CNN-based Stock Price Trend Image Recognition

Team Members and Contributions
Sun Tiantian: Designed the training pipeline and the implementation of optimization strategies.
Liang Jing: Conducted model interpretability analysis using Grad-CAM to visualize predictive patterns.
Zhang Xinyi: Performed robustness testing through systematic hyperparameter sensitivity analysis.
Zhong Qi: Evaluated model performance and extended binary prediction to return value prediction.
Liu Sihui: Wrote and organized the technical report.

Abstract

This paper revisits stock price prediction from a machine learning perspective by transforming historical price data into two-dimensional images and employing convolutional neural networks (CNNs) to automatically uncover latent predictive patterns from raw data. The proposed method standardizes cross-asset data scales and captures relational attributes difficult to extract with traditional time-series approaches. Empirical results demonstrate the model's ability to generate accurate short- and medium-term return forecasts, which translate into actionable trading signals with robust out-of-sample performance. Through comprehensive backtesting, visualization techniques, and sensitivity analyses, we validate the model's capacity to identify context-independent patterns while also elucidating its limitations in long-term large-cap predictions.Besides, we extend the binary classification to directly predict the detailed return values.This work provides a reproducible, interpretable, and scalable framework for data-driven financial trend recognition.

Github: https://github.com/Aaaaaant97/CNN-based-Stock-Price-Trend-Image-Recognition

1 Introduction

Traditional asset pricing research has predominantly relied on hypothesis testing and often ad hoc predictors discovered through human-intensive statistical learning to investigate price trends (Lo et al.[4]; Sullivan et al.[6]). This paper addresses these limitations by introducing a machine learning framework that transforms historical price and volume data into two-dimensional images and employs convolutional neural networks (CNN) to automatically uncover latent predictive patterns from the raw data (Gu et al.[2]). The methodology standardizes cross-asset data scales and captures relational attributes difficult to extract with traditional time-series methods, moving beyond pre-specified patterns like momentum and reversal (Jiang et al.[3]). Empirically, the model estimates probabilities of positive returns over short-, medium-, and long-term horizons, with predictions forming actionable trading signals that outperform traditional technical indicators such as those examined by Brock et al.[1]. This approach not only validates the model's ability to identify robust, context-independent patterns that generate accurate forecasts and excess returns but also elucidates the mechanisms behind these patterns through visualization techniques like Grad-CAM (Selvaraju et al.[5]).

2 Data Preprocessing and CNN Architecture Integration

2.1 Pipeline Design for Financial Chart Image Representation

This study proposes an innovative architecture that deeply integrates data representation with model design: by transforming one-dimensional financial time series into standardized two-dimensional images and leveraging Convolutional Neural Networks (CNNs) to automatically extract predictive

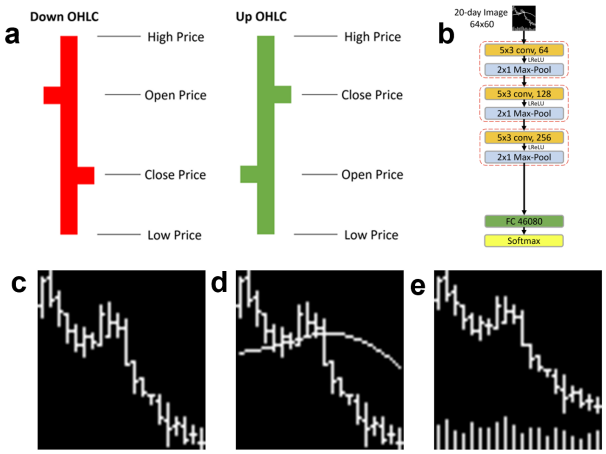

Figure 1: Data preprocessing and CNN architecture integration (a) OHLC chart (b) CNN architecture (c) 20-day images without volume bar and moving average line (d) 20-day images without volume bar but with moving average line (e) 20-day images with volume bar but without moving average line

patterns from these images, it achieves end-to-end intelligent generation of investment signals directly from raw market data. Specifically, based on OHLC price and trading volume data from U.S. stocks spanning 1993 to 2019, the system generates structured images: using more pixel-efficient OHLC bar charts (each trading day occupying only a 3-pixel width), constructing price series with CRSP-adjusted returns and normalizing the first day's closing price to 1, and aligning the highest and lowest prices within the period to the top and bottom boundaries of the image through vertical scaling—placing all stocks on a unified and comparable scale. Additionally, moving average lines and volume bars placed in the bottom one-fifth of the image are integrated. The resulting standardized sparse image matrices, featuring black backgrounds with white lines, balance computational burden with rich multidimensional information. At the model design level, the CNN architecture forms a symbiotic relationship with the image-based data representation, enabling it to automatically learn predictive patterns directly from raw pixels without predefined hypotheses. Its core building blocks—convolutional layers, Leaky ReLU activations, and max-pooling layers—extract features and reduce dimensionality. The network depth adapts to the prediction horizon (2, 3, and 4 convolutional blocks for 5-, 20-, and 60-day forecasts, respectively), with filter numbers doubling in deeper layers to capture more complex patterns. Finally, the output is flattened, processed through a fully connected layer with Softmax, and translated into the probability of a positive return—completing the transformation from raw data to actionable investment signals.

2.2 Baseline CNN Model

This study adopts a systematic training workflow and optimization strategy, implementing a unified training framework across three specific prediction tasks: I20R20, I20R60, and I20R5. Regarding dataset partitioning, we strictly follow time-series principles, designating the 1993–1999 period as the training-validation phase and the 2000–2019 period as the purely out-of-sample testing phase. The dataset sizes for each task are detailed in Table 1.

Table 1: Dataset Partition for Different Prediction Tasks

Task	Training Set	Validation Set	Test Set
I20R20	482,771	206,902	1,490,937
I20R60	475,913	203,964	1,471,336
I20R5	485,125	207,912	1,497,687

Model training centers on the cross-entropy loss function as the core optimization objective, framing stock price trend prediction as a binary classification problem. The mathematical expression is given

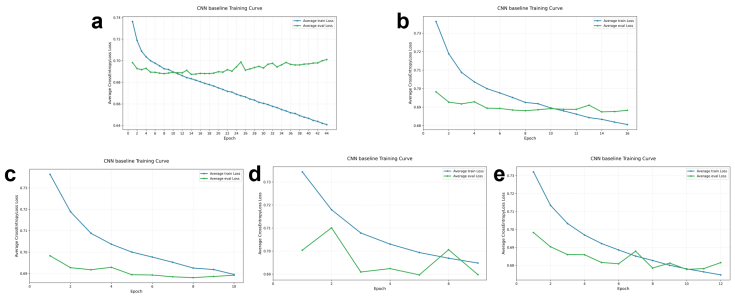

Figure 2: CNN baseline training curve (a) I20R20: CNN baseline without early stopping (b) I20R20:CNN baseline with eralystopping(start epoch = 10) (c) I20R20: CNN baseline with eralystopping(start epoch = 0) (d) I20R60: CNN baseline with eralystopping(start epoch = 0) (e) I20R5: CNN baseline with eralystopping(start epoch = 0)

by:

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

which precisely quantifies the discrepancy between the predicted probability distribution and the true label distribution. In terms of training strategy, we implement multi-level optimization techniques: ensuring ideal scaling of weights across all network layers through Xavier initialization; employing the Adam optimizer for gradient updates with a learning rate of 1×10^{-5}, alongside a tiered processing scheme with batch sizes of 128 for training, 256 for validation, and 2048 for testing; to prevent overfitting, we comprehensively apply Dropout technique with a 50% drop rate (in fully connected layers), batch normalization (inserted between convolutional and activation layers), and a dynamic early stopping mechanism (patience=2). Notably, the early stopping method utilizes two initiation modes—starting monitoring from either epoch 0 or epoch 10—which can avoid interference from early training fluctuations while effectively preventing overfitting. Simultaneously, samples containing NaN values are directly removed during training to ensure data quality.These optimization strategies form an organic whole, demonstrating excellent training stability and generalization performance across all three tasks: I20R20, I20R60, and I20R5.

Table 2: Performance Metrics Across Different CNN Model Configurations

Model	**Epoch**	**Type**	**Confusion Matrix**				**Avg.Acc**	**Corr.**
			TN	FP	FN	TP		
CNN baseline without early stopping (I20R20)	14	best	324,941	391,986	315,490	458,520	52.5%	3.0%
	44	latest	265,196	451,731	259,294	514,716	52.3%	2.3%
CNN baseline with early stopping (start epoch=10) (I20R20)	14	best	324,941	391,986	315,490	458,520	52.5%	3.0%
	16	latest	277,485	439,442	262,673	511,337	52.9%	3.0%
CNN baseline with early stopping (start epoch=0) (I20R20)	8	best	267,309	449,618	251,140	522,870	52.5%	3.0%
	10	latest	484,003	232,924	487,484	286,526	51.6%	3.0%
CNN baseline with early stopping (start epoch=0) (I20R60)	5	best	330,095	353,400	343,172	444,669	52.7%	2.0%
	7	latest	142,293	541,202	135,891	651,950	54.2%	2.1%
CNN baseline with early stopping (start epoch=0) (I20R5)	10	best	330,075	405,394	295,735	466,483	53.1%	5.3%
	12	latest	460,093	275,376	432,908	329,310	52.6%	5.3%

3 Performance Evaluation

3.1 Classification Accuracy

For model performance evaluation, we followed the methodology of the original paper to compute both classification accuracy (Accuracy) and Pearson correlation coefficient (Correlation). Specifically, the accuracy Accuracy_t and correlation Correlation_t are calculated for each cross-section at time t.

The final reported performance metrics are the time-series averages of these cross-sectional values over the entire testing period:

$$\text{Accuracy} = \frac{1}{T}\sum_{t=1}^{T}\text{Accuracy}_t, \qquad \text{Correlation} = \frac{1}{T}\sum_{t=1}^{T}\text{Correlation}_t \tag{2}$$

The evaluation results for the three prediction horizons, based on our optimal model configuration (the CNN baseline model with early stopping, starting monitoring from epoch 0), are summarized in Table 3. Notably, for the I20R60 task, we report the results from the latest model due to its higher accuracy.

Table 3: Performance Comparison: Original Paper vs. Our Replication

Task	**I20R5**		**I20R20**		**I20R60**	
	Acc.	Corr.	Acc.	Corr.	Acc.	Corr.
Original Paper	–	–	53.3%	3.4%	53.2%	2.4%
Our Work	53.1%	5.3%	52.5%	3.0%	54.2%	2.1%

In summary, our replication results are consistent with the original paper at a comparable level. Furthermore, we have successfully extended the evaluation framework to the short-term (5-day) prediction task, which was not reported in the original study.

3.2 Portfolio Performance

This section evaluates the performance of the CNN model by calculating the average annualized return, annualized volatility, Sharpe ratio, and turnover rate for each decile portfolio. Specifically, at each time point, stocks are sorted into ten decile groups (Decile 1 to Decile 10) based on the predicted probability of a positive return (p) generated by the CNN model, where Decile 10 comprises stocks with the highest predicted probability. The rebalancing frequency is matched to the prediction horizon: for the **I20R60** strategy (predicting 60-day returns), given the quarterly prediction window and monthly data frequency, we adopt a quarterly rebalancing strategy, with portfolios fully adjusted every three months and held for three months; for the **I20R20** strategy (predicting 20-day returns), we use a monthly rebalancing strategy, aligning with the data frequency; the **I20R5** strategy (predicting 5-day returns) is omitted from portfolio performance tests because the monthly data frequency would result in most days being out of the market, preventing the construction of a continuously compounded portfolio.

Let $r_{i,t}$ denote the holding period return of decile group i (where $i = 1, \ldots, 10$) at rebalancing period t, calculated as:

$$r_{i,t} = \sum_{j \in D_{i,t}} w_{j,t} \cdot R_{j,t}, \tag{3}$$

where $D_{i,t}$ is the set of stocks in group i at period t, and $R_{j,t}$ is the realized return of stock j over the holding period (i.e., Ret_60d for I20R60 and Ret_20d for I20R20). Two weighting schemes are tested: **Equal Weight (EW)**, where each stock in the group is weighted equally, i.e., $w_{j,t} = 1/N_{i,t}$ with $N_{i,t}$ being the number of stocks in group i at time t; and **Value Weight (VW)**, where stocks are weighted by their market capitalization at portfolio formation, i.e., $w_{j,t} = \text{MarketCap}_{j,t} / \sum_{k \in D_{i,t}} \text{MarketCap}_{k,t}$.

Assuming T rebalancing periods in the backtest, the time-series mean $\bar{r}_i$ and standard deviation $\sigma(r_i)$ of returns are computed for each group. To annualize the metrics, an annualization factor K is introduced (with $K = 12$ for monthly rebalancing and $K = 4$ for quarterly rebalancing):

$$\begin{aligned} \text{Annualized Return}_i &= \bar{r}_i \times K, \\ \text{Annualized Volatility}_i &= \sigma(r_i) \times \sqrt{K}, \\ \text{Sharpe Ratio}_i &= \frac{\text{Annualized Return}_i}{\text{Annualized Volatility}_i}. \end{aligned} \tag{4}$$

Additionally, the turnover rate for the long portfolio (Decile 10) is calculated. Following Gu et al. [2] and Jiang et al. [3], turnover measures the proportion of trading required to adjust the portfolio

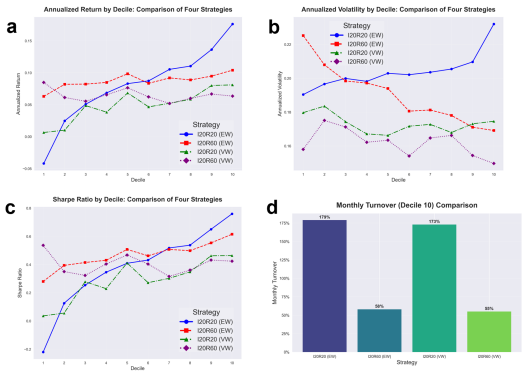

Figure 3: Performance comparison of CNN strategies (a) annualized return comparison (b) annualized volatility comparison (c) sharpe ratio comparison (d) monthly turnover comparison

from the current weights to the target weights at the next rebalancing. To facilitate cross-frequency comparison, all turnover rates are standardized to a monthly equivalent turnover:

$$\text{Turnover} = \frac{1}{M} \cdot \frac{1}{T-1} \sum_{t=1}^{T-1} \left(\sum_{j} |w_{j,t+1}^{\text{target}} - w_{j,t}^{\text{drifted}}| \right), \tag{5}$$

where M is the rebalancing interval in months (i.e., $M = 3$ for quarterly rebalancing in I20R60 and $M = 1$ for monthly rebalancing in I20R20), $w_{j,t+1}^{\text{target}}$ is the target weight of stock j at time $t+1$ as suggested by the model, and $w_{j,t}^{\text{drifted}}$ is the naturally drifted weight of stock j at time t after holding for one period, calculated as:

$$w_{j,t}^{\text{drifted}} = w_{j,t}^{\text{target}} \times \frac{1 + R_{j,t}}{1 + r_{i,t}}. \tag{6}$$

If stock j is not in the portfolio at time t, then $w_{j,t}^{\text{drifted}} = 0$; if it is not in the portfolio at time $t+1$, then $w_{j,t+1}^{\text{target}} = 0$. Based on the constructed portfolios, we evaluate the performance of the CNN model across different prediction windows (20-day and 60-day) and weighting schemes (equal weight and value weight). The backtest results reveal the model's effectiveness in capturing short-term price trends while also exposing its limitations in long-term predictions for large-cap stocks.

For the 20-day horizon (I20R20), the CNN model demonstrates robust short-term predictive ability, with the equal-weight (EW) long-short portfolio achieving a Sharpe ratio of 1.73 and annualized returns showing a clear monotonic increase from Decile 1 (-4%) to Decile 10 (18%). The value-weight (VW) variant retains a positive Sharpe ratio of 0.71, confirming the model's efficacy across market capitalizations. However, performance decays significantly for the 60-day horizon (I20R60): the EW Sharpe ratio drops to 0.44, while the VW strategy fails with a negative Sharpe ratio of -0.28, indicating the model's inability to capture reliable long-term patterns in large-cap stocks. Furthermore, turnover analysis reveals high monthly turnover (179%) for the monthly-rebalanced I20R20 EW strategy, whereas the quarterly-rebalanced I20R60 strategy exhibits a more feasible monthly equivalent turnover of 55%–58%, though its VW performance remains a critical limitation.

4 Robustness Verification and Interpretation of Decision Patterns

4.1 Sensitivity Analysis

This section systematically evaluates the robustness of the CNN model to verify that its performance is not attributable to parameter overfitting. Following the sensitivity analysis framework in Jiang et al., we conduct controlled perturbations of key architectural and training hyperparameters—such as network depth, filter count, and regularization settings—for the I20R20 task. If the model captures genuine market patterns, its performance should remain stable under such variations; significant declines would highlight which components are essential for capturing time-series predictability. For clarity, our baseline model adopts the following hyperparameter settings, consistent with the original paper:

Table 4: Hyperparameter Settings of the Baseline Model

Parameter Category	Hyperparameter Value / Setting
Input	
Image Size (H × W)	64×60
Input Channels	1
Architecture	
Convolutional Layers	3
Filter Size (Kernel)	5×3
Number of Filters	$64 \rightarrow 128 \rightarrow 256$
Pooling Structure	Max-Pool $(2, 1)$
Activation Function	Leaky ReLU ($k = 0.01$)
Batch Normalization	Yes
Training	
Optimizer	Adam ($lr = 1 \times 10^{-5}$)
Batch Size	128
Dropout Rate	0.50
Initialization	Xavier
Loss Function	Cross-Entropy

The table below reports the sensitivity of model performance to hyperparameter variations. For each experiment, we systematically perturb one specific hyperparameter while keeping others fixed at baseline values, including but not limited to: altering the number of convolution kernels (**Filters**) from 64 to 32 or 128; modifying the number of convolutional layers (**Layers**) from 3 to 2 or 4; adjusting the dropout rate (**Dropout**) from 0.50 to 0.00, 0.25, or 0.75; toggling the use of batch normalization (**BN**) and Xavier weight initialization (**Xavier**); changing the activation function (**Activation**) from LeakyReLU to ReLU; varying the max-pooling size from 2×1 to 2×2; testing different convolution kernel sizes (**Filter Size**) of 3×3 or 7×3 instead of 5×3; and exploring alternative dilation and stride configurations (**Dilation/Stride**) such as $(2, 1)/(1, 1)$, $(1, 1)/(3, 1)$, and $(1, 1)/(1, 1)$ in place of the baseline $(2, 1)/(3, 1)$.

Table 5: Sensitivity Analysis of Model Performance to Hyperparameter Variations

Model Structure	Parameter	Loss (V)	Loss (T)	Acc. (V)	Acc. (T)	Corr. (Spearman)	Corr. (Pearson)	Sharpe Ratio (EW)	Sharpe Ratio (VW)
Baseline	-	0.687	0.689	0.542	0.530	0.066	0.044	1.73	0.71
Filters (64)	32	0.689	0.692	0.535	0.523	0.060	0.037	1.31	0.23
	128	0.689	0.695	0.536	0.517	0.053	0.034	1.49	0.61
Layers (3)	2	0.689	0.693	0.535	0.520	0.052	0.031	1.40	0.36
	4	0.688	0.696	0.541	0.517	0.060	0.040	1.58	0.66
Dropout (0.50)	0.00	0.706	0.708	0.525	0.512	0.038	0.025	1.73	0.70
	0.25	0.688	0.695	0.542	0.524	0.059	0.040	1.77	0.71
	0.75	0.688	0.691	0.540	0.526	0.066	0.042	1.56	0.42
BN (yes)	no	0.687	0.691	0.541	0.528	0.063	0.041	1.92	0.57
Xavier (yes)	no	0.686	0.691	0.547	0.528	0.075	0.049	1.99	0.51
Activation (LReLU)	ReLU	0.687	0.691	0.542	0.531	0.064	0.041	1.72	0.67
Max-pool Size (2x1)	(2x2)	0.688	0.692	0.539	0.528	0.068	0.042	1.43	0.41
FilterSize (5x3)	(3x3)	0.687	0.693	0.544	0.524	0.064	0.041	1.62	0.34
	(7x3)	0.689	0.691	0.535	0.531	0.062	0.038	1.46	0.33
Dilation/Stride (2,1)/(3,1)	(2,1)/(1,1)	0.691	0.696	0.535	0.521	0.052	0.028	1.69	0.35
	(1,1)/(3,1)	0.686	0.691	0.545	0.530	0.067	0.045	1.78	0.51
	(1,1)/(1,1)	0.689	0.694	0.538	0.523	0.058	0.036	1.47	0.29

Note: The accuracy and correlation coefficients reported here are calculated by mixing the entire test set samples, rather than using the method of "first calculating cross-sectional metrics for each period and then taking the time-series average."

The robustness analysis of the I20R20 model reveals that its performance remains stable across a wide spectrum of hyperparameter variations, with validation/test losses confined to 0.68–0.71 and

accuracy around 0.51–0.55. This indicates the model captures robust market features rather than fitting parameter-specific noise. Attempts to simplify core architecture (e.g., reducing filters or layers) consistently degrade accuracy and increase loss, validating the baseline design. Dropout proves critical for generalization, as its removal raises validation loss to 0.706. Conversely, the model exhibits tolerance to micro-structural changes (e.g., dilation/stride configurations), confirming that feature extraction does not depend on singular operator choices. Overall, the model demonstrates "macro robustness and micro precision," sustaining predictive power across diverse configurations.

4.2 Grad-CAM

We follow the original paper and employ Gradient-weighted Class Activation Mapping (Grad-CAM) for model interpretability analysis (Selvaraju et al.[5]). This technique identifies the most important regions in the input images for predicting "up" or "down" movements, revealing the price levels, volatility structures, and temporal intervals that the model focuses on across different convolutional layers. The standard Grad-CAM procedure is implemented by first extracting the activation maps (A_k) from a target convolutional layer, then computing the gradients (G_k) of the target class score with respect to A_k, followed by calculating the channel importance weights as:

$$\alpha_k = \frac{1}{HW} \sum_{i,j} G_{k,ij}, \tag{7}$$

and finally generating the class activation map via:

$$\text{CAM}(i,j) = \text{ReLU}\left(\sum_k \alpha_k A_{k,ij} \right), \tag{8}$$

which is then upsampled to the input image size (64×60). Following the original setup, we randomly sample 10 images predicted as "up" and 10 as "down" from the test set and compute their Grad-CAM heatmaps for three convolutional layers. Overall, the results show that the model forms a financially

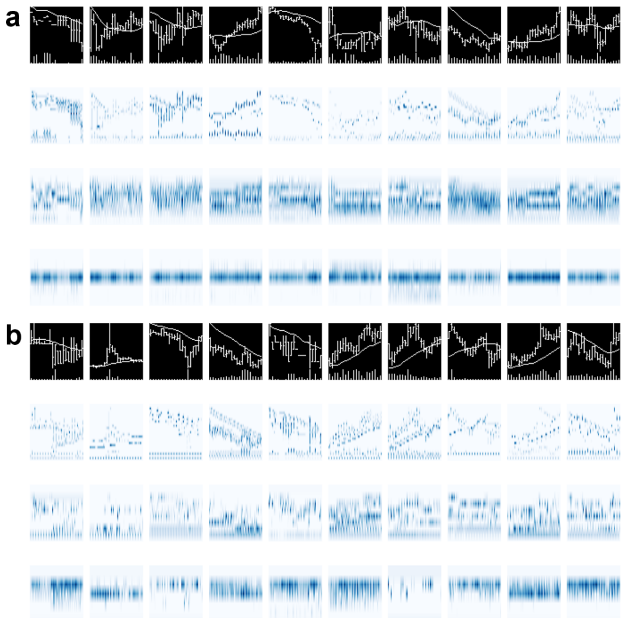

Figure 4: Analyzing hierarchical feature representation and activation patterns (a) UP-classification (b) DOWN-classification

meaningful hierarchical representation across depths: shallow layers activate strongly around local price extremes (e.g., large candles or sharp reversals), extracting short-term structural signals, while deeper layers integrate information across time, focusing on trends, relative opening/closing positions within price bars, and shifts in volatility regimes. Notably, due to stronger vertical down-sampling

in our replicated model, the deepest convolutional layer has a height of 3, resulting in horizontal band-shaped activations. Despite this structural compression, the activations retain a directional bias consistent with the original paper—shifting toward higher price regions for "up" predictions and lower regions for "down" predictions—indicating that the model's final decision relies on the relative price position within the lookback window to form a global judgment on future returns.

5 Prediction Return Values

In this section, we shift the CNN model's prediction target from a simple binary classification (up/down prediction) to the more challenging regression task of predicting specific return values. The technical adjustments are as follows: the convolutional block structure remains identical to the original CNN, while the output dimension of the fully connected layer is changed from 2 (corresponding to the logits for binary classification) to 1, enabling it to output a continuous predicted return value directly. The loss function is also changed from Cross Entropy Loss to Mean Squared Error Loss (MSELoss). We only test the performance for I20R20 return prediction, and the results are summarized in Table 6.

Table 6: Performance of CNN Models on the I20R20 Regression Task

Model	Epoch	MSE	Corr.
CNN baseline without early stopping (I20R20)	best (epoch=15)	2.91%	2.7%
	latest (epoch=101)	3.23%	1.2%
CNN baseline with early stopping (start epoch=20) (I20R20)	best (epoch=15)	2.91%	2.7%
	latest (epoch=30)	2.94%	2.5%
CNN baseline with early stopping (start epoch=0) (I20R20)	best (epoch=15)	2.91%	2.7%
	latest (epoch=18)	2.92%	2.6%

Based on the results, we selected the cnn_baseline with early stopping (starting monitoring from epoch 0) as our final model, as it achieved the lowest MSE with high training efficiency. We subsequently conducted a detailed decile portfolio return test using this model.

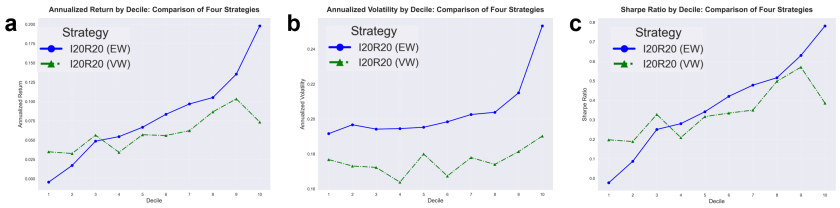

Figure 5: CNN-based framework for predicting return values (a) annualized return comparison (b) annualized volatility comparison (c) sharpe ratio comparison

6 Conclusion

This study establishes a systematic CNN-based framework for stock price trend recognition, demonstrating strong predictive performance in short horizons under equal-weight schemes while also revealing limitations in long-term large-cap predictions under value-weight settings. Through rigorous sensitivity tests and visualization techniques, we verify the robustness and interpretability of the model architecture. Moreover, this approach can be extended beyond binary classification to predict returns over various future horizons, further demonstrating its ability to extract generalizable patterns. Future work may focus on improving long-horizon value-weight performance via transfer learning from short-term models or sample weighting based on market capitalization, and further optimizing the regression-specific CNN structure for direct return prediction tasks to enhance precision and efficiency.

Acknowledgments

We thank our institution for providing computational resources, and acknowledge the CRSP database for the market data used in this study. We are also grateful to colleagues for their valuable input, and to the open-source community (especially PyTorch) for the essential tools that enabled this work. Finally, we thank the authors of the original paper for their inspiring research.

References

[1] William Brock, Josef Lakonishok, and Blake LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *The Journal of finance*, 47(5):1731–1764, 1992.

[2] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273, 2020.

[3] Jingwen Jiang, Bryan Kelly, and Dacheng Xiu. (re-) imag (in) ing price trends. *The Journal of Finance*, 78(6):3193–3249, 2023.

[4] Andrew W Lo, Harry Mamaysky, and Jiang Wang. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The journal of finance*, 55(5):1705–1765, 2000.

[5] 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.

[6] Ryan Sullivan, Allan Timmermann, and Halbert White. Data-snooping, technical trading rule performance, and the bootstrap. *The journal of Finance*, 54(5):1647–1691, 1999.