(Re-)Imag(in)ing Price Trends

WANG Yufeng, 21020948
Independent Project R2 (4499)
Department of Mathematics
HKUST

1 Introduction

Convolutional neural networks (CNN) show remarkable performance in image classification, object detection and segmentation. In this project, we will refer to the paper entitled (Re-)Imag(in)ing Price Trends(Jiang et al., 2020) and reconstruct the proposed CNN model to predict prices from historical stock price dataset, the trend signals can be detected and provide the investors portable investing advice.

2 Data

The core basis of the data is to convert the stock data from a simple one-dimensional time series to a two-dimensional image, so that the CNN model can be trained, which may be more helpful to help the model understand the data relationship and improve the prediction effect of the stock trend.

OHLC bar charts use fewer pixels than candlestick charts to convey the same information. The 20-day intervals of daily data are combined and the heights of all images are uniformly scaled on a vertical axis to form an image-stock horizontal price OHLC chart so that each image in the sample shows the trend over a 20-day period. The chart includes the 20-day moving average price, and the "OHLC" bar represents the opening, high, low, and close prices for each day, as well as the daily trading volume of the stock, as shown in the example chart below as figure 1.

Figure 1: OHLC Chart

We used black for the background color and white for the chart to minimize the memory footprint of the picture. Each image consists of a matrix of pixel values of 64×60 data points, where the black pixel is 0 and the white pixel is 255. An example diagram is shown below as figure 2.

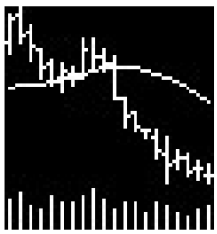


Figure 2: Data Picture Example

3 CNN Architecture

We model the predictive association between images and future returns using a convolutional neural network (CNN) shown in figure 3.

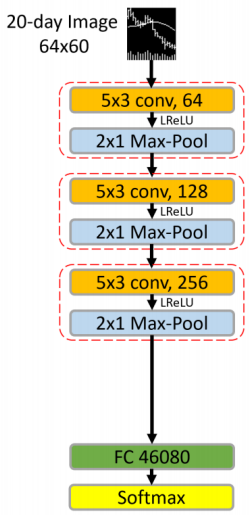
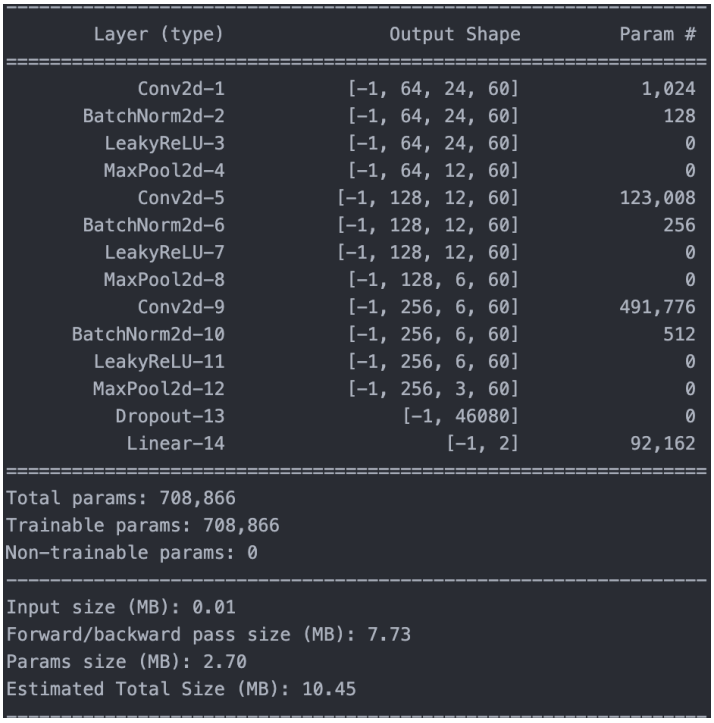


Figure 3: Diagram of CNN Model

A core building block concerns of three operations: convolution, activation, and pooling, We also choose a variety of different filter to extract different features. After passing convolutional layer, we normalized the batch. The existence of filters helps to identify some specific shapes in the picture. After using the filter, we go to the "activation" step. We select "leaky ReLU" as the activation function, takes the max of the filter output value and zero. The last step is "Max pooling", in this step, we continued to use the filter and select the maximum value in the range selected by the filter. Max pooling has two main functions, firstly, Max pooling can reduce the dimension, secondly, the method of selecting the local maximum also effectively avoids the disturbance of unimportant data to the final result. Within a building block, the output from all convolutional filters are fed element-wise through the leaky ReLU activation function. The output of ReLU activation function continues to the next building block. After processing, we can finally get the linear combination of vectorized pictures, after these linear combinations are processed by softmax function, we can get the probability of the rise of stock prices in the future.

We now discuss specific choices for our models. Since our images are largely sparse in the vertical dimension, we use 5×3 convolutional filters and 2×1 max-pooling filters. We use the same filter sizes in all layers for convenience. We use horizontal and vertical strides of 1 and 3 and vertical dilation rates of 2 for 20-day images, respectively, only on the first layer because raw images are sparse. Besides, we use horizontal and vertical padding of 7 and 1. The number of CNN building blocks in our model is based on the size of the input image. We use 3 blocks for 20-day images.

The network structure of our CNN model is shown in the figure 4.



Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 24, 60]	1,024
BatchNorm2d-2	[-1, 64, 24, 60]	128
LeakyReLU-3	[-1, 64, 24, 60]	0
MaxPool2d-4	[-1, 64, 12, 60]	0
Conv2d-5	[-1, 128, 12, 60]	123,008
BatchNorm2d-6	[-1, 128, 12, 60]	256
LeakyReLU-7	[-1, 128, 12, 60]	0
MaxPool2d-8	[-1, 128, 6, 60]	0
Conv2d-9	[-1, 256, 6, 60]	491,776
BatchNorm2d-10	[-1, 256, 6, 60]	512
LeakyReLU-11	[-1, 256, 6, 60]	0
MaxPool2d-12	[-1, 256, 3, 60]	0
Dropout-13	[-1, 46080]	0
Linear-14	[-1, 2]	92,162

Total params: 708,866
Trainable params: 708,866
Non-trainable params: 0

Input size (MB): 0.01
Forward/backward pass size (MB): 7.73
Params size (MB): 2.70
Estimated Total Size (MB): 10.45

Figure 4: Net Construction

4 Model Training

We divide the entire samples into training, validation and testing splits. In the original paper, they use the first seven-year sample (1993-1999) to train and validate the model, in which 70% of the sample are randomly selected for training and the remaining 30% for validation. Here, we follow their method while setting the seed of np.random as 10 and let the remaining twenty years (2000-2019) of data comprise the out-of-sample test dataset. The training goal is to minimize the cross-entropy loss which is defined as

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

and we use Adam algorithm to renew network weight. Moreover, to measure the classification accuracy, a true positive (TP) or true negative (TN) occurs when a predicted "up" probability of greater than 50% coincides with a positive realized return and a probability less than 50% coincides with a negative return. False positives and negatives (FP and FN) are the complementary outcomes

$$Auccracy = (TP + TN)/(TP + TN + FP + FN).$$

Then, we put image data storing in 64×60 memmap data type into the model. For the training process, we set the baseline of the batchsize at 128, learning rate at 10^{-4} and dropout rate at 50%. If the loss function does not improve in the validation set in consecutive two epochs then we stop training and save the model as our final model. By applying this mechanism, we can deal with overfitting problem as mentioned in the paper.

5 Sensitivity Testing

Table 1 summarizes the out-of-sample classification accuracy of CNN model in the 5-day and 20-day stock price prediction under different parameter settings. By calculating and comparing the accuracy, we can analyze the influence of parameters on the model.And figure 5 is a visualization of the accuracy of the 5-day and 20-day stock price prediction results respectively.

		5d-test-accuracy	20d-test-accuracy
Baseline		0.5285	0.5252
Filter	32	0.5318	0.5140
Filter	128	0.5325	0.5188
Dropout	0.75	0.5327	0.5240
Dropout	0.25	0.5066	0.5186
Dropout	0	0.5269	0.5144
Batch Normalization	No	0.5248	0.5246
Xaiver Initialization	No	0.5300	0.5198
Activation	ReLU	0.5246	0.5236
Pooling	2×2	0.5210	0.5006
Learning Rate	$1*e^{-5}$	0.5191	0.5225
Learning Rate	$5*e^{-5}$	0.5216	0.4851

Table 1: Sensitivity Test

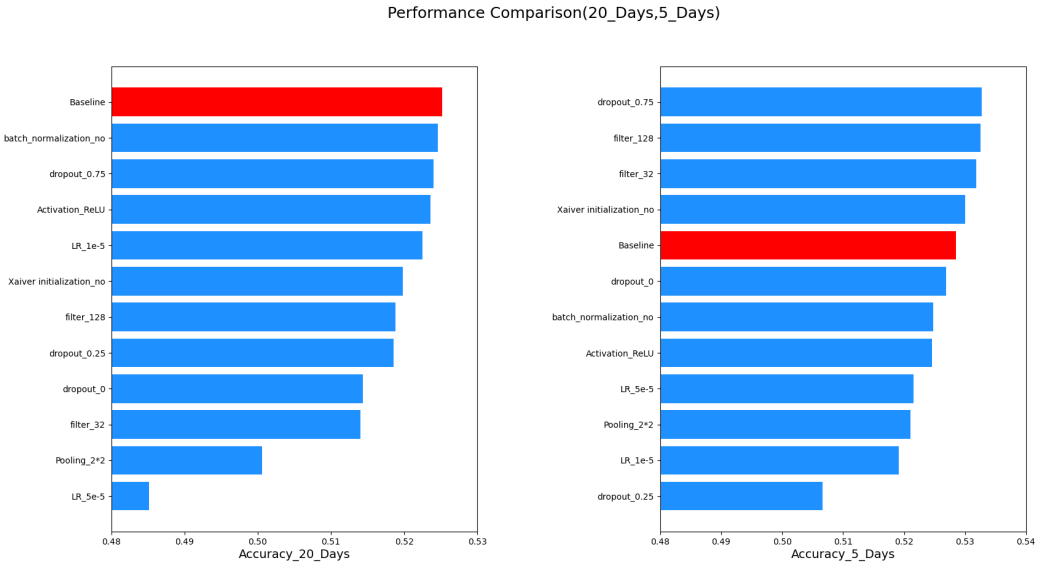


Figure 5: Performance Comparison

- **Filter**

  Typically, using smaller filters improves the sensitivity of the model because they can better capture details and features in the image, but also results in loss of information because large features cannot be processed efficiently. Using larger filters can handle large features better, but may result in overfitting and increased computational costs.

  The experimental results show that compared with 32 and 128 filters, the accuracy of the model reaches the highest when the filter is 64, indicating that the model has underfitting phenomenon when the filter is 32, and the model has overfitting phenomenon when the filter is 128.

- **Dropout**

  When training a CNN model, we can randomly remove a subset of neurons and let a different model learn each time to avoid overfitting, which is called dropout. We can see that when the dropout value increases, which means more neurons are dropped during training, the training accuracy of the network is effectively improved because of the decrease of overfitting.

- **Batch Normalization**

We perform batch normalization on the features of a certain layer in the network, so that the distribution of each feature in the same batch is similar, which can speed up the convergence of model training, make the model training process more stable, and solve the internal covariate shift problem caused by the high correlation and coupling between the layers of the network.

After deleting batch normalization, the accuracy of our model results decreased, indicating that batch normalization can improve the accuracy of model prediction.

- **Xaiver Initialization**

  Xaiver initialization solves the problem of the initial setup of the weight matrix W in the neural network, aiming to keep the variance of the activation function roughly the same during forward and back propagation, thus avoiding gradient disappearance or explosion and accelerating convergence as well.

  After deleting Xaiver initialization, the accuracy of our model results decreased, indicating that Xaiver initialization can improve the accuracy of model prediction.

- **Activation**

  LeakyReLU is very similar to ReLU, unless when the input is less than 0, ReLU inputs 0, while LeakyReLU inputs negative values with a tiny gradient, which can solve the neuron "death" problem.

  After we change the activation model from Leaky RELU to RELU, the prediction effect of the model decreases, because in the process of backpropagation, the gradient can also be calculated for the part of the input of the LeakyReLU activation function less than zero, avoiding the problem of gradient direction sawtooth, while RELU does not have such good performance.

- **Pooling**

  Pooling is used to reduce the dimension of the Feature Map in the CNN model. We try to change the size of the pooling window from 2×1 to 2×2, and find that the prediction ability of the model decreases, which may be due to the excessive loss of image information caused by the rapid decline of the feature dimension.

- **Learning Rate**

  The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.

  The paper uses a learning rate of 10^{-5} initially and 10^{-4} later. While we not only test the standard learning rates as demonstrated in the paper, but also test the performance of the model by setting learning rate at 5×10^{-5}. According to the performance in both cases, baseline learning rate 10^{-4} is better than others, which might because others get stuck in somewhere thus unable to make progress further.

6 Interpretation

It is difficult for us to give a clear explanation for the outcome of a network structure like CNN, but we still hope to give which areas the network pays more attention to.

Given an image and a class of interest (e.g. 'Up class OHLC image' or 'Down class OHLC image') as input, we forward propagate the image through the CNN part of the model and then through task-specific computations to obtain a raw score for the category. The gradients are set to zero for all classes except the desired class , which is set to 1. This signal is then backpropagated to the rectified convolutional feature maps of interest, which we combine to compute the coarse Grad-CAM localization which represents where the model has to look to make the particular decision. Finally, we pointwise multiply the heatmap with guided backpropagation to get Guided Grad-CAM visualizations which are both high-resolution and concept-specific.

We randomly select a sample from the dataset that were classified as 'Down', Figure 6 shows the original images and the grad-CAM heatmap for each of the three layers of the CNN. The brighter the area in the image, the greater the derivative of the output with respect to that area, and the more important the area is for predicting the label. We found that the first layer successfully captured the volume and price information of the stocks in the image, but it was relatively broad. The second layer gives more weight to high trading volume. The third layer focuses on the two longest shadow lines, that is, high intraday fluctuations, which is more consistent with common technical analysis.

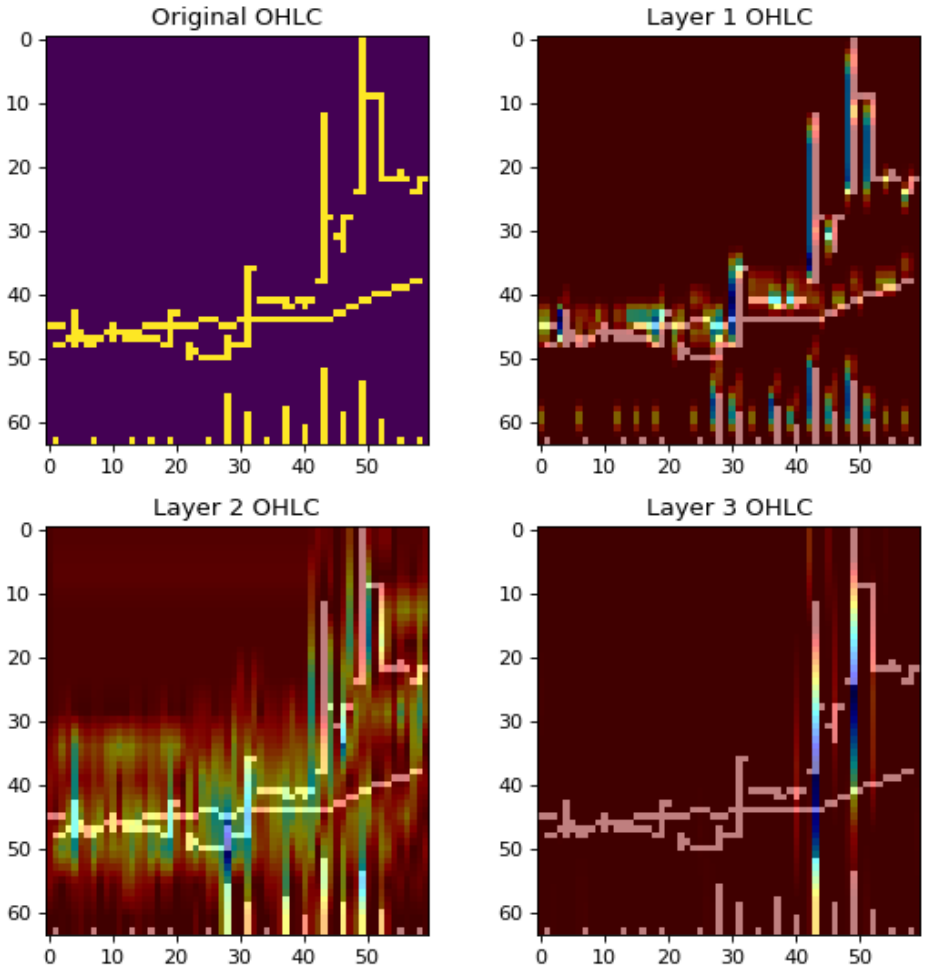


Figure 6: Grad-CAM Heatmap

7 Decile and Long-Short Portfolio Performance

In this part, we apply our baseline model to the test set to construct Decile portfolio and Long-short portfolio. Particularly, we use the "up" probability, which comes from the softmax layer to sort the sample into 10 decile and construct corresponding portfolio with equal weight. Decile 1 corresponds to stocks with the lowest probability of positive future return, Decile 10 stocks are most likely to earn according to the model.

We can see from the Figure 7 that the annual return trend of decile portfolio from 1 to 10 is very significant which means that the higher "up" probability shown form the CNN model, the higher the future return of corresponding stocks. So we can trust our model predictive power as it successfully grasp some features that are valid for a long time. Especially the return of decile 1 and 10 are more distinct from their nerghboring which give us the sense that we can construct a long-short portfolio by long buying decile 10 stocks and short selling decile 1 stocks.

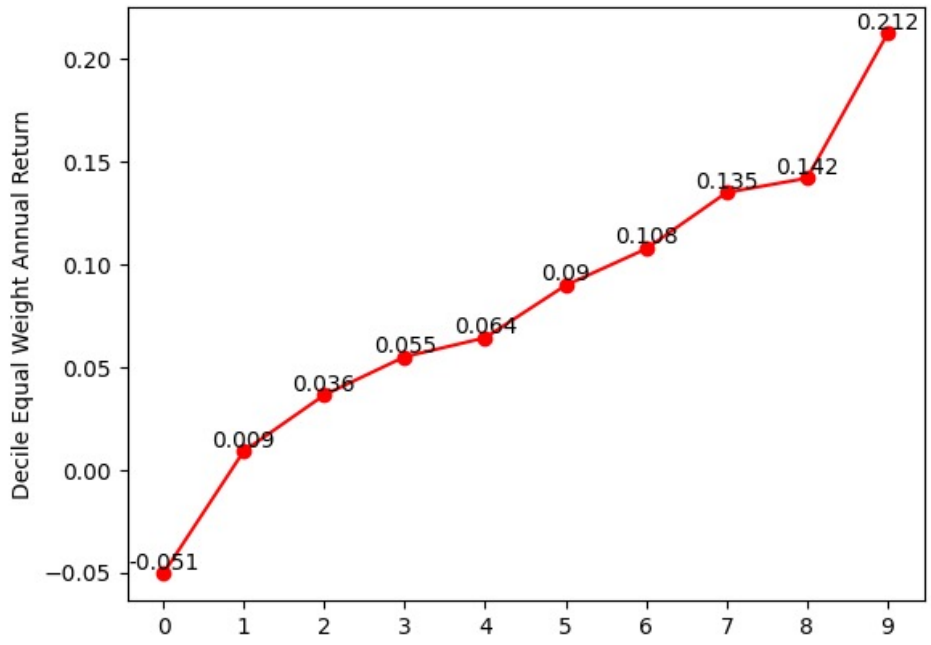


Figure 7: Decile Annualy Return

Then we get the long-short portfolio monthly and cumulative return shown in figure 8. We can see that except the extremely high return in the first several months after 2000, the returns in the whole test time period are relatively stationary and most of them are larger than zero. So we can see that the cumulative returns of long short portfolio continuously increase with little and acceptable drawdown. From 2000 to 2019, the average return of long short portfolio is 25.25%, Sharpe ratio is 1.74 and max drawdown is 5.33

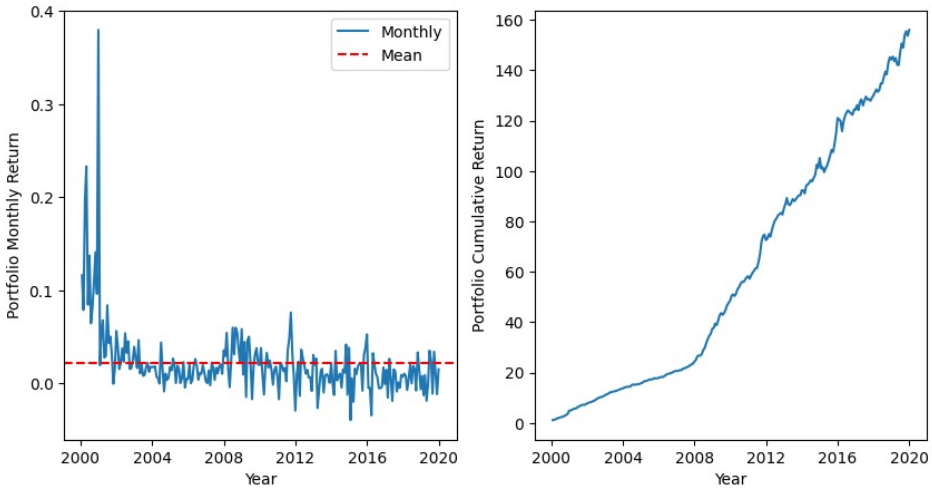


Figure 8: Portfolio Perfomance

8 Conclusion and Future Direction

8.1 Conclusion

From the sensitivity test, GRAD-Cam visualization, portfolio performance analysis, we find our model is robust, able to grasp predictive patterns and inline with investors' perception which means that CNN model is:

- Certain parameters and techniques help alleviate overfitting, and CNN model is robust to specification variations
- From Grad-CAM, the CNN can deal with the 2-D image like human
- Successful to extract predictive features and predict future return

8.2 Future Direction

Except the above property, we are also interested in transferability and uniqueness of CNN model:

- Whether the prediction of CNN is similar to other factors or models, which means they bear the similar risk?
- Whether the CNN trained from certain dataset can be transfer to another dataset?
- Whether the portfolio constructed according to the signal from CNN model can perform over other machine learning model, like LSTM?
- Whether it is possible to analyze the outcome of the last CNN block so that we may treat them as inputs to a new neutral network.

References

Jiang, J., Kelly, B. T., & Xiu, D. (2020). (re-)imag(in)ing price trends. *Chicago Booth Research Paper No. 21-01*. http://dx.doi.org/10.2139/ssrn.3756587