REPORT FOR PROJECT 2

Li Chengxin (20745826)
Li Xinyi (20745163)
Liang Xin (20745785)
Lu Lanxi (20744822)

November 14, 2021

Abstract

We work on the paper which considers using methods that flexibly learn price patterns that are most predictive of future returns to forecast future returns, rather than testing hypothesized or pre-specified patterns. We follow the steps, using the images of OHLC as raw predictor data, splitting train and test data by the year of 2000, building and training the CNN model to find the best parameters with considering some details in the paper, such as Xavier initialization and normalization. Our model has an accuracy of 0.509 out of samples after 1900 iterations, which is the threshold for overfitting we find through many trials. Then we test the model under different situations and parameters, find the baseline has the best performance.

1 Introduction

In this project, we use the transformed OHLC charts images dataset introduced in the paper. Our aim is to reproduce the convolutional neural networks(CNN) showed in the paper.

The input to the CNN is typically an image which is a plot of past market information(open, high, low, close prices, and volume) over past 20 days represented as a 2D matrix. The output of the CNN is a set of stock-level estimates for the probability of a positive subsequent return over last 20-day horizons. The CNN achieves out-of-sample classification accuracy with an accuracy 0.509.

2 The CNN Model

The CNN is a modelling scheme that stacks together a sequence of operations to transform a raw image into a set of predictive features and, ultimately, a prediction. It usually consists of multiple building blocks, each of which consists three operations: convolution, activation, and pooling.

We build the 20-day CNN models of the original paper, which are built with 3 CNN building blocks, and their structure is shown below.

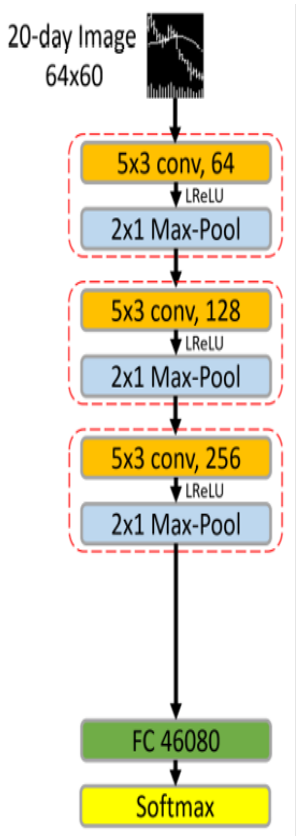

Figure 1: Diagram of CNN Model

In addition to the above structure, according to the original paper, we use the batch normalization layer between the convolution and activation within each building block. We also apply the Xavier initializer for weights in each layer, and 50% dropout to the fully connected layer.

Especially for the sparseness of the first block, we use horizontal and vertical strides of 1 and 3 and vertical dilation rate of 2 and 3 in this block. Apart from this, we also try to use that in other blocks which has the convolutional layers, which has an average accuracy of test data about 50.9%.

It's worth noting that, unlike the original paper, the number of FC neurons we set is consistent with the output of the last blocks (256*8*60), not 46080.

Our data processing procedure is same as the original paper, we randomly select 70% images of eight-year sample for training and 30% for validation. The remaining nineteen years of data is for testing.

As for the label of the images, firstly we choose "Retx_20d" to make a category by ourselves, and then we choose "Retx_20d_label" in the label data, which means the return of the following 20 days after the last day in the images is positive or negative. We notice that there are 0, 1, 2 three types of labels, representing down, up, and nan. We do not think the label of 2 is useful to improve our model, so we delete them in both training and testing data.

When we train the model, for every 220 batches (batch size is 128), we validate the model and calculate the loss. We use early stopping to halt training once the validation sample loss function fails to improve for two consecutive epochs.

Below are the plots of the loss during training, we find that the loss is stable after 80 iterations and the value of it starts to grow after about 1200 iterations. At this time, the limits we set that loss value of the validation samples cannot be decreased for two consecutive epochs halt the training too. We also plot some of our validation result. As we do the validation after several times of iteration of train, so the amount of data for validation is smaller than the train part's.

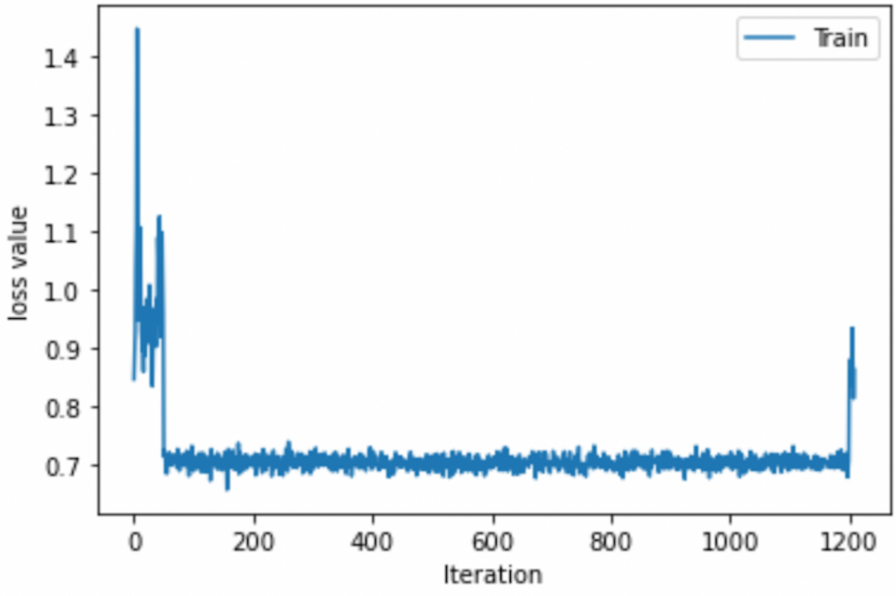

Figure 2: Train Loss

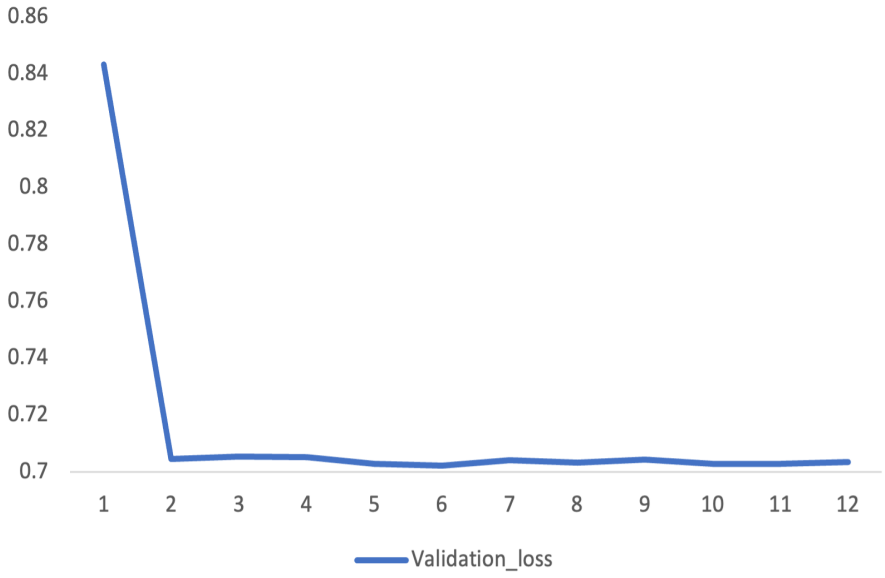

Figure 3: Validation Loss

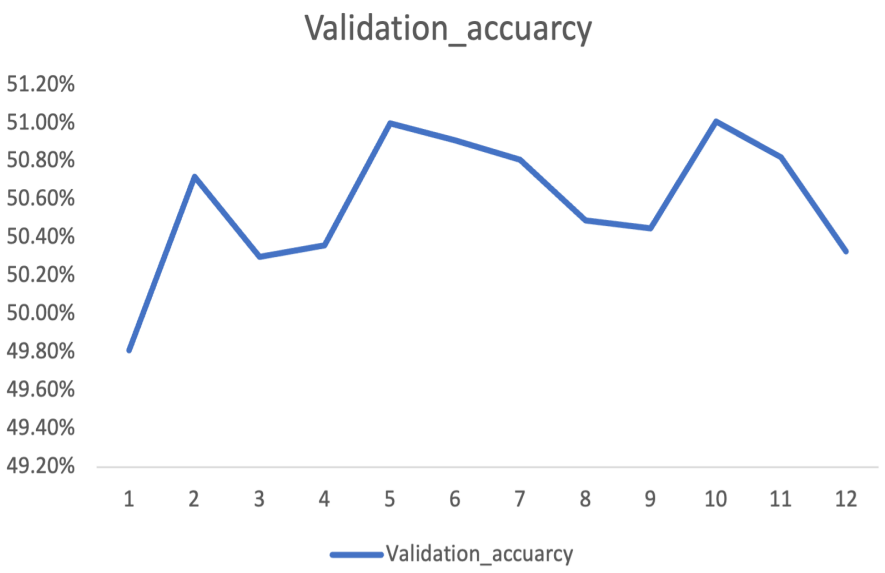

Figure 4: Validation Accuracy

2.1 Sensitivity Test

The table below illustrates the performance sensitivity to the model including the dropout probability, use of batch normalization (BN), use of Xavier initialization, and type of activation function. The columns indicate the average cross-entropy loss and prediction accuracy on the validation set and the test set.

		Loss		Accuracy	
		Validation	Test	Validation	Test
Dropout	0	0.698	0.7	0.515	0.516
	0.25	0.7	0.701	0.511	0.515
	0.75	0.785	0.803	0.512	0.494
BN	no	0.715	0.712	0.493	0.509
Xavier	no	0.705	0.711	0.516	0.499
Activation	ReLU	0.706	0.717	0.517	0.496

Figure 5: Sensitivity Test

The dropout probability almost does not vary the model performance, although raising it to 0.75 produces a higher loss. The performance is also insensitive to whether or not to include the batch normalization step or Xavier initialization. However, the accuracy decreases after switching from leaky ReLU to ReLU.

Therefore, the model performance is robust to the model construction.

2.2 Prediction for Other Labels

We also apply the baseline model to predict labels of daily,five-day return and sixty-day return. The highest is the accuracy of the daily return, which is about 0.65 for train data and 0.64 for validation data after 2000 iterations. The accuracy of the five-day return for validation data and test data is 0.507 and 0.510 separately. Besides, the accuracy of the sixty-day return for validation data and test data is 0.495 and 0.486. Hence, this model predicts the five-day label with the best accuracy.

The below plot is the loss trend through the iteration

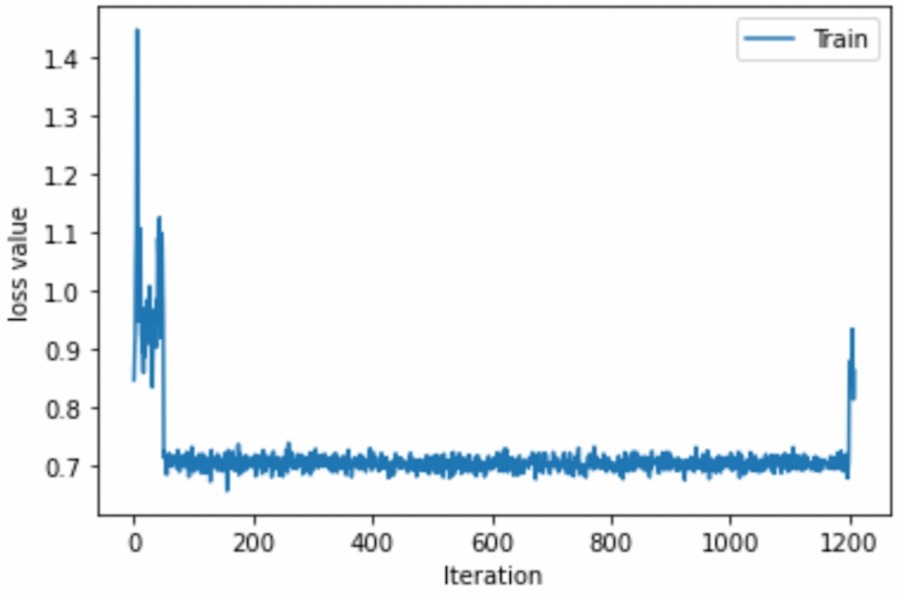

Figure 6: Loss for Daily Return

3 Conclusion

We use the image dataset as the input to CNN model and get the output of the probability of a positive subsequent return from the model. After several trails, we find the baseline model performs the best. Sharing the same opinion of the paper, we also agree that the details of training process, such as Xavier initialization, batch optimization, and drop-out probability, are important. The accuracy varies from different length of periods; in other words, the closer the forecasted days, the higher the accuracy of the model.