Revisiting (Re-)Imag(in)ing Price Trends

Wentao Yu and Ruoxiao Cao*
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{wyuaq, rcaoah}@connect.ust.hk

Abstract

Predicting future returns based on past price data is of significant practical interest and has been extensively studied in the literature. Traditional methods mostly rely on the philosophy of hypothesis testing. By contrast, one recent work proposes to predict future returns based on machine learning [2]. By encoding the dynamics of past price data into two-dimensional images, the powerful learning capability of convolutional neural networks (CNNs) is leveraged to extract the price patterns. In this project, we first reproduce the main experimental results presented in the original paper. Then, we implement several suggested extensions: i) testing the robustness of the model in predicting the price trends of different time scales; ii) providing Grad-CAM visualizations to interpret the learned model. In addition to paper replication, we propose an *attention*-aided CNN model to further enhance the prediction accuracy. The attention module can guide the CNN to focus on the most important parts in the feature maps. The mechanism behind attention is also validated by Grad-CAM visualization results.

https://hkustconnect-my.sharepoint.com/personal/rcaoah_connect_ust_hk/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Frcaoah%5Fconnect%5Fust%5Fhk%2FDocuments%2FPhD1S%2FMATH5470%2FProject&ga=1

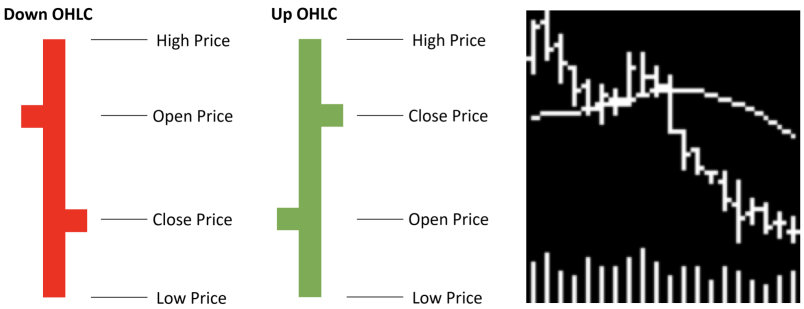
1 Introduction

In the finance literature, enormous efforts have been devoted to the investigation of predicting future returns based on past price trend data. Traditional predictors, such as price momentum and reversal, are mostly established on the the philosophy of hypothesis testing. Recently a focus has been placed on incorporating powerful machine learning algorithms to mine the price data. Specifically, by encoding the price trend data into two-dimensional images, convolutional neural networks (CNNs) can be trained to automatically learn from massive historical price data and then make predictions for future returns [2]. This report details the reproduction of the main results presented in the paper, and also includes several suggested extensions on sensitivity analysis, interpretation (via Grad-CAM visualization), robustness analysis (for different time scales), etc. The reproduced simulation results well match the ones reported in the original paper.

In addition to paper replication, we seek to further improve the prediction accuracy of the CNN, because even a slight increase in accuracy would result in huge profits in real markets [2]. Since the price trend image contains a mixture of different sources of information, our main idea is to incorporate *attention* modules into the network to guide the model to focus on more meaningful parts of the feature maps. Specifically, inspired by [4], we add two sequential attention modules, i.e., one channel attention and one spatial attention. A faster convergence speed (during training) and an improved prediction accuracy (during testing) is observed in our experiments thanks to the attention module. Such an additional module is extremely lightweight and costs negligible training overhead.

*Both authors contributed equally to this project during the whole process of coding, report writing and presentation preparation. Codes to reproduce the results and the video presentation are available at this link.

Course Project, MATH 5470 Statistical Machine Learning (2022 Spring), HKUST.



(a) (b)

Figure 1: (a) An example of the OHLC chart, in which the open and close prices can be easily identified by the directions of the bars. (b) One instance of the price trend image in the dataset. From top to bottom are the OHLC chart, the moving average line, and the daily trading volume bars.

The remainder of this report is organized as follows. In Section 2, we first introduce the background, including the problem statement and the dataset of price trend images. In Section 3, we detail on paper replication, including various suggested extensions. In Section 4, we propose an attention-aided CNN model to further improve the prediction accuracy. We conclude the report in Section 5.

2 Background

2.1 Problem statement

We seek to predict future returns based on past price data by using machine learning. A dataset of historical price trend data is collected beforehand and transformed into the form two-dimensional images so that the powerful CNN models can be utilized. During training, the input of the CNN is a price trend image over a constant period (e.g., 20 days), while the corresponding label is whether the price trend will increase or decrease over the next few weeks (e.g., 20 days). The CNN model is trained purely on historical price trend image data (i.e., imaging) while making predictions for the future (i.e., imagining), which is the reason behind the term '(re-)imag(in)ing' in the title. In essence, this is a binary image classification problem that can be solved by CNN pipelines.

2.2 The dataset of price trend images

Previous studies mostly represented the price trend as one-dimensional time series. In the replicated paper, to exploit the learning capability of CNNs, the time series data were embedded in a higher dimensional space, i.e., represented as two-dimensional images [2]. The reasons in doing so is two-fold. On the one hand, image classification have been extensively studied in the computer vision literature. There are many off-the-shelf methods that can be readily adapted and applied. On the other hand, image representation of price trends can convey more relational attributes of the data that can hardly be spotted in time series representations.

The information in the price trend image contains both price and volume data, including daily open, close, high, and low prices, daily trading volume, and moving average price, for a constant time period. The data price trends are represented by OHLC charts, as shown in Fig. (1a). Since the the the open and close prices can be easily identified by the directions of the bars, transforming the chart into a grey-scale image will not affect the embedded information. This is favorable as CNNs for gray-scale images are generally easier and more economical to train compared to the ones for color images. The whole price trend image of 20 days is shown in Fig. (1b). In addition to OHLC charts, the relative smooth curve in the middle represents the moving average price with a window size of 20 days. Finally, the bars in the bottom of the image stand for the daily trading volume. The size of 20-day price trend image is truncated into 64×60 pixels, where the two price data curves take $\frac{4}{5}$ of the height while the volume bars take $\frac{1}{5}$ of the height. The whole image is in gray scale with pixel values spanning from 0 to 255. The background are set as black while the curves are set as white.

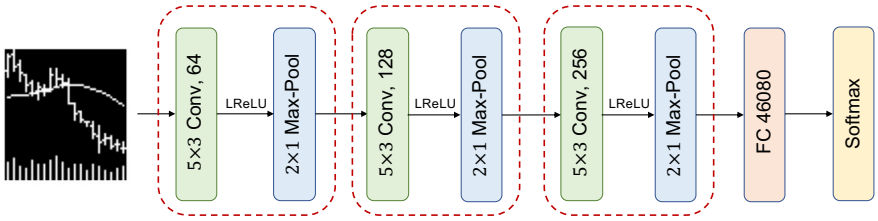


Figure 2: Overall architecture of the baseline CNN for 20-day price trend images in the paper [2].

The corresponding labels contain the returns for the next 5, 20, and 60 days following the last day in the image. The training and validation dataset comprises price data from year 1993 to 2000, while the test dataset contains price data from year 2001 to 2019. Therefore, the task is as of imagining the future price trends based on the knowledge of historical data. The trained model is of practical relevance as for designing trading strategies in the finance market. In the following, we denote the task of predicting the future returns (positive or negative) for the next 20 days based on the price image of past 20 days as the 'I20R20' task. Other abbreviations are defined accordingly.

3 Paper replication

3.1 CNN architecture and training details

We first consider the I20R20 task and reimplement the baseline CNN model described in the paper. The overall architecture of the network is shown in Fig. (2). A total of three convolution (Conv) layers with 5×3 kernels are used, generating 64, 128, and 256 feature maps, respectively. Each Conv layer is followed by an leaky ReLU (LReLU) activation function and one 2×1 max pooling layer. After the last Conv layer, the feature map is flattened into a vector and passed through a fully-connected (FC) layer with softmax activation function. The FC layer has a dropout rate of 50% to avoid overfitting.

Since the goal is predicting the future returns (positive or negative), it is a binary classification problem. In particular, the label is denoted as $y = 1$ if the return in the next 20 days is positive, and $y = 0$ otherwise. Then the cross entropy loss function for the problem is given by

$$\ell_{\text{CE}}(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}), \tag{1}$$

where $\hat{y}$ is the predicted label of the CNN. Based on the loss function, we use an Adam optimizer with a constant learning rate of 1×10^{-5} to train the network. To make the training process converge faster, we initialized the weights by Xavier initialization, and use batch normalization in each layer. We set the early stopping patience as 2, i.e., the training will be stopped if the validation accuracy does not increase for two consecutive epochs.

The prediction accuracy is tested on an out-of-sample test dataset collected in the future years of the training dataset. We say that a prediction is true positive (TP) if both the ground-truth and predicted returns are positive, and false positive (FP) if the ground-truth return is negative but the predicted return is positive. True negative (TN) and false negative (FN) are defined in a similar manner. Two performance criterion are adopted to evaluate the prediction performance, i.e., the accuracy-score and f1-score. The accuracy score is given by

$$\text{accuracy-score} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{2}$$

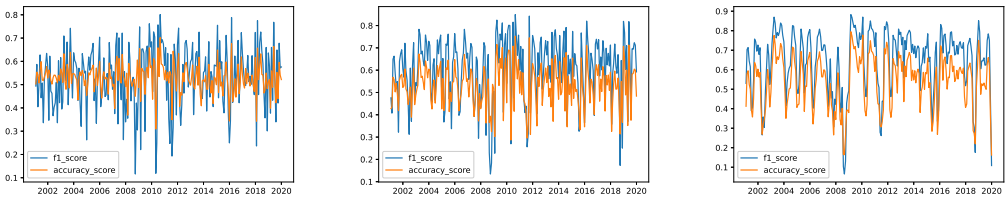
The f1-score is defined as

$$\text{f1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{3}$$

where precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that were retrieved, respectively given by

	accuracy-score	f1-score
I20R5 task	0.5290	0.5570
I20R20 task	0.5254	0.5368
I20R60 task	0.5376	0.6656

Table 1: Performance of the baseline CNN over the whole dataset.



(a) (b) (c)

Figure 3: Performance of the baseline CNN on the (a) I20R5 task, (b) I20R20 task, (c) I20R60 task, respectively, over each month. Each data point in the figure represents one month.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{4}$$

The f1-score is actually the harmonic mean of the precision and recall.

3.2 Experimental results

Following the original paper, we train the baseline CNN on the I20R5, I20R20, and I20R60 tasks using data collected from 1993 to 2000, and test on data from 2001 to 2019. The training is carried out on a Nvidia 2080 GPU. Each training epoch takes about 25 minutes to complete.

3.2.1 Performance over the whole dataset

We first test the result over the whole test dataset (of 19 years from 2001 to 2019), as shown in the table. The accuracy-score and f1-score are very similar to those reported in the original paper. Since every 0.5% increase in accuracy will result in a roughly 0.1 increase in annualized Sharpe ratio, the CNN predictor is already able to create huge profits for investors despite the seemingly low accuracy.

3.2.2 Performance in each month

It is worth noting that the table shows the accuracy over 19 years. To put it another way, it is the expected result of an investor who keeps relying on such a CNN predictor for 19 years. This is not very realistic in the practical sense, since investors are often short-sighted and may give up the predictor if they find it inaccurate. Therefore, it will be interesting to evaluate the performance of the model for a shorter period, e.g., one month. To this end, we resort to the pandas package to divide the dataset by months, and then plot the curve of prediction accuracy-score and f1-score versus time, as shown in Fig. (3). Each data point in the figure corresponds to the result in one month. As can be seen in the figure, the prediction performance keeps fluctuating. Therefore, from our personal perspective, it remains questionable whether an investor would stick to the predictor after seeing the unstable performance. However, from an academic point of view, these results are still promising.

3.2.3 Robustness to different time scales

To test the robustness to different time scales, in addition to the I20R20 task, we also test the capacity of CNN models to predict returns for shorter (5 days, I20R5 task) and longer (60 days, I20R60

Figure 4: Grad-CAM++ visualization of the baseline CNN for the L20R20 task.

task) time scales based on historical data. For the ease of comparison, the results have already been presented before in the table above and in Fig. (3). As can be seen, the performance is stable for different time scales, which illustrates the robustness of the CNN model.

3.2.4 Interpretation via Grad-CAM visualization

Despite the experimental success, interpretation what the CNN has learned from the price data is also very important. However, such interpretation is challenging from a theoretical perspective. One empirical interpretation named improved gradient-weighted class activation mapping (Grad-CAM++) [3, 1] was proposed to provide illustrative ideas of the internal mechanism of CNNs. For each class ("up" or "down" returns), Grad-CAM++ produces heatmaps for each layer of the CNN that illustrate the regions of the input that are most important for predicting a class. The red regions are more focused, while the blue regions are less focused.

In Fig. (4), we plot the Grad-CAM++ image heatmap for each layer of the baseline CNN for the I20R20 task. From left to right, the figures are the original price trend image, the heatmap after the first Conv layer, the heatmap after the second Conv layer, the heatmap after the third Conv layer, the heatmap after the max pooling layer in the third Conv block. It can be seen that in the first Conv layer, the model focuses more on the price trends. From the second Conv layer onwards, we generally cannot observe very clear semantic information. The layers seem to focus on a rather wide region, instead of some particular spots. Such phenomenon seems reasonable given the fact that the classification accuracy (both reported in the original paper and according to our own replication) is only slightly higher than 50%, i.e., only a little bit better than guessing. The baseline CNN cannot spot very meaningful information for classification in the price trend image. This agrees with our personal opinion that it may be too ideal to expect that a simple CNN model can predict the market from limited price trend information (although we admit the academic contribution of the paper).

4 Further improvement: the *attention* module

Since the baseline CNN struggles to spot meaningful information in the price trend images according to our Grad-CAM++ visualization, we are interested in further improving the performance by adding *attention* modules to the CNN [4], hoping that it can focus on more important parts of the images/feature maps. The attention modules has almost been a default choice in model computer vision pipelines due to its lightweight nature and powerful performance. We next introduce the spatial-channel attention module that we implement, provide relevant experimental results, and give Grad-CAM visualization to interpret the result.

4.1 Sequential spatial-channel attention modules

The components of the spatial-channel attention module is similar to the famous convolutional block attention module (CBAM) [4], including one channel attention module and one spatial module, as illustrated in Fig. (5). The channel attention module seeks to exploit the inter-channel relationship, which passes the max-pooled and avg-pooled representation of the intermediate feature map to a shared multi-layer perceptron to generate channel attention weights $\mathbf{M}_c$. Similarly, the spatial attention module concatenates the pooled representations and pass them to a Conv layer to generate spatial attention weights $\mathbf{M}_s$. The channel and spatial attention modules are applied sequentially. Let the input feature map be $\mathbf{F}$, the refined feature map after the channel module be $\mathbf{F}'$, and the refined feature map after the spatial module be $\mathbf{F}''$. We have the following relationship [4]

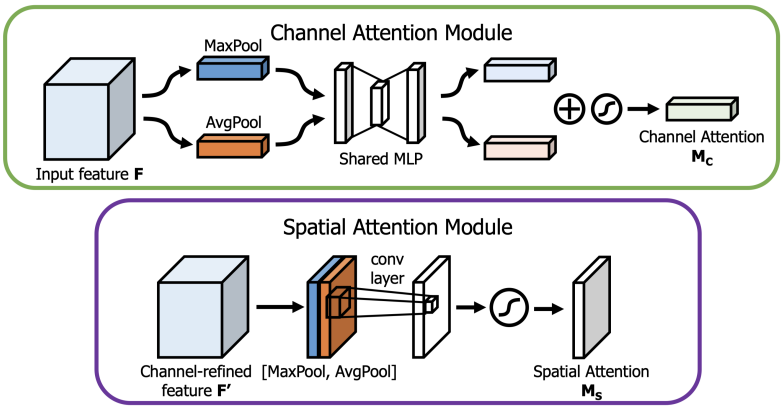


Figure 5: An illustration of the spatial-channel attention module. The source of this figure is [4].

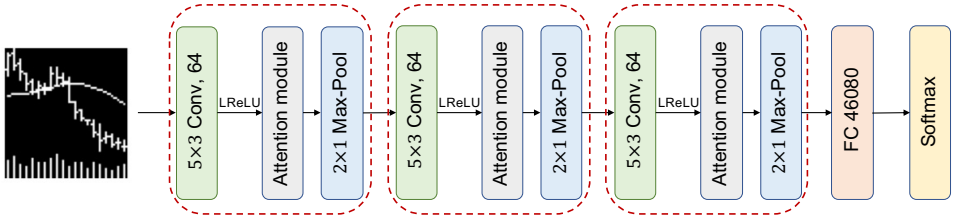


Figure 6: Overall architecture of our proposed attention-aided CNN for 20-day price trend images.

$$\begin{aligned} \mathbf{F}' &= \mathbf{M}_c(\mathbf{F}) \otimes \mathbf{F} \\ \mathbf{F}'' &= \mathbf{M}_s(\mathbf{F}') \otimes \mathbf{F}' \end{aligned} \tag{5}$$

where $\otimes$ here denotes element-wise product. The spatial-channel attention is in essence a weighted refinement of the feature map, whose weights are learned end-to-end.

The attention module is appended right after each Conv layer to refine the feature maps, as shown in Fig. (6). Different from CBAM, we do not use the residual link so as to fairly compare with the baseline CNN architecture in the original paper.

4.2 Performance gain of the attention modules

We train the attention-aided CNN under exactly the same setting as the baseline one. The prediction performance over the whole test dataset is shown on Table (2). As can be seen, the attention-aided module can slightly improve the accuracy-score and f1-score. Note again that every 0.5% (i.e., 0.005) increase in accuracy will result in a roughly 0.1 increase in annualized Sharpe ratio, which means huge returns [2]. Therefore, the performance gain brought by attention is of practical relevance.

The prediction performance over each month is plotted in Fig. (7). In this figure, the difference between the baseline CNN and the attention-aided one is subtle and can hardly be identified by eyes.

4.3 Interpretation via Grad-CAM visualization

We next illustrate what has been learned by the additional attention module, as shown in Fig. (8). Compared with the heatmap of the baseline CNN, i.e., Fig. (4), we notice that the focused region (red regions) are much smaller, which means the network can selectively pay attention to the more meaningful parts of the feature map. Particularly, in the final heatmap, the network focuses on a very narrow region, i.e., the most recent price and volume data, which can partially explain the mechanism behind the attention module.

	accuracy-score	f1-score
I20R20 task (baseline CNN)	0.5254	0.5368
I20R20 task (attention-aided CNN)	0.5290	0.5995

Table 2: Performance of the attention-aided CNN over the whole dataset.

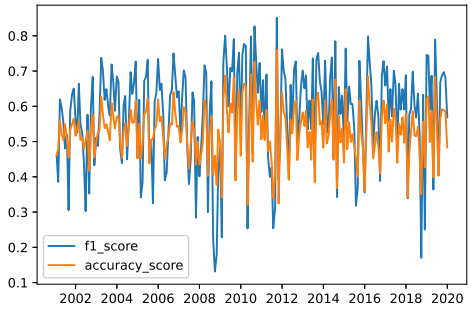


Figure 7: Overall architecture of our proposed attention-aided CNN for 20-day price trend images.

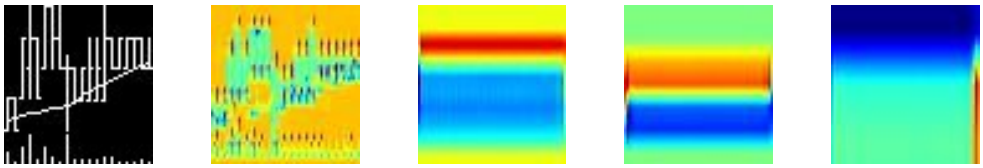


Figure 8: Grad-CAM++ visualization of the attention-aided CNN for the L20R20 task.

5 Conclusion

In this project, we have first replicated the main simulation results of the original paper [2]. Then, we have provided robustness test and Grad-CAM visualization to to interpret the learned CNN model. Additionally, we have also proposed an improved CNN model based on the powerful attention modules. Grad-CAM heatmaps are also provided to supplement the empirical performance gains.

References

[1] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[2] Jingwen Jiang, Bryan T. Kelly, and Dacheng Xiu. (Re-)imag(in)ing price trends. *Chicago Booth Research Paper*, (21-01), 2020.

[3] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[4] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.