutional Network) model and some machine learning methods. I will analyze each model in depth and show the problems and solutions encountered in the implementation process, as well as some reflections on them.

## MODEL ONE: LSTM

## WHY I CHOSE IT?

*As this is a popular model, I am not going to show many details of itself.*

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs.

Through this feature, we can design an LSTM network with a time step of 10 to learn which features of which points in the ten time nodes are important for prediction, which are relatively redundant, and how the features interact with each other.

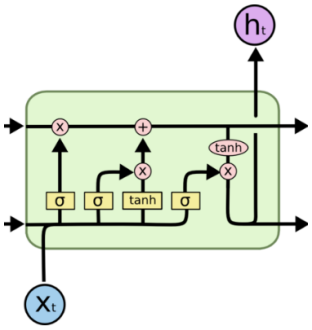


**Figure 1.1:** LSTM Cell

# DATA PROCESSING AND FEATURE SELECTION

## 1

Firstly, we need to compare the differences between training dataset and testing dataset. The time intervals in training dataset have many types: 1s, 3s, 5s and even no interval. But there is only one kind of interval in the testing set:3s. So we need to decide whether to delete different intervals in the training set.

At the same time, the data in training set covers many days. The data in different days are not continuous and we need to avoid the 10 put in a time step contains data in two or more days. Besides, I have also considered the data in the morning and afternoon, by some tests, these data are related.

Another major difference is the distribution of these two datasets. If we use a Gaussian Distribution to represent them, their mean values and variances are quite different. This is a troublesome fact when we apply the normalization. On the one hand, if we use the mean and variance of training dataset to normalize the test set, it is ridiculous and the result is bad. On the other hand, if we use the mean and variance of all the data in testing set to normalize it, it is a data breakdown. That is because we use the data in the future to process data at present. Although this use is not for prediction, it can make present data have future properties.

## 2

Therefore, in my implementation, I try a new way to normalize the data: normalize on each time-step and predict the difference value. There are some key points in this method:

1)We must use difference value prediction instead of predicting the result directly. That is because when we apply normalization on each time step(that means 10 rows of input

features), we only consider the distribution of these data. For example, in time step 1 and time step 2, they become the same distribution after normalization but their mean value is different, such as 3.4 and 3.7. If we use the final result to predict, it may predict 3.7 as 3.4 or 3.4 as 3.7. But if we use the difference value, it will work well.

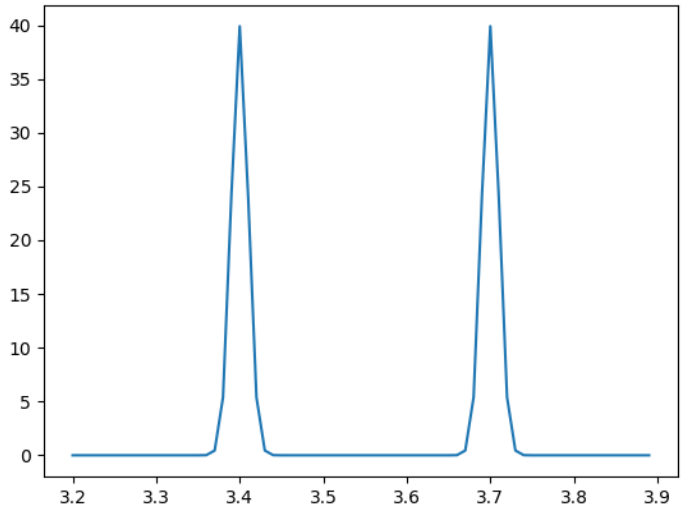


**Figure 1.2:** Distribution example

2)In many time steps, the variance is zero.In the formulation:

$normal = (data - mean)/standard\ deviation$

zero variance will produce a mistake. So I try to add a constant to all the standard deviation to avoid this mistake. I tried many numbers with different orders of magnitudes and found 0.1 and 0.01 are good choices.

## 3

As for the feature selection. I choose all the features at first because I think the "Time" should be an important feature as the stock price at the beginning of trading and after some time should be distinct. But by comparing the result of using and not using "Time", I found my idea was wrong. Actually, the "Time" information is already "stored" in other features. And the one of the meanings of LSTM is just pick out these features. So using "Time" redundant.

Besides, "Volume" is not a important feature, I have tried to use $\Delta V$ to replace this feature, but the result shows no much change. So I use 'MidPrice', 'Volume', 'BidPrice1', 'BidVolume1','AskPrice1','AskVolume1','LastPrice' as my input feature. And the normalization is done

as showed above.

## NETWORK STRUCTURE AND HYPER-PARAMETER

I used a simple but clear structure: two layers of LSTM and one linear output linear. In the process of testing the network, I have tried to deepen or widen the network in order to get a better result, and the change of score with different choices are showed below:

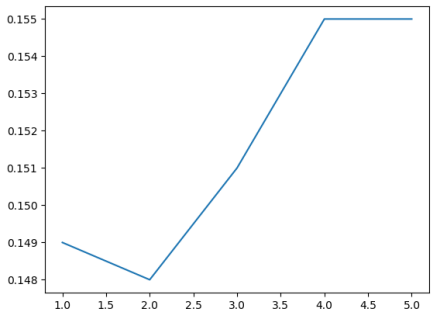


**Figure 1.3:** score-layer number

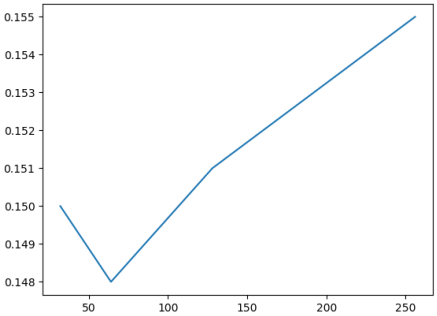


**Figure 1.4:** score-hidden size

Therefore, I chose to use two layers of LSTM and both of them have the hidden-size of 64.

By the same way, I tried to tune the learning rate and batch size. And the final result is 0.001 and 64.

Another parameter is the epoch number. In a normal work, we need to set this parameter to a number that is enough to converge and not too large to overfit. In the LSTM model, I found a number smaller than 10 shows good ability. And 5 may be the best choice.

### DATA SHUFFLE

Another import operation when processing the data is "shuffle". There is a problem: if we shuffle the dataset without organized as batches(that means the same condition as batch size equals to 1), the result will be pretty bad. But if we firstly divide it to batches and shuffle them, the result will be good.

A possible explanation I came up with is that many information are "stored" in a longer time step than 10. The use batch is to magnify these information . But if we just shuffle the original dataset, many information will "cancel" each other. A example is two time step data with similar distribution, but different conditions happened in the next 20 steps, then one's price will go up, the other's will go down, this two data will "cancel" each other. If the batch size is chosen properly, this probability of this kind of thing will become small. So we will get a acceptable result.

## CONCLUSION

LSTM is the most popular model in this project, but I think I am not just using it but learning it. As far as the result is concerned, I think around 0.00145 is the upper bound of this model. Maybe better feature engineering and data processing will make a better result. But this kind of progress will be little and I think that was not the key point of this project.

# MODEL TWO: TEMPORAL CONVOLUTIONAL NETWORK

## 1.1 WHY I CHOSE IT?

*This is the last model I tried on the last day. This model's result may be not outstanding, but the advantages are speed and its novel architecture. And I think if time permits, I can tune it to the same or even better result as LSTM*

Since this is not an article dedicated to this model, I will use my own understanding to briefly summarize this model:

In temporal convolutional network, both one-dimensional causal convolution and extended convolution are used as standard convolution layers, and every two such convolution layers and identity mapping can be encapsulated as a residual module (including relu function).Then the residual modules stack the deep network and replace the full connection layer with the full convolution layer in the last few layers.

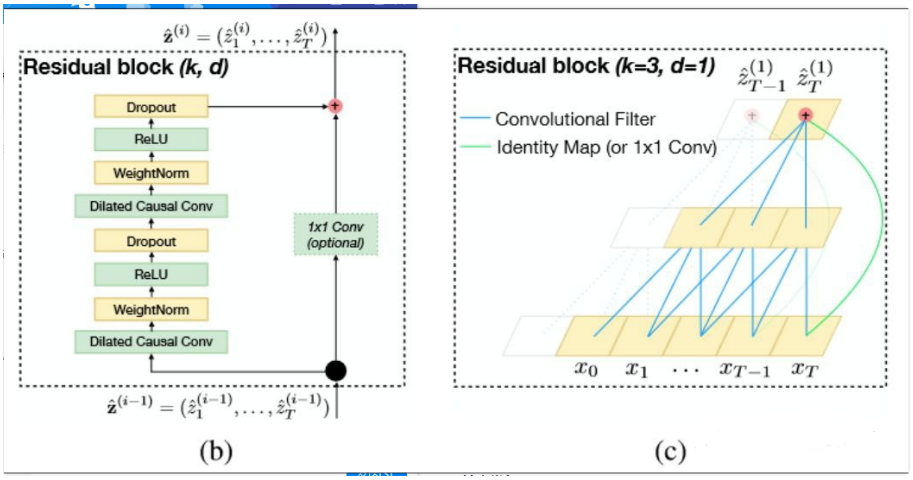


**Figure 1.5:** Residual Block

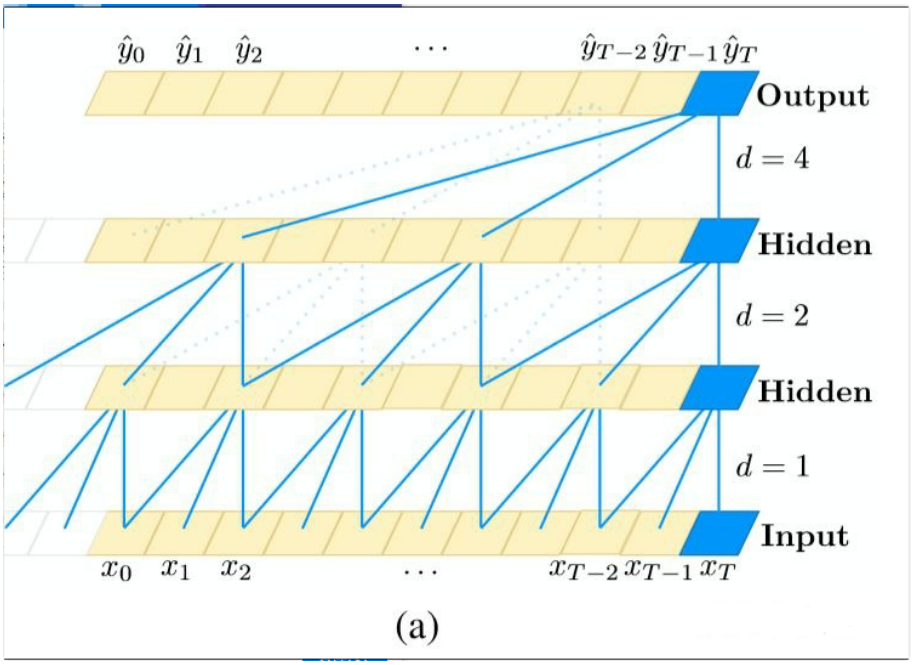


**Figure 1.6:** The network

The convolution layer of TCN combines extended convolution with causal convolution.The purpose of using causal convolution is to ensure that the prediction of the previous time step does not use future information, because the output of time step t will only be obtained based on the convolution operation of t-1 and the previous time step, which is very similar to LSTM.

At the same time, it can be seen that the convolution of TCN is very similar to the ordinary one-dimensional convolution, except the biggest difference is that the expanded convolution is used. As the number of layers increases, the larger the convolution window is, the more

empty holes in the convolution window will be.This gives the whole convolutional layer a greater sensory field and a better ability to learn from the past.

In addition, it is worth mentioning that there are two layers of extended convolution and ReLU nonlinear function in the residual module of TCN (that is, figure b), and the weight of convolution kernel is normalized by weight.In addition, in TCN, Dropout is added to each cavity convolution in the residual module to achieve regularization.

## 1.2 DATA PROCESSING AND FEATURE SELECTION

The data processing is almost the same with LSTM model.

But unlike LSTM, add the time property to TCN will get a more excellent performance, I think this could be due to the difference between the network structure, LSTM information transfer is achieved by three kinds of gate, and TCN is through expansion convolution and causal convolution, therefore, TCN can better learn from more feature information, and has faster speed.

At the same time, there is no concept of time step in TCN, so features are passed in once through a one-dimensional array.

## 1.3 NETWORK STRUCTURE AND HYPER-PARAMETER

At the beginning, I tried a similar simple network structure, for example, a residual block with 64 channels.But it doesn't work very well.The difference between convolutional neural network and LSTM is that it needs to transfer information layer by layer, and extract important information in time series by expanding convolution and causal convolution, and then approximate it.

After a lot of attempts and interpretation of the original paper, I found that the network structure of four to six layers is most suitable for this task, and the number of channels in each layer should be decreasing (as shown in the previous figure).The following figure shows the structure of each layer (i.e., each residual block), but the difference between each layer is the number of channels. Finally, I built a TCN network with four layers, with each layer having a number of channels of 128,64,32, and 16 respectively.

As for the selection of other hyper parameters, due to the limited time, I did not try too much, but determined a set of relatively good values in a similar way to LSTM.

```
(network): Sequential(
  (0): TemporalBlock(
    (conv1): Conv1d(10, 128, kernel_size=(2,), stride=(1,), padding=(1,))
    (chomp1): Chomp1d()
    (relu1): ReLU()
    (dropout1): Dropout(p=0.2)
    (conv2): Conv1d(128, 128, kernel_size=(2,), stride=(1,), padding=(1,))
    (chomp2): Chomp1d()
    (relu2): ReLU()
    (dropout2): Dropout(p=0.2)
    (net): Sequential(
      (0): Conv1d(10, 128, kernel_size=(2,), stride=(1,), padding=(1,))
      (1): Chomp1d()
      (2): ReLU()
      (3): Dropout(p=0.2)
      (4): Conv1d(128, 128, kernel_size=(2,), stride=(1,), padding=(1,))
      (5): Chomp1d()
      (6): ReLU()
      (7): Dropout(p=0.2)
    )
    (downsample): Conv1d(10, 128, kernel_size=(1,), stride=(1,))
    (relu): ReLU()
  )
```

**Figure 1.7:** Residual Block in Pytorch

## 1.4 CONCLUSION

I tried this model on the last day. My best score on the public board was about 0.00150, but I put it in the second place because of its novel network structure.At the same time, in many other similar tasks, TCN has surpassed LSTM in both speed and accuracy, so many people believe that the king in the field of time series has gradually changed from LSTM to TCN.

And in my mind, there are many advantages of TCN:

Speed matters.Faster networks make feedback loops shorter.Since large-scale parallel processing can be carried out in TCN, the time of network training and verification will be shortened.

TCN provides more flexibility to change the size of the sensing field, mainly by stacking more convolution layers, using larger expansion coefficients, and increasing the size of the filter.These operations can better control the memory length of the model.

The reverse propagation path of TCN is different from the time direction of sequence.This avoids the gradient explosion or gradient disappearance problems that often occur in RNN.

Less memory is required for training, especially for long input sequences.

# MODEL THREE: XGBOOSTING

## WHY I CHOSE IT?

The idea of this algorithm is to keep adding CART trees—CART regression tree assumes that the tree is a binary tree and splits the features continuously.For example, the current

tree node is split based on the JTH eigenvalue. Let samples with the eigenvalue less than s be divided into left subtree and samples with the eigenvalue greater than s be divided into right subtree.CART regression tree essentially divides the sample space in this feature dimension, and the optimization of this space division is a NP hard problem. Therefore, heuristic method is used to solve this problem in the decision tree model—, keep splitting features to grow a tree, and each time adding a tree, it is actually learning a new function to fit the residual predicted last time.When we finish the training and get k trees, we need to predict the score of a sample. In fact, according to the characteristics of this sample, a leaf node will fall on each tree, and each leaf node will have a score. Finally, we just need to add up the score of each tree and it will be the predicted value of the sample.

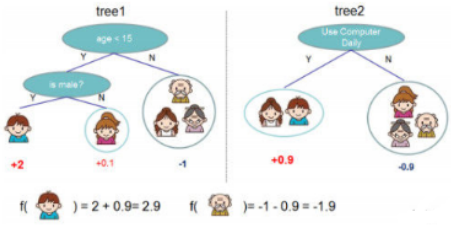


**Figure 1.8:** XGBoosting

The idea of this model is extremely simple, and because of its excellent universality and practicality, it is very popular in the field of machine learning.The reason why I choose this model is that there is the concept of model fusion in its algorithm, and the fusion of different regression trees can better adapt to this task.

## DATA PROCESSING AND FEATURE SELECTION

### 1

There is no difference between the data preprocessing and the previous one, so I won't go into details here.

### 2

"The data determines the upper bound of machine learning, and the algorithm just tries to get close to that upper bound."Feature engineering is the most important part of this model that can work.

Only through feature construction, feature selection and reasonable feature selection can this model get excellent results.If you don't do anything but pass seventy (or eighty) features of a time step directly into the model, it doesn't work very well.

For construction, it takes time to look at the raw data, think about the underlying form and structure of the problem, and be sensitive to the data and have hands-on experience with machine learning to help build features.

Let me give you a couple of examples of what's called representation.Since it is different from LSTM, it cannot automatically analyze time series problems, so it adds time attribute characteristics.For example: the derivative of each feature, that is, the difference with the previous moment.

$$V_t = V_t - V_{t-1}$$

Claim to buy quantity, claim to sell quantity, claim to sell one price, claim to buy one price to decide whether the transaction succeeds or not.So we can add a feature in this way:

$$F_{new} = BP1 * BV1 - AP1 * AV1$$

In this way, we can add a lot of new features.

Sometimes, you may notice that some features have much higher span values than others.For example, compare a person's income to their age, and more specifically, some models (like ridge regression) require you to scale the eigenvalues to the same range.Scaling prevents some features from getting very different weights than others. So standardization or normalization is also necessary.

After that, feature extraction is carried out.I mainly used PCA principal component analysis to find the optimal subspace of data distribution through coordinate axis transformation, so as to achieve the purpose of dimensionality reduction and de-correlation.

Finally, because feature selection is a process of repeated iteration, sometimes I may think that feature selection is done very well, but the actual model training is not very good, so every feature selection must use the model to verify, the ultimate goal is to obtain data that can train a good model, improve the performance of the model.

Feature selection is to eliminate irrelevant or redundant features, reduce the number of effective features, reduce the time of model training, and improve the accuracy of the model.Feature extraction achieves dimensionality reduction through feature transformation, while feature selection relies on statistical methods or the feature selection (sorting) function of the machine learning model itself to achieve dimensionality reduction.

I try to use linear regression for selection, because the more important features will have larger coefficients in the model, while the more independent features are from the output variables, the closer the corresponding coefficients will be to 0. After removing some features,

the training can be carried out.

## PARAMETER TUNING

I used the API in sklearn and the parameter tuning can also be done by some built-in functions. The commands are showed as following:

```
model = XGBRegressor()
optimized_GBM = GridSearchCV(estimator=model,param_grid = cv_para,scoring='neg_mean_squared_error',cv=5,verbose = 1, n_jobs = 4)
optimized_GBM.fit(features, labels)
eval_res = optimized_GBM.score
print("每轮结果：{0}".format(eval_res))
print("参数取值：{0}".format(optimized_GBM.best_params_))
print("最佳模型得分：{0}".format(math.sqrt(abs(optimized_GBM.best_score_))))
```

**Figure 1.9:** score-layer number

## CONCLUSION

The implementation of this model itself is very simple, because there are ready-made interfaces available. The key and difficulty lies in the implementation of feature engineering and the adjustment of parameters. Through continuous attempts and adjustments, I got a score of about 0.00149 on the public board.

## MODEL FOUR: CLASSIFICATION WITH MANY MODELS

## WHY CLASSIFICATION?

Most people solve this task as a regression task, but I tried all sorts of classification methods from the beginning.There are several reasons:

First, if one looks closely at the curve of a stock, one can see that it is never smooth, there are always many bumps, and it is very difficult to fit this frequent small range of violent fluctuations by regression.

Second, we can get some inspiration from "difference prediction". Instead of learning what the next moment should be, it is simpler and more accurate to predict how much change it takes compared to the last second.

Third, in this task, categories are divided by ourselves, which provides great flexibility and operability. We can not only determine the distribution of data through the selection of thresholds, but also control errors to a certain extent.

## CLASS SELECTION

I think of this problem as a tripartite problem, where prices go up, prices go down, prices don't change.At first, I set the threshold to 0.001, and finally found that most of the data was concentrated in the invariant interval, which led to the final classification results being all invariant.Then, we should adjust the threshold value reasonably to make the distribution of the three types of data as uniform as possible. After some mathematical operations, we found that the threshold value between 0.0004 and 0.0005 was the best effect.

Then, I tried to increase categories, namely, up 0.0005, up 0.001, the same, drop of 0.005, and so on, but the same problem, but this time it is difficult to solve, because the data quantity is large, it is hard to reasonably setting threshold, let so many categories of each category is very uniform, destined to some of the categories of data are insufficient, and for this problem, we can't artificially generated some data, difficult to solve this problem.

Finally, I chose to use three classifications and conduct experiments on multiple models with a threshold of 0.005.

## CONCLUSION

First, I conducted experiments on the above three models.Experiments show that LSTM classification can reach the same level as regression.However, TCN and XGB have poor effects, which may also have something to do with insufficient tuning parameters.Then I tried some machine learning methods, such as Random Forest. And the results are good.

But the point is, no matter how I adjust the threshold and classification, I still can't achieve better results.It seems that this is the limit of the classification method, which I think needs further study.

## DISCUSSION FOR OTHER PROBLEMS

## DATA DISTRIBUTION PROBLEM

This problem has been bothering me all the time. Through error analysis, I can determine that the test set and the training set are distributed differently, or even greatly different, which brings great difficulty to the whole task.Because we have to predict the current data with another distribution data without data breakdown, which is undoubtedly very difficult.At the same time, the data of the training set are very messy, even the distribution of time steps is

not uniform, there are one second, three seconds, and the same.And after processing these data, the effect is not a good improvement.I think there are some problems with the data provided in this experiment. If more rigorous data can be provided, the upper limit of many models will be raised.

### THE RIGOR OF THE PROBLEM

Using the information of the first 10 time nodes to predict the mean of the middle price of the second 20 is also a little confusing.The normal way of thinking is to use more data to predict less data.Because there are a lot of things that can happen in the last 20, and the first 10 don't reflect that at all.This is why the effect of data breakdown will be so much improved.It's like the squeeze theorem in math. It's also like watching a TV series.I only know the beginning, so it's hard for me to predict what's going to happen in the middle, but if I watch the finale in advance, then of course I can get a sense of what's going on in the middle.

### FEATURES PROBLEM

Careful analysis of eight characteristic information, will find that many of them are related, and even some of them are strongly related, and the advantage of neural network was, although it does not know what is associated, but it can be trained to learn these associations, which means we only need to do some basic work of regularization, don't need to do too much with characteristics (except XGB), and the facts also prove that, only use the midprice and with the effect of the seven characteristics were similar.

## FINAL CONCLUSION

Although the process was very bumpy, this course assignment was of great significance to me.I not only consolidated the knowledge taught by teachers in class, such as hidden markov model, deep learning, and reinforcement learning, but also greatly expanded my scope of knowledge by consulting materials and reading papers, and promoted my understanding of the field of artificial intelligence.And the solution of practical problems has trained my practical ability.Many model choices, diverse data processing, and very magical tuning parameters have made me deeply feel the charm of this discipline, but at the same time, I also feel its extensive and profound.This assignment is just the beginning of my journey. I

have a long way to go on this road.Thank you very much for your guidance and help.I think this is a successful assignment.

# REFERENCES

Shaojie Bai , J. Zico Kolter , Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*Available at: <https://arxiv.org/pdf/1803.01271.pdf> [Accessed 29 Aug 2016]

Alec N.Kercheval. *Modeling high-frequency limit order book dynamics with support vector machines* 2013

Colin Lea, Rene Vidal, Austin Reiter, Gregory D. Hager. *Temporal Convolutional Networks: A Unified Approach to Action Segmentation*Available at: <https://arxiv.org/abs/1608.08242> [Accessed 19 Apr 2018]

Online course: https://morvanzhou.github.io/tutorials/machine-learning/torch/