Paper Replication: (Re-)Imag(in)ing Price Trends

Wentao Wang, Yingyue Han, Yuhang Jin
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

This project replicates and extends the seminal work by Jiang, Kelly, and Xiu on employing a convolutional neural network (CNN) to predict stock returns from financial market images. Our study reaffirms their findings by demonstrating CNN's robust ability to extract predictive patterns from stock price charts. We conducted rigorous robustness checks, confirming the CNN's effectiveness. Furthermore, we integrate Gradient-weighted Class Activation Mapping (Grad-CAM) for enhanced interpretability and refine a regression model to forecast Sharpe ratios. The project underlines the robustness of CNN predictions in financial analysis and provides valuable insights into the transferability of machine learning models in finance.

1 Introduction

The predictive capacity of past prices for future returns has been a subject of enduring interest and debate within financial economics. The seminal work of Jiang, Kelly, and Xiu stands as a transformative approach to this inquiry, employing the advanced mechanisms of machine learning to discern patterns predictive of future returns from stock-level price charts [3]. Their pioneering application of convolutional neural network (CNN) represented a departure from the traditional empirical methods, moving beyond pre-specified patterns such as momentum and reversal to discover data-driven trends.

This replication project endeavors to reassess the findings by closely re-examining the efficacy of CNN in capturing these predictive price patterns. As we endeavor to replicate and scrutinize this innovative approach, our project aligns with a broader conversation on the role of advanced computational techniques in financial analysis. By recreating the study's datasets, methodologies, and analyses, we aim to validate the robustness of their findings and offer insights into the transferability of machine learning models in finance.

To expand our comprehension of the versatility and potential applications of the proposed method, we embarked on a tripartite extension study. This encompassed: 1) a robustness test, which assesses the method's performance under varying parameters; 2) the implementation of Gradient-weighted Class Activation Mapping (Grad-CAM), a technique that visualizes the focus areas of CNN; and 3) a regression analysis to examine the predictive relationships and dependencies within the data. These extensions are designed to evaluate the method's efficacy and adaptability across different scenarios and analytical frameworks.

Through this replication, we contribute to the academic discourse surrounding the fusion of machine intelligence and market analysis while providing practical implications for the investment management industry.

2 Data description and preparation

The foundation of our analysis is predicated on the stock-level price charts, which serve as the raw predictor data (Figure 1). These charts, presented as images, encapsulate critical market information such as open, high, low, and close prices and trading volume transcribed into a matrix of pixel values to facilitate the application of CNN.

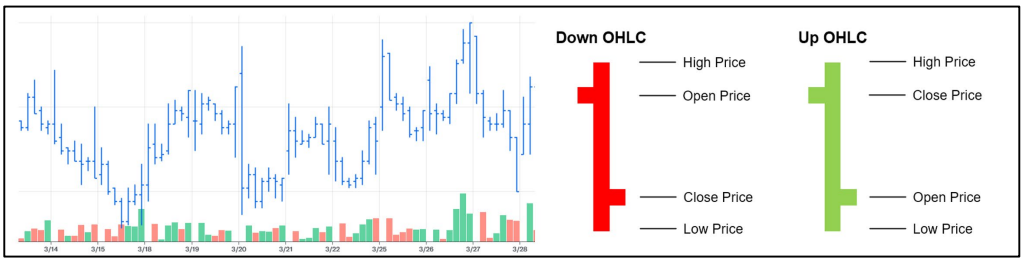

Figure 1: OHLC chart for Xiaomi Corporation stock with daily volume bars.

2.1 Dataset acquisition

Utilizing the Center for Research in Security Prices (CRSP) database, we collated daily stock data from 1993 to 2019 for firms listed on NYSE, AMEX, and NASDAQ, aligning with the original study's temporal scope.

2.2 Image construction

The image covered a specific time frame (20 days). We transformed the historical market data into images. This process involved visually plotting the open, high, low, and close prices alongside trading volume. The imagery representation provides a standardized comparison scale across all assets, crucial for CNN's interpretability and prediction consistency.

2.3 Data labeling

Labels for the images were derived based on whether the subsequent return was positive or non-positive, encapsulating the binary outcome essential for CNN classification. The data was divided into training, validation, and testing sets to enable model learning and subsequent evaluation of predictive performance.

2.4 Computational considerations

Given the resource-intensive nature of CNN training and image analysis, appropriate computational infrastructure was employed to manage the sizable dataset and complex model estimations. We ensured that our analysis incorporated regularized procedures to mitigate overfitting, such as Xavier initialization, as outlined in the original paper.

3 Methodology

Our approach closely mirrors the intricate processes in their original work, focusing on capturing the nuanced predictive associations between historical price patterns and future returns.

3.1 Convolutional neural network

The architecture of our CNN model is a simulacrum of the one used in the primary study, constructed with multiple layers designed to automate feature extraction from input images. As per the original methodology, our input data consists of images representing time-series market data, with each image formed from open, high, low, and close prices, alongside trading volume, transformed into a grid of pixel values. It encapsulates convolutional layers, activation functions, pooling, and fully connected layers (Figure 2).

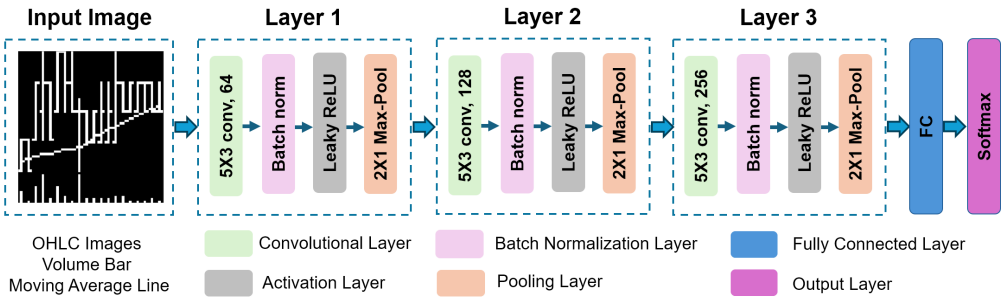

Figure 2: Overview of baseline model of CNN-based return trend classifier.

Convolutional layers perform the crucial task of identifying patterns within the images by applying various filters, capturing both local and global features. We use non-linear activation functions, specifically Leaky Rectified Linear Unit (LReLU), to introduce non-linear properties into the network, allowing for complex pattern recognition. Pooling layers reduce dimensionality and computational complexity by summarizing the features detected in previous layers, further contributing to the network's robustness to variations within the images. Fully connected layers synthesize the features extracted by convolutional and pooling layers to make final predictions about stock return directions.

3.2 Model training and validation

Consistent with the approach of the original paper, we split our dataset into distinct training, validation, and testing sets, with the training and validation phases leveraging data from 1993 to 2000. The CNN is trained using the backpropagation algorithm, with gradient descent optimizations to iteratively adjust weights and minimize predictive error measured by a loss function. Through the validation set, we assess model performance and fine-tune hyperparameters to enhance the predictive accuracy, ensuring that the model neither underfits nor overfits the data.

3.3 Predictive modeling and output

Our predictive model is structured as a binary classification task, identifying the likelihood of positive future returns based on the historical price patterns captured in the images. The output layer employs a softmax activation function to provide probabilistic predictions, resulting in an estimate of the probability of positive stock returns.

3.4 Interpretability and economic relevance

We use a visualization method, Gradient-weighted Class Activation Mapping (Grad-CAM), to understand how different image examples activate the regions of the CNN to trigger up or down return predictions (Figure 3) [9].

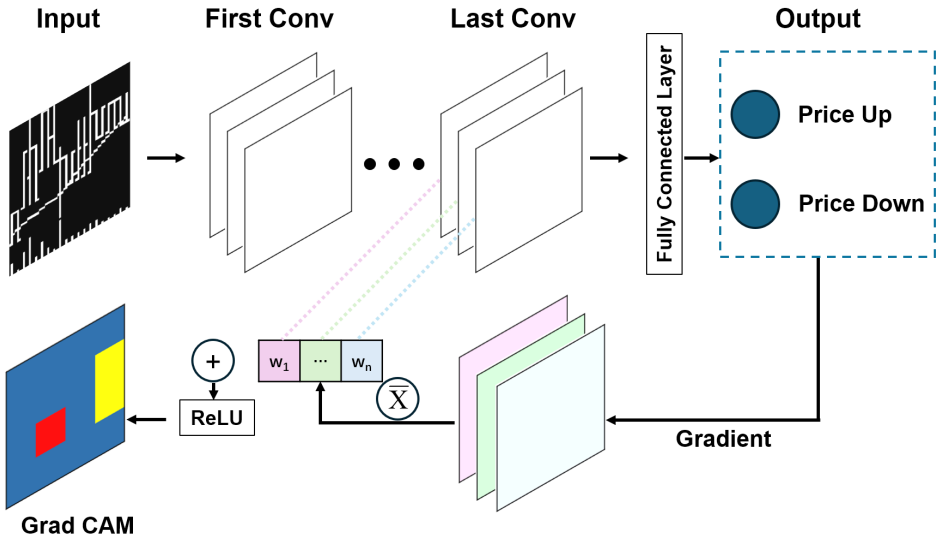

Figure 3: A schematic representation of the Grad CAM method adapted for CNN.

4 Empirical results

4.1 CNN model predictions

Our replication confirms the primary study's assertion that CNN models can predict future returns with considerable accuracy. Training and validation losses among three predicting tasks, including the return trends of 5-day, 20-day, and 60-day, are shown in Figure 4.

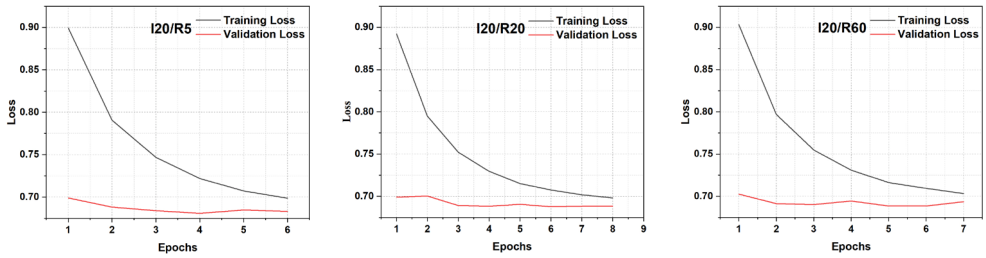

Figure 4: Training and validation losses among three predicting tasks, including the 5-day, 20-day, and 60-day return trends.

When employed as signals in various asset pricing analyses, the out-of-sample predictions provided by CNN displayed robust predictive power. The empirical analysis reaffirmed that the CNN model, trained on images depicting past prices and trading volume, successfully distinguished between stocks with future return prediction (Table 1).

Table 1: Classification Accuracy, F1-score, Recall, ROC AUC, and Sharp Ratio on Test Set.

Return Horizon	Accuracy	F1-score	Recall	ROC AUC	Sharp Ratio
05-Day	0.531	0.585	0.649	0.503	1.278
20-Day	0.518	0.471	0.411	0.485	0.797
60-Day	0.535	0.603	0.654	0.508	1.309

5 Extensions

5.1 Robustness test

A battery of variations was conducted to test the robustness of the CNN model's predictions. These tests corroborated the model's stability and capacity to identify predictive patterns consistently. In Table 2, we replicate and assess the performance sensitivity to various aspects of model structure and estimation decisions. We evaluate sensitivity based on several metrics: the value of the loss function, classification accuracy, the average cross-sectional correlation between forecasted and actual returns (including both Pearson and Spearman rank correlations), and the annualized Sharpe ratios for long-short decile strategies, as documented in prior research. The results indicate that performance remains unaffected mainly by minor adjustments to critical parameters, demonstrating the robustness of our replication models.

Table 2: Sensitivity to model structure and estimation.

	Variation	Accuracy	F1-score	Recall	ROC	Sharp Ratio
Baseline		0.518	0.471	0.411	0.485	0.797
Filter (64)	32	0.527	0.548	0.549	0.498	1.050
	128	0.528	0.578	0.621	0.504	1.220
Layer (3)	2	0.522	0.521	0.499	0.494	0.955
	4	0.528	0.584	0.634	0.505	1.255
Dropout (0.50)	0.00	0.502	0.380	0.293	0.486	0.626
	0.25	0.524	0.575	0.617	0.501	1.219
	0.75	0.532	0.581	0.623	0.503	1.215
BN (yes)	no	0.523	0.527	0.509	0.493	0.972
Xavier (yes)	no	0.530	0.577	0.613	0.504	1.195
Activation (LReLU)	ReLU	0.525	0.553	0.563	0.496	1.087
Max-pool Size (2×1)	(2×2)	0.525	0.595	0.669	0.508	1.366
Filter Size (5×3)	(3×3)	0.521	0.506	0.471	0.491	0.901
Dilation/Stride	(2,1)/(1,1)	0.528	0.578	0.621	0.505	1.220

(2,1)/(3,1)					
	(1,1)/(3,1)	0.527	0.582	0.630	0.507	1.249
	(1,1)/(1,1)	0.528	0.560	0.575	0.502	1.105

In a CNN, filters are sets of learnable weights that are convolved (i.e., slid over) the input data to capture spatial hierarchies and features such as edges, shapes, or more complex patterns in the case of deeper layers. The number of filters in a layer dictates the number of features a layer can extract. More filters can increase the model's capacity but also its complexity and computational demand.

The number of layers in a CNN refers to how many times the process of convolution, activation, and pooling is applied. Each layer has the potential to extract higher-level features from the input data. More layers typically mean more abstraction and complexity, with the potential for better performance increased risk of overfitting, and longer training times.

Dropout is a regularization technique used to prevent overfitting in neural networks. During training, a certain fraction of the nodes (set by the dropout rate) is randomly "dropped" or ignored, which forces the network to learn more robust features that are not reliant on any single node.

Batch Normalization (BN) is a technique to address the problem of internal covariate shift, where the distribution of each layer's inputs changes as the parameters of the previous layers change during training [2]. To enhance training efficiency, minimizing this internal covariate shift is crucial. Batch normalization is an effective method to manage the variability of predictors across various network segments and datasets (Figure 5). This technique is inspired by the internal covariate shift phenomenon, where the input distributions of hidden layers during training differ from those during validation [1]. This challenge is prevalent in training deep neural networks, which typically have numerous parameters and intricate structures. Batch normalization works by normalizing the inputs for each hidden unit in every training step, or "batch" by subtracting the mean and dividing by the standard deviation, thereby maintaining the symbolic integrity of the unit. It can effectively improve the problem of slow network training and unstable training due to distribution changes between different increases.

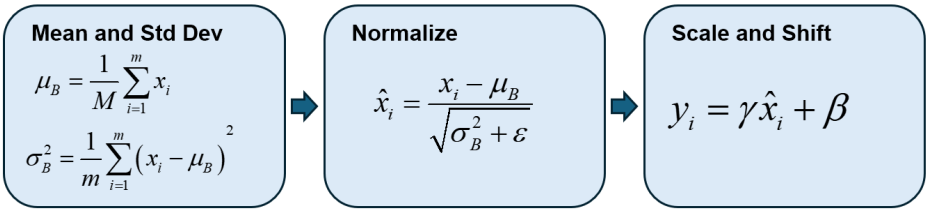

Figure 5: Diagram of Batch Normalization steps.

Xavier Initialization is used to initialize the weights of the network. It aims to maintain the variance of activations and back-propagated gradients throughout the network [4]. If the weights are too small, the signal shrinks as it passes through each layer, and if they are too large, it can blow up—Xavier initialization keeps the signal in a reasonable range.

Activation functions like ReLU or LReLU introduce non-linearities into the CNN, allowing the model to learn more complex patterns (Figure 6) [6]. This non-linearity is crucial for the CNN to make sense of complex data.

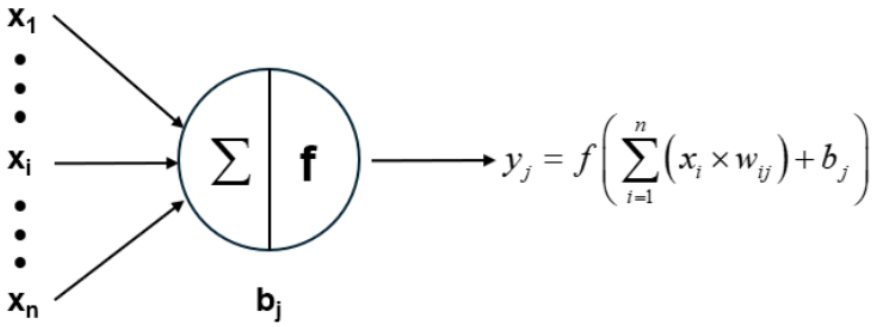

Figure 6: General activation function structure [8].

ReLU sets all negative values in the output to zero, while Leaky ReLU allows small negative values (Figure 7) [7]. The ReLU is an efficient activation function, offering a piecewise linear output that is zero for negative inputs and equals the input for non-negative inputs. This attribute of ReLU accelerates the learning process due to its computational simplicity. Nonetheless, ReLU possesses a drawback where its gradient becomes zero for any negative input. This characteristic means that during back-propagation, errors do not propagate back through the network for these neurons, effectively rendering them inactive, a state often described as 'dying.' To mitigate this issue, variants such as LReLU have been introduced [10]. LReLU aims to sustain neuron activity by allowing a slight, non-zero gradient when the input is negative. Specifically, it outputs a fraction of the negative input instead of zero. This tweak ensures that neurons continue to learn and adapt even when receiving negative inputs, enhancing the robustness of the training process.

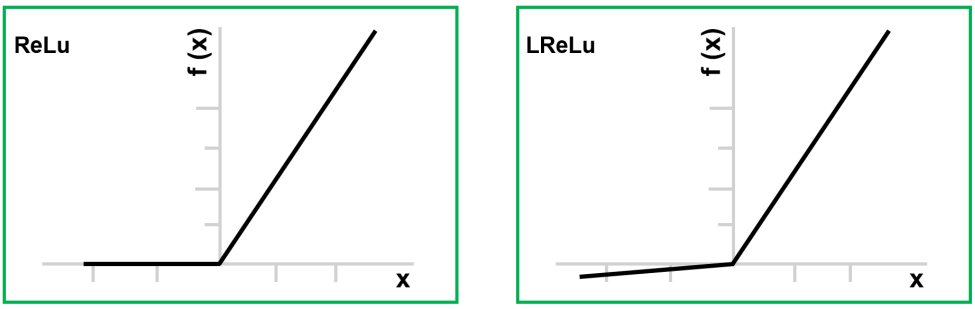

Figure 7: Diagrams of ReLu and LReLU.

A pooling layer capitalizes on the principle of local correlation within images to condense the image size effectively. This process decreases the data volume while preserving essential information. Simultaneously, it curtails the parameter count by eliminating inconsequential features, streamlining the network's complexity without compromising its ability to capture significant patterns. Max-pooling reduces the spatial dimensions of the input volume for the next convolutional layer. It takes the maximum value over a window of a specified size and stride. It uses it to downsample the feature map, reducing the computational load and memory usage while retaining the most critical feature information.

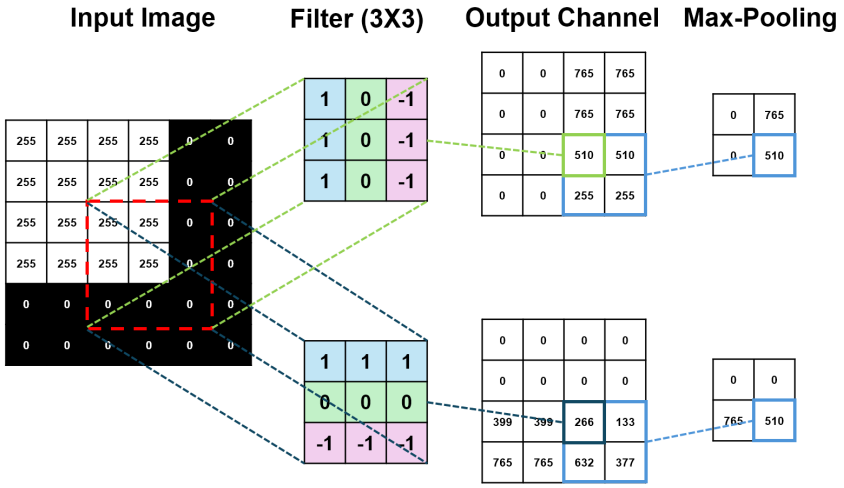

Figure 8: Illustration of convolutional and max-pooling operations.

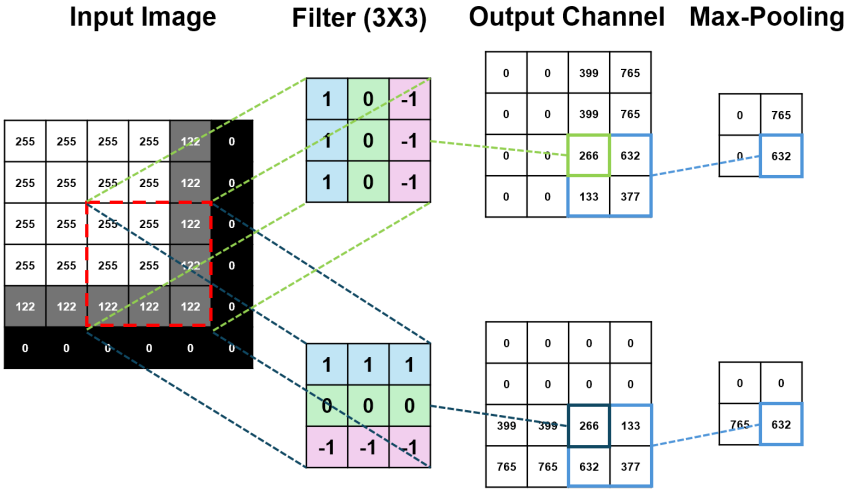

Figure 9: Robustness to Noise from Max-Pooling.

Filter Size refers to the dimensions of the filters used in convolutional layers. Larger filters can capture broader features of the input space, while smaller filters will capture finer, more localized features. The choice of filter size can significantly affect the network's performance and the type of features it can learn.

Dilation refers to the spacing between the values in a kernel (Figure 10) [5]. A dilation of (1,1) means no dilation; the kernel processes every adjacent pixel without skipping. A dilation of (2,2) means the kernel skips one pixel horizontally and vertically between each element it processes.

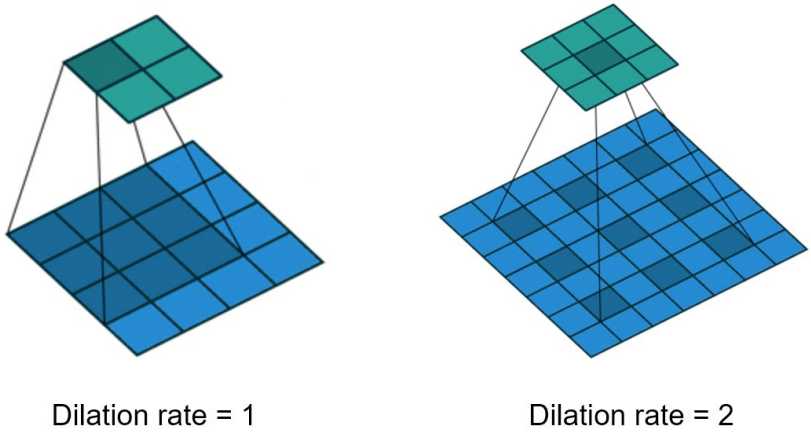

Figure 10: Schematic diagram of Dilation Rate.

The comparison of model performance under different dilations is shown in Table 3. Among the combinations of dilations, the model has the highest accuracy when the dilations of the three convolutional layers are (1,1) (2,2) (3,3), respectively.

Table 3: Comparison of model performance under different dilations.

Dilation	Accuracy	F1-score	Recall	ROC AUC	Sharp Ratio
(1,1) (1,1) (1,1)	0.525	0.542	0.538	0.497	1.028
(2,2) (2,2) (2,2)	0.522	0.542	0.551	0.499	1.065
(1,1) (2,2) (3,3)	0.531	0.630	0.763	0.517	1.700
(1,1) (2,1) (3,1)	0.523	0.542	0.542	0.495	1.042
(2,1) (2,1) (2,1)	0.520	0.481	0.426	0.482	0.821

Figure 11 shows how the dilation rate leads to a “grid effect”. This effect creates a sparser receptive field, potentially overlooking segments of the input space. Such sparsity poses a challenge, as it may yield a model that overlooks critical information contained in the input data.

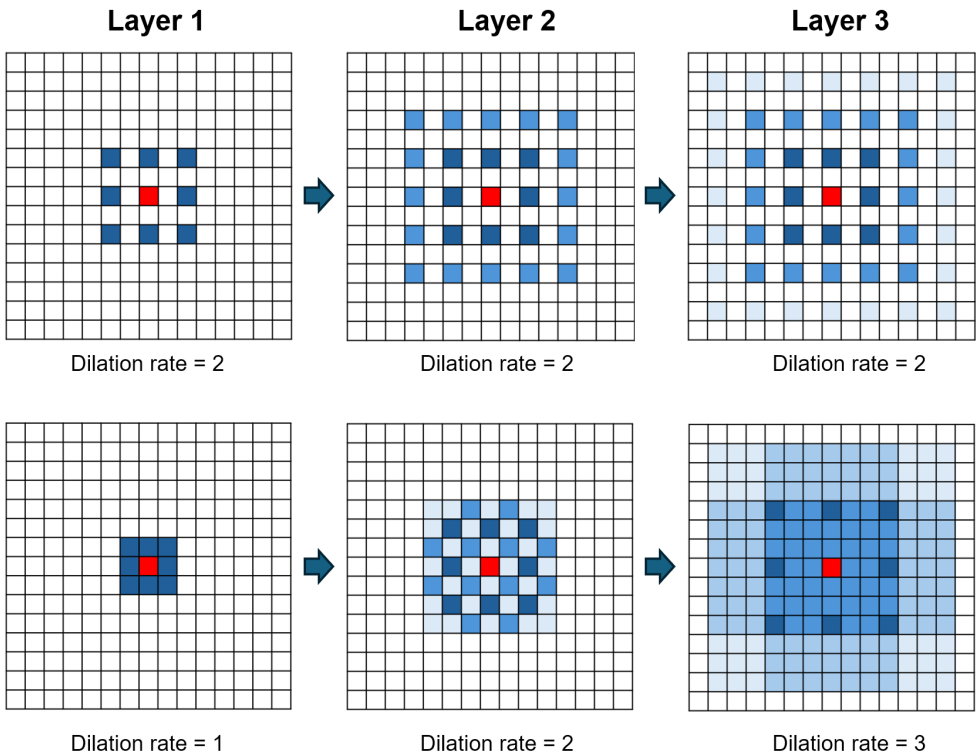

Figure 11: Illustration of gridding effect problem. Left to right: the pixels (marked in blue) contribute to calculating the center pixel (marked in red) through three convolution layers with kernel size 3X3.

Stride is the number of pixels by which we move the filter across the input. For example, a stride of (1,1) moves the filter one pixel to the right and one down (Figure 12). Adjusting stride can change the field of view, and the amount of down-sampling the filter does, affecting which features are captured and the size of the output feature map. Introducing a convolution kernel of a specific dimension inevitably leads to omitting border information. To counteract this, padding is employed, extending the input with zeros to subtly alter its size.

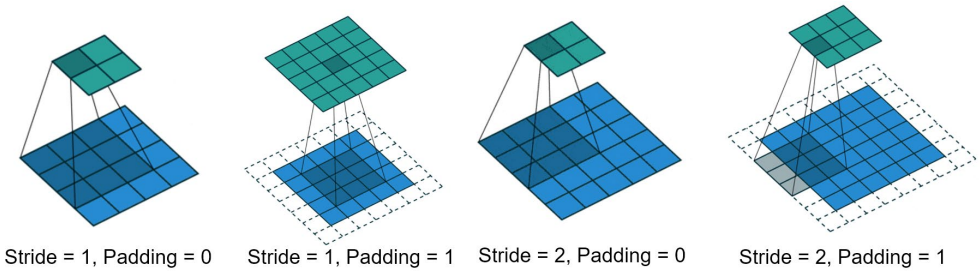

Figure 12: Illustration of Stride and Padding.

5.2 Implementation of Grad-CAM

Grad-CAM is a powerful visualization tool that sheds light on the decision-making process within CNNs. It pinpoints and illuminates critical areas of an input image that significantly sway the model's predictions for categories. The central concept of Grad-CAM lies in harnessing the gradient data coursing through a CNN's final convolutional layers to discern which aspects of the image hold the most weight in each decision. By mapping out the image segments that most affect class scores, Grad-CAM offers a window into the model's analytical process. Grad-CAM reveals the decision-making process inside CNN to determine the key areas that affect the model. The heatmaps of the three convolutional layers visualized by Grad-CAM are shown in Figures 13 and 14. In the first convolutional layer, the areas of interest are relatively small. In the second and third convolutional layers, the areas of interest are larger, and the areas of interest are consistent with the trend. We also found that the moving average line is essential in both "up" and "down" trends. Especially when there are large fluctuations, the moving average line almost dominates. In a down trend, the lowest price receives more attention.

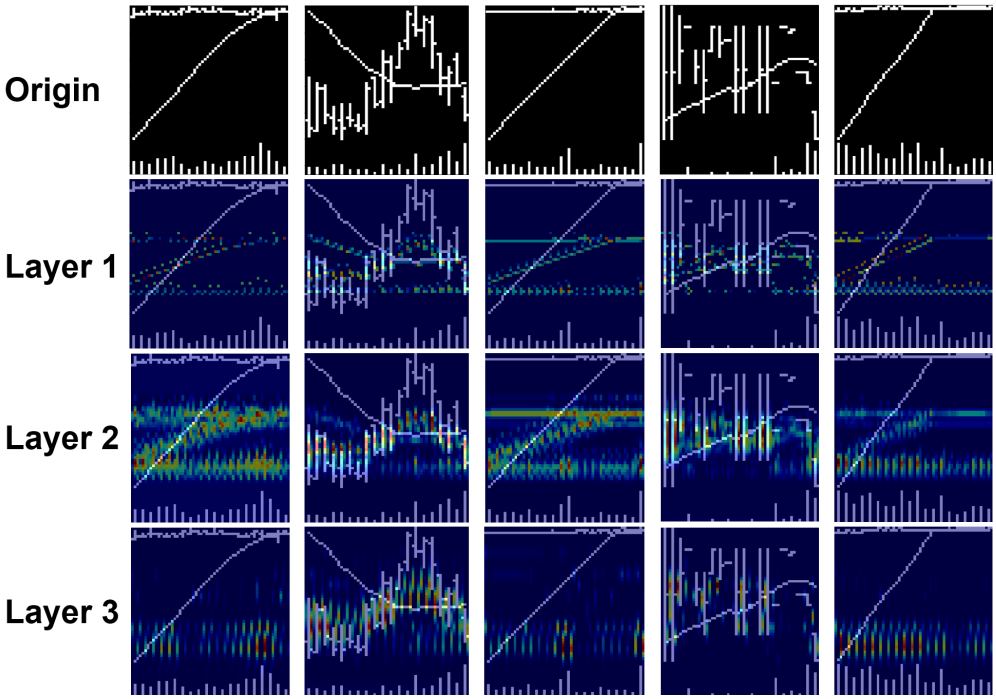

Figure 13: Plots of Grad-CAM for predicting 20-day return horizon for images classified as "up".

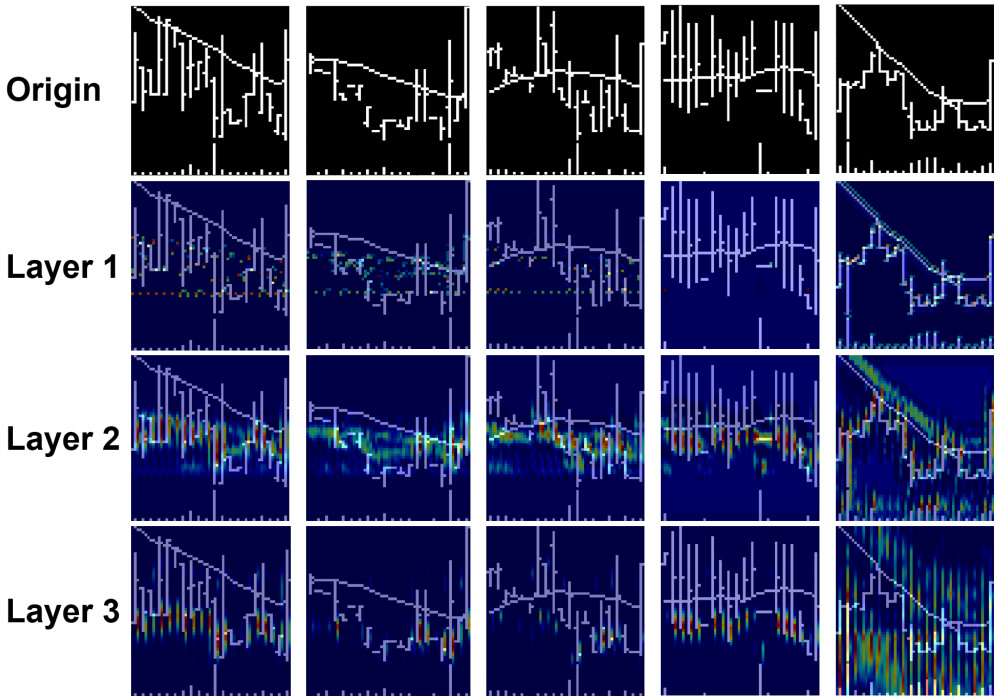

Figure 14: Plots of Grad-CAM for predicting 20-day return horizon for images classified as "down".

5.3 Regression analysis

We refined the regression model by adopting the mean absolute errors as the loss function. The model was calibrated to forecast Sharpe ratios. Given that the candlestick charts adapt to the inherent volatility of a stock, the Sharpe ratios emerge as a more suitable target variable for our analysis. The performance of the regression model is shown in Table 4.

Table 4: Performance of regression model.

	MAE	MAPE	R-Squared	Pearson Correlation	Sharpe Ratio
Baseline	0.097	30.307	-0.018	0.032	0.566

7 Conclusion

The replication project has substantiated the robust predictive ability of CNNs in analyzing stock-level price charts. Our findings reaffirm the original study's conclusions, underscoring the potential of machine learning in financial economics. We've demonstrated the stability of CNN models against parameter variations and highlighted the economic relevance of our visual interpretability technique. This report conducts sensitivity tests on the parameters in the model and focuses on the influence and principles of essential parameters on the prediction effect of the CNN model. Furthermore, we go beyond simple binary classification tasks and train a regression model that can predict not only the trend but also the magnitude of the corresponding change.

Authorship contribution statement

Wentao Wang: Wentao handled the coding and execution of results. For further details, refer to the "codes and results" folder.

Yingyue Han: Yingyue focused on statistical analysis and result interpretation. He evaluated the model's performance and took a role in manuscript preparation, clarifying the methodology and findings.

Yuhang Jin: Yuhang managed the visual representation of data, enhancing the clarity and impact of the presented results. He is responsible for preparing the data for the neural network setup.

References

[1] Gu, S., Kelly, B., and Xiu, D.: 'Empirical asset pricing via machine learning', The Review of Financial Studies, 2020, 33, (5), pp. 2223-2273.

[2] Ioffe, S., and Szegedy, C.: 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', in Editor (Ed.)^(Eds.): 'Book Batch normalization: Accelerating deep network training by reducing internal covariate shift' (pmlr, 2015, edn.), pp. 448-456.

[3] Jiang, J., Kelly, B., and Xiu, D.: '(Re-) Imag (in) ing Price Trends', The Journal of Finance, 2023, 78, (6), pp. 3193-3249.

[4] Kumar, S.K.: 'On weight initialization in deep neural networks', arXiv preprint arXiv:1704.08863, 2017.

[5] Lei, X., Pan, H., and Huang, X.: 'A dilated CNN model for image classification', IEEE Access, 2019, 7, pp. 124087-124095.

[6] Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J.: 'A survey of convolutional neural networks: analysis, applications, and prospects', IEEE transactions on neural networks and learning systems, 2021, 33, (12), pp. 6999-7019.

[7] Nair, V., and Hinton, G.E.: 'Rectified linear units improve restricted boltzmann machines', in Editor (Ed.)^(Eds.): 'Book Rectified linear units improve restricted boltzmann machines' (2010, edn.), pp. 807-814.

[8] Ramachandran, P., Zoph, B., and Le, Q.V.: 'Searching for activation functions', arXiv preprint arXiv:1710.05941, 2017.

[9] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D.: 'Grad-cam: Visual explanations from deep networks via gradient-based localization', in Editor (Ed.)^(Eds.): 'Book Grad-cam: Visual explanations from deep networks via gradient-based localization' (2017, edn.), pp. 618-626.

[10] Xu, B., Wang, N., Chen, T., and Li, M.: 'Empirical evaluation of rectified activations in convolutional network', arXiv preprint arXiv:1505.00853, 2015.