Channel and Spatial Attention CNN: Predicting Price Trends from Images

Ruixun Zhang[*], Guanglian Lin[†], and Chaoyi Zhao[‡]

July 27, 2022

Abstract

Deep learning has been successfully applied for predicting asset prices using financial time series data. However, image-based deep learning models excel at extracting spatial information from images and their potential in financial applications has not been fully explored. Here we propose a new model—channel and spatial attention convolutional neural network (CS-ACNN)—for price trend prediction that takes arbitrary images constructed from financial time series data as input. The model incorporates attention mechanisms between convolutional layers to focus on specific areas of each image that are the most relevant for price trends. CS-ACNN outperforms benchmarks on exchange-traded funds (ETF) data in terms of both model classification metrics and investment profitability, achieving out-of-sample Sharpe ratios ranging from 1.57 to 3.03 after accounting for transaction costs. In addition, we confirm that the images constructed based on our methodology lead to better performance when compared to models based on traditional time series data. Finally, the model learns visual patterns that are consistent with traditional technical analysis, providing an economic rationale for learned patterns and allowing investors to interpret the model.

Keywords: Price trend prediction; Convolutional neural network (CNN); Attention; Image; Interpretability

[*] Corresponding Author. Peking University School of Mathematical Sciences, Center for Statistical Science, and Laboratory for Mathematical Economics and Quantitative Finance. 5 Yiheyuan Road, Beijing, China 100871 (postal). zhangruixun@pku.edu.cn (email). +86-136-1571-0256 (phone).

[†] College of Cumputer Science, Nankai University. 94 Weijin Road, Tianjin, China 300071 (postal). linguanglian@mail.nankai.edu.cn (email).

[‡] School of Mathematical Sciences, Peking University. 5 Yiheyuan Road, Beijing, China 100871 (postal). zhaochaoyi@pku.edu.cn (email).

Contents

1 Introduction

In recent years, machine learning has been adopted to examine the complex and nonlinear dynamics in asset pricing (Giglio, Kelly, and Xiu, 2021; Nagel, 2021), derivative pricing (Culkin and Das, 2017), volatility forecasting (Vrontos, Galakis, and Vrontos, 2021), index tracking (Shu, Shi, and Tian, 2020), and portfolio selection (Wang and Zhou, 2020; Barroso and Saxena, 2022). In particular, research has shown that machine learning models can be successfully applied to improve return predictability for both stocks (Fischer and Krauss, 2018; Freyberger, Neuhierl, and Weber, 2020; Gu, Kelly, and Xiu, 2020, 2021) and bonds (Bianchi, Büchner, and Tamoni, 2021; Fan, Ke, and Liao, 2021).

A particular class of machine learning models known as deep learning has enjoyed great success across many domains. Although deep learning in asset pricing has gained momentum in recent years,[1] these methods typically use financial time series or panel data as the input. On the other hand, Jiang, Kelly, and Xiu (2022) apply simple convolutional neural networks to predict stock trends using candlestick charts, thus demonstrating that certain types of images may also contain valuable information. Although deep learning is particularly good at extracting spatial information from images, it is as yet unclear whether and how state-of-the-art image-based deep learning models can be applied to model the complex dependence structure behind financial features.

In this paper, we develop a new deep learning model for stock price prediction that uses not only candlestick image data as input, but also time series data that can be converted into images. To utilize image-based deep learning models to explore the spatial relationships within multivariate time series data, two challenges must be addressed. First, we need to convert the financial time series into images without losing any information. Second, because candlestick images and time series-converted images are significantly different from ordinary images such as those of cats and cars, we need to enhance the model's ability to focus on the features relevant to stock prices.

To address these issues, we consider two types of image inputs in our model. The first is an augmented version of the candlestick charts, in which the contour features of each candlestick are enhanced using the image distance transformation algorithm. This process yields four additional images that highlight different regions of the original image. The second is based on the Gramian Angular Field (GAF) (Wang and Oates, 2015)—a bijective encoding method for multivariable time series—which is applied to relevant data including the opening price, high price, low price, closing price, and volume (OHLCV).

[1] Specific neural network architectures in this literature include the fully-connected neural networks (Gu, Kelly, and Xiu, 2020), autoencoders (Gu, Kelly, and Xiu, 2021), and sequence models (Cong et al., 2021a,b).

Taking these images as the input, we propose a novel deep learning architecture by introducing the channel and spatial attention module (CS-Attention) between layers in the convolutional neural network (CNN). This would improve the CNN's flexibility, thus enabling it to capture the most relevant information from the image. For simplicity, we name our model the channel and spatial attention convolutional neural network (CS-ACNN). Here, as in traditional CNNs, the convolution kernel extracts local information from the input image and subsequent layers. On top of that, the CS-Attention mechanism assigns different weights to each channel based on their learned correlation structure, thereby realizing channel-level attention. In addition, the spatial attention module is designed to learn the spatial dependencies of all the input features. Together, the CS-Attention allows for different attention weights along both the channel and spatial dimensions, and the entire network can therefore learn to focus on the most important features.

We test our proposed method empirically using data for three equity index exchange-traded funds (ETF), including the SPDR S&P 500 ETF (SPY) for US stocks, the Hang Seng Index ETF (2833.HK) for Hong Kong stocks, and the China AMC SSE 50 ETF (510050.SS) for Chinese A-shares. For each ETF, we use the direction of movements in daily closing prices as the prediction label. By feeding one of the two types of image data mentioned previously into our CS-ACNN model, we analyze both model performance metrics and profitability metrics of common investment strategies. In addition, we conduct extensive analysis to understand the effect of each component in our framework. Figure 1 summarizes our methodology.

Finally, we develop a methodology based on the gradient-weighted class activation mapping (Grad-CAM) (Selvaraju et al., 2017) to interpret our black-box model and visualize important regions of the input image that are relevant for price trend predictions. We find that our CS-ACNN model successfully learns several patterns that closely match classical technical analysis.

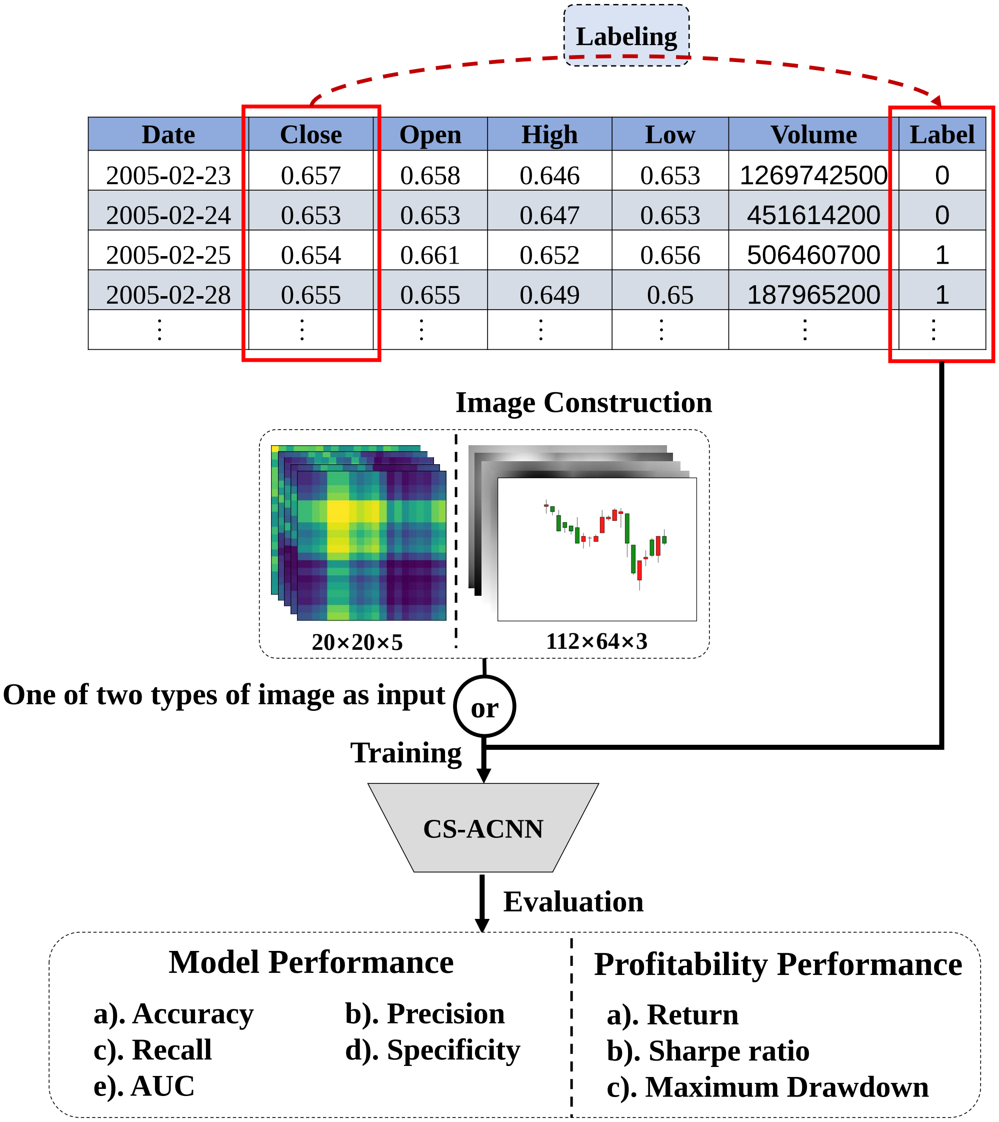
In summary, the primary contributions of our paper include the following:

1. We provide a data augmentation method based on candlestick charts, as well as GAF-based images constructed from financial time series. We empirically demonstrate the effectiveness of these images as inputs to deep learning models, thereby providing a prototype methodology for transforming any financial time series into useful images without losing information.

2. We propose a novel deep learning model (CS-ACNN) that introduces the CS-Attention module between convolutional layers. We demonstrate the superior performance of our model over existing models in terms of both model classification metrics and investment

profitability. The out-of-sample after-cost Sharpe ratios range from 1.57 to 3.03 for different strategies across different markets. This highlights the potential of applying image-based deep learning models to predict stock prices over traditional time series-based models.

3. We demonstrate that it is possible to open our black-box CS-ACNN model by providing a methodology to derive model-learned visual patterns that are interpretable by humans. This shows that our model learns several patterns in a manner that is consistent with traditional technical analysis, thus increasing investors' confidence in the model based on its underlying economic rationale.



Date	Close	Open	High	Low	Volume	Label
2005-02-23	0.657	0.658	0.646	0.653	1269742500	0
2005-02-24	0.653	0.653	0.647	0.653	451614200	0
2005-02-25	0.654	0.661	0.652	0.656	506460700	1
2005-02-28	0.655	0.655	0.649	0.65	187965200	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 1: Illustration of our framework for price trend prediction based on images.

The rest of this paper is organized as follows. We provide a brief review of literature in Section 2. Section 3 describes the image construction methodology and Section 4 presents the CS-ACNN model. We conduct empirical analysis in Section 5 and provide model interpretations in Section 6. Section 7 concludes.

2 Literature Review

There is a growing literature on the application of deep neural networks for stock market predictions. A wide range of architectures have been shown effective for forecasting future price movements, including feedforward neural networks (Chen, Pelger, and Zhu, 2022; Gu, Kelly, and Xiu, 2020), convolutional neural networks (CNN) (Long, Lu, and Cui, 2019; Jerez and Kristjanpoller, 2020; Gao et al., 2022), and recurrent neural networks such as long short-term memory (LSTM) (Fischer and Krauss, 2018; Chen, Wu, and Wu, 2022) and deep regressions (Kim, 2019). These new methodologies are particularly useful in exploring lagged correlation structures (Moews, Herrmann, and Ibikunle, 2019), multi-task learning (Ma and Tan, 2022), adversarial learning (Chen, Pelger, and Zhu, 2022; Koshiyama, Firoozye, and Treleaven, 2021), and factor constructions (Fang et al., 2020; Gu, Kelly, and Xiu, 2021). Giglio, Kelly, and Xiu (2021) and Kumbure et al. (2022) provide two recent reviews. However, the vast majority of these models use financial time series data as inputs rather than images.

In recent years, many researchers have started exploring the use of images for stock price prediction to fully utilize the capabilities of CNNs and other deep learning models in computer vision. CNNs are effective for extracting visual features (Jmour, Zayen, and Abdelkrim, 2018) and can, therefore, detect complex technical patterns in datasets with a low signal-to-noise ratio. They can transform visual data into trading strategies in a way that mimics human perception and decision-making processes (Jiang, Kelly, and Xiu, 2022). Candlestick charts are commonly used visual aids for investors when making trading decisions, and research on stock forecasting based on candlestick charts has achieved good results (Ghoshal and Roberts, 2020; Guo, Hsu, and Hung, 2018).[2] In addition, technical indicators have been transformed into images as the inputs for deep learning models for price trend prediction (Sezer and Ozbayoglu, 2018; Sim, Kim, and Ahn, 2019).

Finally, recent literature has incorporated the attention mechanism to tackle financial problems, which has been proven to be extremely effective in natural language processing (Vaswani et al., 2017). Examples of such models include transformer-based attention

[2] Jiang, Kelly, and Xiu (2022), a primary example in this literature, focus on learning price patterns from candlestick charts for future price trends, while our framework is able to extract information from both candlestick charts and, more broadly, any images constructed from financial time series.

networks (Zhang et al., 2022), LSTM-based attention networks (Chen and Ge, 2019), and attention networks for portfolio construction (Cong et al., 2021a). In addition, Chatigny, Goyenko, and Zhang (2021) use the attention mechanism to identify the most influential time-varying firm characteristics that contribute to the stochastic discount factor in asset pricing.

3 Image Construction

Technical analysis has been adopted in financial practice for several decades. Lo, Mamaysky, and Wang (2000) provide a systematic approach to technical pattern recognition using nonparametric kernel regression. In standard technical analysis, also known as “charting,” price and volume information is used to generate a chart of an asset’s recent history in a given time window, and inferences about future price movements can then be made based on the chart.

We generalize this standard approach by considering two different kinds of images based on the price and volume of an asset, including the augmented candlestick charts and the Gramian Angular Field (GAF). The common goal for these image construction methods is to preserve as much information as possible from the financial time series, while making them suitable for image-based deep learning methods.

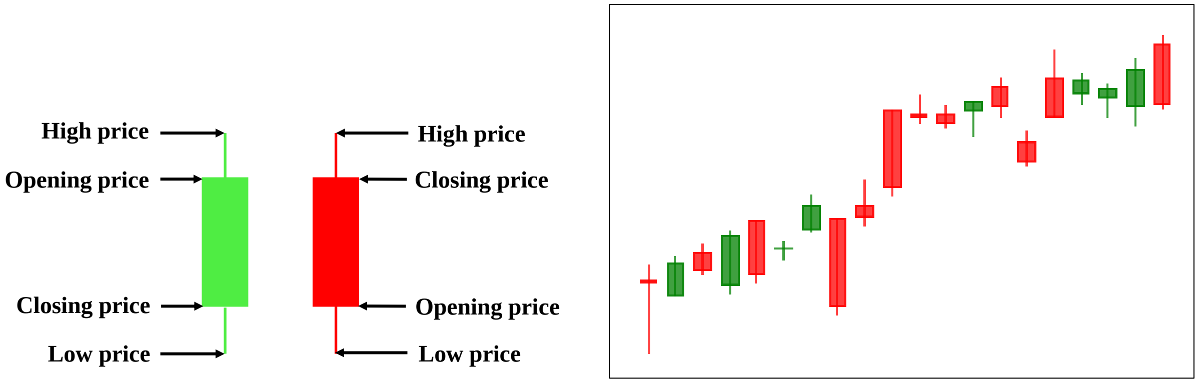
3.1 Augmented Candlestick Charts

A candlestick consists of the opening price, high price, low price, and closing price (OHLC) of an asset in a unit trading period, which, in this study, is one day. Figure 2 shows an example of a candlestick chart.[3] Candlesticks have a wide “body” section that represents the range between the opening and closing prices for the entire day. The high price is marked by the top of the stick, while the low price is marked by the bottom. We follow the convention used in the Chinese stock market where an upward price movement is represented by a red candlestick and a downward price movement is represented by a green candlestick.[4]

Prior attempts to use candlestick charts for forecasting asset prices typically use them directly as the model input (Lee and Jo, 1999; Jiang, Kelly, and Xiu, 2022). However, we develop a data augmentation method for these charts to obtain more training data, which is a common technique used in deep learning to make neural networks more robust.

[3] We use Python’s mpl_finance module, and adopt the convention in China to represent positive trends with red and negative trends with green.

[4] In particular, they are defined by whether the closing price is higher than the opening price of the day.



(a) Examples of candlesticks representing downward and upward price movements, respectively.

(b) A sample candlestick chart for 20 days.

Figure 2: Illustration of a candlestick chart.

It is evident that candlestick charts differ from ordinary image classification datasets. While ordinary images may be affected by the environment or the photographing technology used, candlestick charts are drawn from historical price data. Therefore, it is not necessary to perform data augmentation on them by rotating, clipping, or adding noise.

We first convert candlestick charts into binary images, in which the upward candlesticks are converted to white and downward candlesticks to black. We then enhance their contour features based on the image distance transformation algorithm.[5] In particular, for each pixel p with value $v(p)$ (1 for white and 0 for black), we define an intermediate pixel value $d(p)$ in the transformed image. This value represents the minimum distance between pixel p in the binary candlestick image and all background pixels, p_b, and is given by:

$$d(p) = \begin{cases} \min_{p_b \notin I_f} \operatorname{dis}(p, p_b), & \text{if } p \in I_f, \\ 0, & \text{if } p \notin I_f, \end{cases} \tag{1}$$

where $\operatorname{dis}(\cdot)$ is the Euclidean distance, and I_f represents the set of all foreground (candlestick) pixels. We then normalize the intermediate pixel value $d(p)$ by:

$$p' = \frac{\sqrt{d(p)} - \min(\sqrt{d(p)})}{\max(\sqrt{d(p)}) - \min(\sqrt{d(p)})} \in [0, 1]. \tag{2}$$

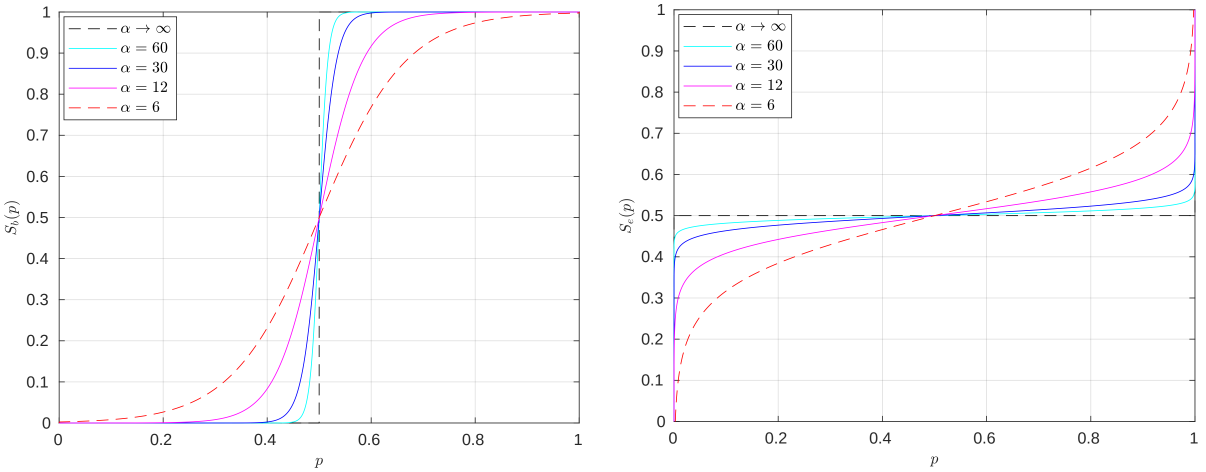
From Eqs.(1)–(2), the pixel value near the center of the candlestick is relatively large, while that near the edge is relatively small. The opposite effect can be achieved by subtracting p' from the original pixel in the binary candlestick image.

[5] See, for example, Borgefors (1986) and Fang et al. (2021).

Following Fang et al. (2021), we smooth the transformed image further by:

$$C_b = v(p) \times p' \times S_b(p'), \tag{3}$$
$$C_b' = v(p) - C_b, \tag{4}$$
$$C_e = v(p) \times p' \times S_e(p'), \tag{5}$$
$$C_e' = v(p) - C_e, \tag{6}$$
$$S_b(p') = \frac{1}{1 + e^{-2\alpha p' + \alpha}}, \tag{7}$$
$$S_e(p') = S_b^{-1}(p'), \tag{8}$$

where S_b and S_e are two smoothing functions parameterized by $\alpha \in [6, +\infty)$.[6] Figure 3 presents these two functions for different values of α and we set $\alpha = 6$ without loss of generality.



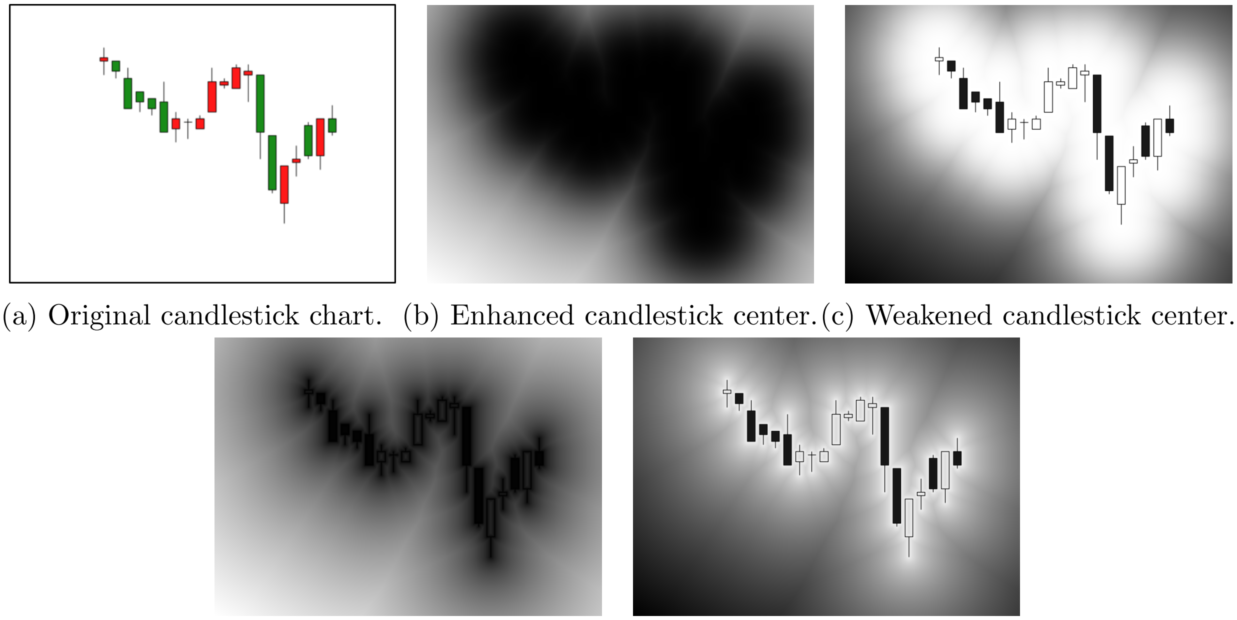
(a) $S_b(p')$ (b) $S_e(p')$

Figure 3: Smoothing functions for augmented candlestick images.

C_b, C_b', C_e, and C_e' represent augmented candlesticks with enhanced and weakened centers and those with enhanced and weakened edges, respectively. Figure 4 demonstrates an example of these augmented images, where the original candlestick chart in Figure 4a is augmented to produce Figures 4b–4e.[7] We feed these augmented images, together with the original image, to the model during training. In other words, each sample of the original candlestick chart is expanded into five images. For model testing, we only use the original image.

[6] $S_e(p')$ goes to $\pm\infty$ when p is very close to 0 or 1. In practice, we clip $S_e(p)$ to be between 0 and 1.

[7] To feed the data into the convolutional neural network, these above images are resized and cropped to 112×64 pixels.



(a) Original candlestick chart. (b) Enhanced candlestick center. (c) Weakened candlestick center.

(d) Enhanced candlestick edges. (e) Weakened candlestick edges.

Figure 4: Examples of augmented candlestick charts using the image transformation algorithm.

3.2 Gramian Angular Field (GAF) for OHLCV

The second method to convert financial time series into images is based on the GAF proposed by Wang and Oates (2015). GAF represents time series in a polar coordinate system instead of the typical Cartesian coordinates, with two main advantages. First, the encoding of GAF is bijective, which guarantees that there will be no information loss in the conversion process. Second, GAF explicitly encodes temporal dependencies of the original time series into the image, thus making it easy for machine learning models to learn.

For each asset, we perform the following steps to convert five time series into five images based on the GAF, including the opening price, high price, low price, closing price, and volume (OHLCV). We use $X = (x_1, x_2, \ldots, x_T)'$ to denote the original time series.

Step 1: We rescale X so that all values fall within the interval $[0, 1]$:

$$\widetilde{x}_t = \frac{x_t - \min(X)}{\max(X) - \min(X)}, \tag{9}$$

where $t = 1, 2, \ldots, T$, and $\widetilde{X}$ denotes the rescaled time series.

Step 2: We represent the rescaled time series $\widetilde{X}$ in polar coordinates by encoding the value as the angle and the timestamp as the radius:

$$\begin{cases} \phi_t = \arccos(\widetilde{x}_t), & \widetilde{x}_t \in [0,1]\,, \\ u_t = \frac{t}{T}, & t \in \{1,2,\ldots,T\}. \end{cases} \tag{10}$$

As a result, (ϕ_t, u_t) defines a time series in the polar coordinate system.

Step 3: We construct a GAF image with $T \times T$ pixels by considering the trigonometric sum between each point in polar coordinates:

$$G = \begin{bmatrix} \cos(\phi_1+\phi_1) & \cdots & \cos(\phi_1+\phi_T) \\ \cos(\phi_2+\phi_1) & \cdots & \cos(\phi_2+\phi_T) \\ \vdots & \ddots & \vdots \\ \cos(\phi_T+\phi_1) & \cdots & \cos(\phi_T+\phi_T) \end{bmatrix} = \widetilde{X} \cdot \widetilde{X}' - \sqrt{I - \widetilde{X}^2} \cdot \sqrt{I - \widetilde{X}^2}', \tag{11}$$

where I is the unit column vector of ones.[8]

Figure 5 shows an example of GAF images constructed using the OHLCV of an asset with $T = 20$. These five images are used as $(20 \times 20 \times 5)$ inputs to the deep learning model that we propose in Section 4.

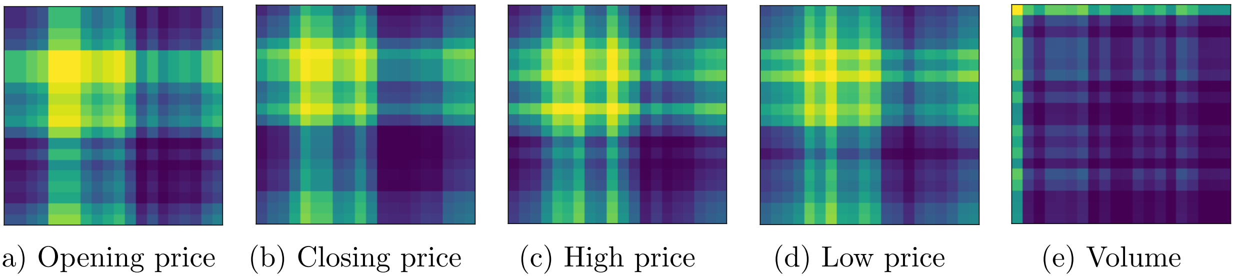

(a) Opening price (b) Closing price (c) High price (d) Low price (e) Volume

Figure 5: An example of Gramian Angular Field (GAF) images constructed from the opening price, high price, low price, closing price, and volume (OHLCV) of an asset with $T = 20$.

These GAF images preserve the original information in the time series X because the mapping in Eq.(11) is bijective. In addition, they provide a way to encode temporal dependencies as time increases and the position moves from the top-left to the bottom-right. In particular, $\{G_{s,t} : |s-t| = L; s,t = 1,2,\ldots,T\}$ represents the dependencies for the time series with lag L. The main diagonal $\{G_{t,t} : t = 1,2,\ldots,T\}$ is the special case for $L = 0$, from which the original time series can be reconstructed.

[8] This is referred to as the Gramian Summation Angular Field (GASF) by Wang and Oates (2015). If we define an inner product as $\langle x,y\rangle = xy - \sqrt{1-x^2} \cdot \sqrt{1-y^2}$, the image G in Eq.(11) constitute a quasi-Gramian matrix under this inner product.

4 Channel and Spatial Attention CNN (CS-ACNN)

Convolutional neural networks (CNNs) have been proven to be effective in extracting structural information from standard images. However, the images that we constructed from the financial time series in Section 3 are significantly different from ordinary images in that they contain important temporal and spatial dependencies. To capture useful information in these images with respect to asset returns, we propose the Channel and Spatial Attention CNN (CS-ACNN). CS-ACNN introduces the channel and spatial attention module (CS-Attention) between the convolutional layers in CNN, which enables the model to generate attention maps along both the channel and spatial dimensions to enhance the input features. This process improves the representation ability of the features that are relevant for asset returns.

On each day t, CS-ACNN takes one of the two classes of the images constructed in Section 3 as input, each containing information for the previous 20 trading days, and predicts the direction of price movement on the next day, or equivalently, the sign of the next daily return. We use z_t to represent the closing price on day t, r_t the rate of return on day t relative to day $t-1$, and y_t the direction of price movement from day t to day $t+1$:

$$r_t = \frac{z_t - z_{t-1}}{z_{t-1}}, \tag{12}$$

$$y_t = \begin{cases} 1, & r_{t+1} \geq 0, \\ 0, & r_{t+1} < 0, \end{cases} \tag{13}$$

where the binary label y_t is used by the model in training. We also refer to $y_t = 1$ as "up" and $y_t = 0$ as "down" for ease of presentation.

4.1 Network Structure

Our CS-ACNN model structure adds the attention mechanism (Vaswani et al., 2017) to deep CNNs. Our starting point is the VggNet, which is a classical deep CNN structure (Simonyan and Zisserman, 2014). The original VggNet contains multiple "convolutional layer → convolutional layer → max-pooling layer" structures, which are followed by three fully connected layers.[9] To mitigate the risk of overfitting given the low signal-to-noise ratio in financial time series, we adopt a simpler structure where each convolutional layer is followed

[9] The number of filters in VggNet (the number of output channels after convolution) starts from 64 and increases exponentially after each max-pooling operation. The convolution mode of VggNet is "same", meaning that the dimension of the output image after convolution is the same as the input, and its downsampling is realized by the max-pooling operation.

by one pooling layer, similar to the classical LeNet (LeCun et al., 1998). The CS-Attention modules are inserted after each max-pooling layer. In addition, the number of convolution kernels for each layer in our CS-ACNN model is set as 32, 48, and 64, respectively.[10]

Figure 6 illustrates the structure of the CS-ACNN model. One of the two types of images that we construct in Section 3 is used as the input. The model has three convolutional segments, each with one convolutional layer, one max-pooling layer, and one CS-Attention module (see Section 4.2 for details). Consistent with VggNet, we set the size of the convolution kernel to be 3 × 3.[11] To help with the convergence of the neural network, a batch normalization layer is added after the ReLU operation in each convolutional segment, which is equivalent to directly normalizing the input of each layer. We also add a dropout layer at the end of each convolutional segment to mitigate the risk of overfitting. The output from the last convolutional segment is passed through a fully-connected layer, and the probability of whether the stock price will rise in the following period ($y_t = 1$) is calculated using the Softmax function.

4.2 Attention Module

We propose the CS-Attention module to enhance the model's ability to extract visual features. In Figure 6, $F \in \mathbb{R}^{H\times W\times C}$ denotes the input feature map for each CS-Attention module, where H, W, and C represent its height, width, and the number of channels, respectively. Figure 7 shows the schema of the CS-Attention module. The module learns the attention weights along the channel and space dimensions sequentially, which is an approach consistent with the convolutional block attention module (CBAM) proposed by Woo et al. (2018). Next, we introduce the channel attention and the spatial attention mechanisms separately. While the former encodes relative importance along the channel dimension, C, the latter encodes relative importance along the spatial dimension, $H \times W$.

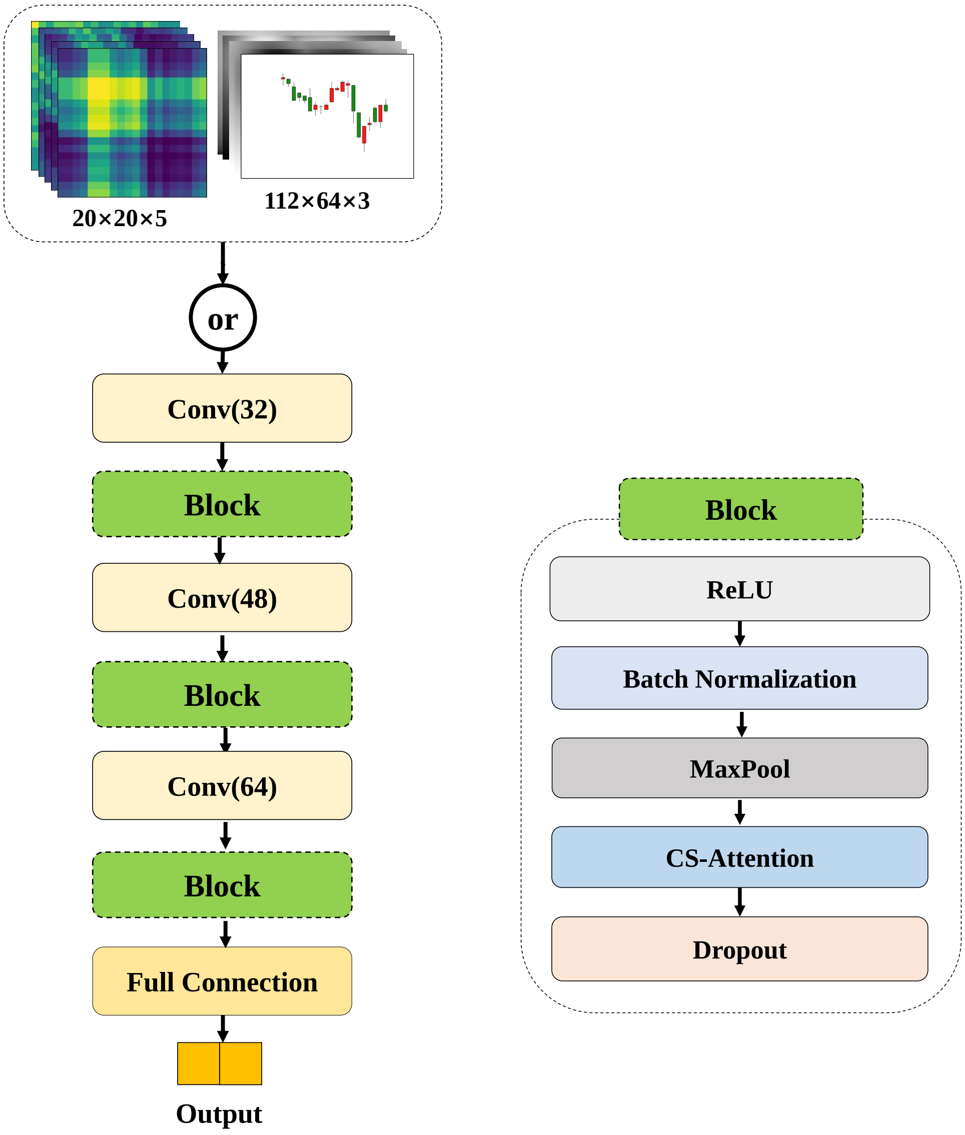
The channel attention mechanism computes attention weights for different channels of the input feature map, F, to encode relative importance at the channel level. Specifically, the channel attention map, $A_c \in \mathbb{R}^{1\times 1\times C}$, and the output feature, $F_c \in \mathbb{R}^{H\times W\times C}$, are given by:

$$A_c = \text{Sigmoid}(\text{Conv1D}(\text{AvgPool}(F)) + \text{Conv1D}(\text{MaxPool}(F))), \tag{14}$$

$$F_c = A_c \otimes F, \tag{15}$$

[10] The number of convolution kernels in the original VggNet grows from 64 to 512. We choose smaller numbers to mitigate overfitting.

[11] A small kernel is also consistent with the fact that our images have a relatively small resolution, and a small filter is able to capture local details better.



(a) Overall structure.

(b) Structure for each block.

Figure 6: Network structure of CS-ACNN model.

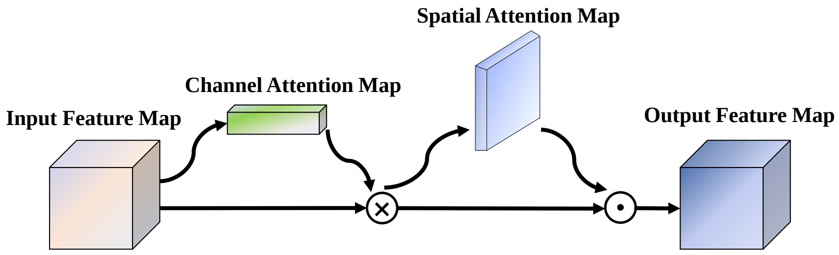


Figure 7: The overall structure of the channel and spatial attention (CS-Attention).

where $\otimes$ denotes element-wise multiplication, and Sigmoid, Conv1D, AvgPool, and MaxPool represent the sigmoid, 1-D convolution, average-pooling, and max-pooling operations, respectively. To compute the channel attention map A_c in Eq.(14), we follow the CBAM of Woo et al. (2018) by aggregating the spatial information of the feature maps using both average-pooling and max-pooling. However, similar to Wang et al. (2020), we replace the one-layer multilayer perceptron in the shared network of CBAM by a 1-D convolution to reduce the complexity of the model. This is because a 1-D convolution is much cheaper computationally than a fully-connected layer. The size k of the 1-D convolution kernel is a hyperparameter, which is determined as follows (Wang et al., 2020):

$$k = \left| \frac{\log_2(C) + b}{\gamma} \right|_{odd} \tag{16}$$

where $|\cdot|_{odd}$ represents the nearest odd number, C represents the number of channels, and $\gamma = 2$ and $b = 1$ are constants. k is a monotonically increasing function of the number of channels.

In contrast to the channel attention mechanism, the spatial attention mechanism computes attention weights for different regions of the input feature map. Because spatial attention follows the process of channel attention, it performs feature extraction on the output obtained from channel attention, F_c. We adopt the self-attention approach (Vaswani et al., 2017) to capture the dependencies between arbitrary spatial locations in F_c. First, F_c is passed through three different convolutional layers to obtain three feature maps, *query*, *key*, and *value*, all of which are of the dimension $\mathbb{R}^{(H\times W)\times C}$:[12]

$$query = \text{reshape}\left(\text{Conv2D}_q(F_c), (H \times W, C)\right), \tag{17}$$

$$key = \text{reshape}(\text{Conv2D}_k(F_c), (H \times W, C)), \tag{18}$$

$$value = \text{reshape}(\text{Conv2D}_v(F_c), (H \times W, C)), \tag{19}$$

where reshape$(M, (x, y))$ represents an operation to reconstruct the feature map M into a feature map of size (x, y), and Conv2D is the 2-D convolution. We then perform a Softmax operation on the product of *query* and *key* to obtain the attention score map, $A_s \in \mathbb{R}^{(H\times W)\times(H\times W)}$, between each spatial location:

$$A_s = \text{Softmax}(query \cdot key'). \tag{20}$$

[12] Here we use parenthesis on $H \times W$ to highlight that *query*, *key*, and *value* are two-demensional matrices, where the first dimension is of length $H \times W$ and the second dimension is of length C.

The (i, j)-th element of A_s indicates the degree of influence of the feature at the j-th spatial position on the feature at the i-th spatial position. Finally, we perform a linear transformation on the product of A_s and *value* and add the result to the original feature map, F, to form a residual connection. This yields the final output feature, $F_s \in \mathbb{R}^{H\times W\times C}$:

$$F_s = W_s \cdot (A_s \cdot value) + F, \tag{21}$$

where $W_s \in \mathbb{R}^{H\times W\times(H\times W)}$ is a weight matrix and "$\cdot$" represents matrix multiplication. Through the spatial attention mechanism illustrated in Eqs.(17)–(21), each element in the final output feature, F_s, is obtained by performing a weighted sum of features at all spatial locations. This allows the model to capture any global dependencies from the original input feature map.

4.3 Implementation Details

To summarize our CS-ACNN model and, in particular, the specific details of the implementation of the CS-Attention mechanism, we present an example where the input image size is $20 \times 20 \times 5$, as shown in Figure 8. After performing the convolution and max-pooling operations on the original input image, the size of the first input feature map F is $10 \times 10 \times 32$. Then, F is passed through the channel attention and spatial attention sequentially.

The channel attention first yields two feature maps, both of dimensions $1\times1\times32$, through global max-pooling and average-pooling, respectively. They are then passed through two 1-D convolution operations with a kernel size of 3, according to Eq.(16). We then use the Sigmoid activation function to obtain a channel attention map, A_c, with dimensions of $1 \times 1 \times 32$. Finally, F and A_c are multiplied element-wise, leading to the channel attention output feature, F_c, which is of dimensions $10 \times 10 \times 32$.

The spatial attention module takes F_c as the input, and uses three different convolution layers to obtain three matrices of dimensions 100×32, which correspond to the *query*, *key*, and *value* respectively. We then perform matrix multiplication on the *query* and *key* to obtain the spatial attention score, which is passed through a Softmax function to produce the spatial attention map, A_s, of dimensions 100×100. Finally, we multiply *value* and A_s to obtain a feature map of dimensions 100×32, which is combined with the original feature map to obtain the final output feature, F_s, of dimensions $10 \times 10 \times 32$ through residual connection. The dimensions of the output feature map from CS-Attention are consistent with those of the input feature map.

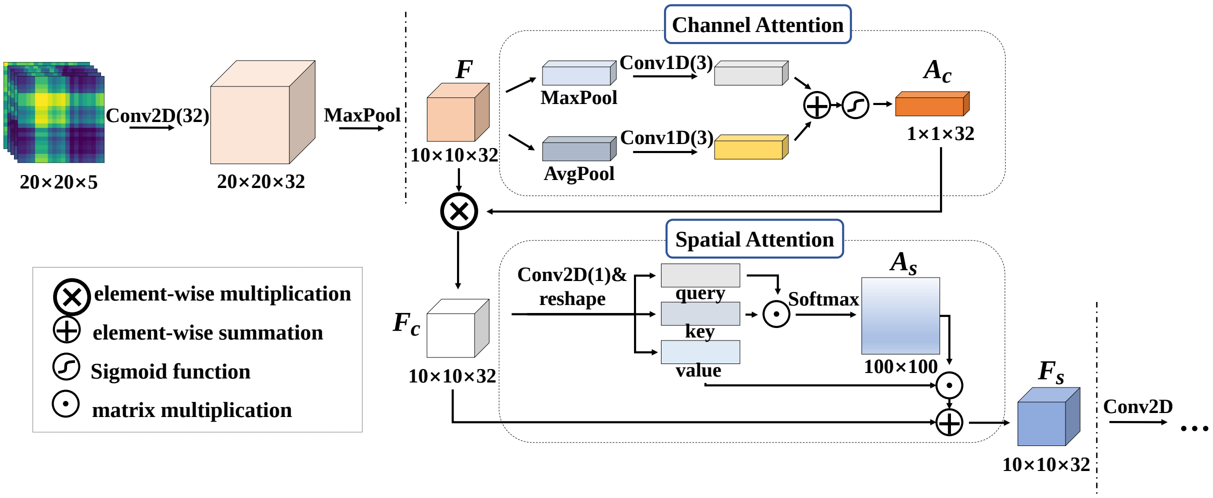


Figure 8: Implementation details of the CS-Attention module for a sample input image of dimensions $20 \times 20 \times 5$.

4.4 Loss Function and Model Training

We use cross-entropy loss because our label, y_t, is binary. We use stochastic gradient descent and the Adam algorithm (Kingma and Ba, 2015) with an initial learning rate of 0.01. We set our batch size at 256. We also adopt several common techniques to further mitigate the risk of overfitting during model training. First, we add a dropout layer at the end of each convolutional segment, where the forgetting rate is set as 0.25. Second, we adopt linear learning rate decay with respect to time to prevent instability in the training loss and gradient norm. Third, we apply early stopping when the loss of the validation set fails to improve over 30 consecutive epochs.

5 Experiment

We empirically test our CS-ACNN model on equity index exchange-traded funds (ETF) data from three major financial markets. We evaluate our model against several benchmark models in terms of both classification performance metrics and strategy profit. The former evaluates how well the model is able to distinguish between upward and downward price movements, while the latter measures the model's profitability in actual financial markets.

We provide a comprehensive analysis to understand the effects of different input images and different components of our model. Section 6 provides additional analysis regarding the interpretability of our model.

5.1 Data

Our dataset consists of three equity index ETFs, including the SPDR S&P 500 ETF (SPY) for US stocks, the Hang Seng Index ETF (2833.HK) for Hong Kong stocks, and the China AMC SSE 50 ETF (510050.SS) for Chinese A-shares, thus covering both developed and emerging markets. We collect daily time series data for opening price, high price, low price, closing price, and volume (OHLCV) starting from the launch date of each ETF until February 10, 2022. Specifically, the starting date is January 29, 1993 for SPY, September 21, 2004 for 2833.HK, and February 23, 2005 for 510050.SS. These time series are converted into 2-D images, as discussed in Section 3, and we divide the data into a training set (64%), a validation set (16%), and a test set (20%) chronologically to avoid look-ahead bias. The validation set is used to determine the hyperparameters of the model. We report out-of-sample results on the test set throughout this section.

5.2 Model Performance

Because our learning task is a binary classification, the confusion matrix characterizes the model's performance, which contains the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Based on this confusion matrix, we calculate the accuracy, precision, recall, and specificity, which are defined as below:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{22}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{23}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{24}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{25}$$

In addition, we also include the area under the receiver operating characteristic (ROC) curve (AUC) as an evaluation metric.

5.2.1 Effectiveness of Image Augmentation

As described in Section 3.1, we augment the candlestick image using four additional images that enhance different regions of the original image. Here we analyze whether this image augmentation procedure contributes to model performance. In particular, we consider six different configurations of input images for the CS-ACNN model:

(1) Original candlestick;

(2) Original candlestick + Enhanced candlestick center + Weakened candlestick center + Enhanced candlestick edge + Weakened candlestick edge;

(3) Original candlestick + Enhanced candlestick center + Enhanced candlestick edge;

(4) Original candlestick + Weakened candlestick center + Weakened candlestick edge;

(5) Original candlestick + Enhanced candlestick center + Weakened candlestick center;

(6) Original candlestick + Enhanced candlestick edge + Weakened candlestick edge.

Configuration (2) uses all five images (original candlestick image and four augmented images), while the other configurations use subsets of these images.

Table 1: Performance of the CS-ACNN model when different configurations of the candlestick images are used as input. In each panel, the best performance for each metric across different configurations is highlighted in bold.

Image Configuration	**Accuracy**	**Precision**	**Recall**	**Specificity**	**AUC**
Panel A: SPY					
(1)	0.534	0.562	0.796	0.193	0.511
(2)	**0.573**	**0.597**	0.753	**0.339**	**0.566**
(3)	0.525	0.558	0.766	0.212	0.513
(4)	0.549	0.571	**0.815**	0.204	0.533
(5)	0.546	0.572	0.782	0.239	0.544
(6)	0.552	0.575	0.800	0.229	0.539
Panel B: 2833.HK					
(1)	0.528	0.592	0.480	0.588	0.533
(2)	**0.571**	**0.638**	0.523	**0.631**	**0.562**
(3)	0.520	0.582	0.471	0.580	0.513
(4)	0.546	0.610	0.501	0.602	0.527
(5)	0.541	0.596	0.533	0.551	0.522
(6)	0.549	0.600	**0.559**	0.537	0.534
Panel C: 510050.SS					
(1)	0.523	0.542	0.566	0.475	0.521
(2)	**0.566**	**0.593**	0.545	**0.590**	**0.557**
(3)	0.518	0.538	0.557	0.475	0.515
(4)	0.542	0.556	**0.611**	0.465	0.528
(5)	0.545	0.560	0.607	0.478	0.540
(6)	0.548	0.564	0.592	0.499	0.522

Table 1 summarizes model performance for different configurations of the input images, which demonstrates the benefit of image augmentation. When all five images are used as the input in configuration (2), the model achieves the highest performance in terms of accuracy, precision, specificity, and AUC. Configurations (3)–(6) also demonstrate similar or slightly better performances compared to the original candlestick image in configuration (1). However, none of these surpass the values of the performance metrics obtained when using all five images. As a result, we use configuration (2) for the candlestick images in all subsequent experiments.

5.2.2 Effectiveness of Images as Input

In this section, we investigate whether the two types of image construction techniques in Section 3 are effective as inputs to the model, as compared to the raw time series data of the financial asset. Here the raw financial time series refers to a matrix constructed directly from the OHLCV in a given time window. We run experiments for the CS-ACNN model by providing the augmented candlestick charts, the GAF images, and the raw time series of OHLCV, respectively, as input. We also compared it with several popular machine learning models that directly take the same OHLCV time series as input, including a support vector machine (SVM), a long short-term memory (LSTM),[13] and a 1D-CNN.[14]

Table 2 summarizes the model performance metrics. The first three rows of each panel contain results for our model. In general, GAF images and candlestick images yield similar performances across the three ETF products in terms of both accuracy and AUC. They also lead to similar precision, recall, and specificity for SPY. However, the CS-ACNN model trained on GAF images has higher recall and lower precision for 2833.HK and 510050.SS, demonstrating slightly different tradeoffs. When CS-ACNN takes time series data as input directly, the performance metrics are generally much worse. The model accuracy and AUC are both approximately 0.05 lower than that for image inputs.

In addition, SVM, LSTM, and 1D-CNN, which directly take time series data as input, generally underperform compared to the CS-ACNN. LSTM performs the best among these three models in terms of the AUC, which is still found to be much lower than the AUC of the best CS-ACNN.

Overall, our results confirm that both the candlestick charts and GAF images constructed from time series data can be used as inputs to neural networks to effectively capture future

[13] We configure the LSTM to be: Hidden layer (32 neurons) + Hidden layer (64 neurons) + Dropout(0.25) + Fully connected layer.

[14] We configure the 1D-CNN to be: Conv1D(32) + MaxPool1D + Conv1D(48) + MaxPool1D + Dropout(0.25) + Conv1D(64) + GlobalAveragePool1D + Dropout(0.25) + Fully connected layer.

Table 2: Performance comparison of GAF images and candlestick charts with time series data as input to machine learning models. In each panel, the best performance for each metric across different configurations is highlighted in bold.

Model	Input Data	Accuracy	Precision	Recall	Specificity	AUC
Panel A: SPY						
CS-ACNN	GAF	0.567	0.591	0.764	0.312	0.547
	Candlestick	**0.573**	0.597	0.753	0.339	**0.568**
	Time Series	0.520	0.623	0.382	0.699	0.517
SVM	Time Series	0.555	0.561	**0.974**	0.010	0.514
LSTM	Time Series	0.529	**0.684**	0.310	0.814	0.522
1D-CNN	Time Series	0.501	0.673	0.229	**0.855**	0.507
Panel B: 2833.HK						
CS-ACNN	GAF	0.565	0.609	0.602	0.519	**0.555**
	Candlestick	**0.571**	0.638	0.523	0.631	0.551
	Time Series	0.524	0.587	0.477	0.583	0.509
SVM	Time Series	0.558	0.557	**0.994**	0.016	0.508
LSTM	Time Series	0.529	0.593	0.480	0.591	0.511
1D-CNN	Time Series	0.509	**0.688**	0.209	**0.882**	0.513
Panel C: 510050.SS						
CS-ACNN	GAF	0.551	0.563	**0.637**	0.457	0.545
	Candlestick	**0.566**	0.593	0.545	0.590	**0.549**
	Time Series	0.512	0.535	0.518	0.505	0.504
SVM	Time Series	0.501	0.529	0.410	0.600	0.487
LSTM	Time Series	0.515	0.537	0.538	0.491	0.515
1D-CNN	Time Series	0.502	**0.656**	0.100	**0.943**	0.498

price movements of financial assets. Moreover, they outperform several powerful benchmark models for time series data. This highlights the potential of converting time series data into images for asset trend predictions, if the machine learning model is properly constructed to extract image features.

5.2.3 Effectiveness of the CS-ACNN Model

To verify the effectiveness of our CS-ACNN model, we compare its performance with two benchmark models that can also handle image inputs but do not use the attention mechanism. The first is the SVM, which directly uses the normalized image matrix as the model input. We use the Gaussian radial basis function as the kernel and search for the optimal hyperparameters by conducting a grid search on the validation data. The second is the CNN-TA proposed by Sezer and Ozbayoglu (2018), which is a state-of-the-art neural network for stock trend prediction based on images. The network structure of CNN-TA is given by: Conv2D(32) + Conv2D(64) + MaxPool(2) + Dropout(0.25) + Fully connected layer + Dropout(0.25) + Output layer, where the size of the convolution kernel is 3×3. Our model is different from the CNN-TA primarily due to the presence of the attention mechanism.

Table 3 compares the performance of our CS-ACNN model with the two benchmark models. Our model achieves the best performance in terms of both accuracy and AUC. The average classification accuracy of CS-ACNN is approximately 5% higher than that of other models, which is very significant in trend prediction tasks on financial data whose signal-to-noise ratio is typically very low. When using GAF images as model input, the CNN-TA achieves better performances compared to the SVM, but falls short when compared to the CS-ACNN. When using candlestick charts as model input, both the SVM and CNN-TA simply classify all test samples into a single category and fail to make a meaningful distinction between upward and downward trends.

Overall, these results demonstrate the superiority of our model, and in particular, that the channel and spatial attention modules can indeed enhance the model's ability to extract visual features.

5.3 Profitability

To evaluate the economic significance of our model, we implemented both long-only and long-short strategies based on model predictions of asset price trends. We compute several metrics that capture the returns and risks of implementing these strategies. In addition, we compare these results with a simple buy-and-hold strategy, which acts as a benchmark. All results in this section are out-of-sample.

Table 3: Performance comparison of the CS-ACNN model against benchmark models when GAF images and candlestick charts are used as input images, respectively. In each panel, the best performance for each metric across different configurations is highlighted in bold.

Model	Image	Accuracy	Precision	Recall	Specificity	AUC
Panel A: SPY						
CS-ACNN	GAF	0.567	0.591	0.764	0.312	0.547
	Candlestick	**0.573**	**0.597**	0.753	**0.339**	**0.568**
SVM	GAF	0.517	0.553	0.764	0.196	0.491
	Candlestick	0.565	0.565	**1.000**	0.000	0.500
CNN-TA	GAF	0.542	0.567	0.803	0.202	0.522
	Candlestick	0.565	0.565	**1.000**	0.000	0.500
Panel B: 2833.HK						
CS-ACNN	GAF	0.565	0.609	0.602	0.519	**0.555**
	Candlestick	**0.571**	**0.638**	0.523	**0.631**	0.551
SVM	GAF	0.504	0.567	0.445	0.578	0.491
	Candlestick	0.554	0.554	**1.000**	0.000	0.500
CNN-TA	GAF	0.524	0.591	0.460	0.604	0.522
	Candlestick	0.554	0.554	**1.000**	0.000	0.500
Panel C: 510050.SS						
CS-ACNN	GAF	0.551	0.563	0.637	0.457	0.545
	Candlestick	**0.566**	**0.593**	0.545	**0.590**	**0.549**
SVM	GAF	0.511	0.532	0.531	0.488	0.518
	Candlestick	0.523	0.523	**1.000**	0.000	0.500
CNN-TA	GAF	0.530	0.546	0.604	0.449	0.522
	Candlestick	0.523	0.523	**1.000**	0.000	0.500

In particular, without loss of generality, we assume that the investor's initial capital is $V_0 = 1$ and the transaction cost is a fixed percentage of the transaction value, given by $c\% = 0.15\%$. Further, we outline the specific trading rules of the three strategies below. We follow the notations given in Eqs.(12)–(13) and denote the model prediction on date $t-1$ by $\hat{y}_{t-1} \in \{0, 1\}$, and the return on date t as r_t.

Long-only strategy. Investors start with zero position and $V_0 = 0$. On date t, the portfolio capital before transaction cost is determined by the model prediction on date $t-1$:

$$\widetilde{V}_t = \begin{cases} V_{t-1}(1+r_t), & \text{if } \hat{y}_{t-1} = 1, \\ V_{t-1}, & \text{if } \hat{y}_{t-1} = 0. \end{cases} \tag{26}$$

The total capital after transaction cost is determined by whether the position on date t is the same as the position on date $t-1$:

$$V_t = \begin{cases} \widetilde{V}_t, & \text{if } \hat{y}_{t-1} = \hat{y}_{t-2}, \\ \widetilde{V}_t(1-c\%), & \text{if } \hat{y}_{t-1} \neq \hat{y}_{t-2}. \end{cases} \tag{27}$$

Long-short strategy. Investors start with zero position and $V_0 = 0$. On date t, the portfolio capital before transaction cost is determined by the model prediction on date $t-1$:

$$\widetilde{V}_t = \begin{cases} V_{t-1}(1+r_t), & \text{if } \hat{y}_{t-1} = 1, \\ V_{t-1}(1-r_t), & \text{if } \hat{y}_{t-1} = 0. \end{cases} \tag{28}$$

The total capital after transaction cost is determined by whether the position on date t is the same as the position on date $t-1$:

$$V_t = \begin{cases} \widetilde{V}_t, & \text{if } \hat{y}_{t-1} = \hat{y}_{t-2}, \\ \widetilde{V}_t(1-2c\%), & \text{if } \hat{y}_{t-1} \neq \hat{y}_{t-2}. \end{cases} \tag{29}$$

The transaction fee here is twice that of the long-only strategy because investors need to close the long (or short) position first before opening a new short (or long) position.

Buy-and-hold strategy. Investors simply allocate all capital to the ETF and stay fully invested throughout the period:

$$V_t = V_0 \cdot \frac{z_t}{z_0} \cdot (1-c\%) = (1-c\%)V_0 \prod_{s=1}^{t}(1+r_t). \tag{30}$$

For each investment strategy, given a realized return time series, $R_t \equiv \frac{V_t - V_{t-1}}{V_{t-1}}$, $t = 1, 2, \cdots, T$, we compute several evaluation metrics including the annualized return, the Sharpe ratio, and the maximum drawdown:

$$\text{AnnualR} = \frac{\sum_{t=1}^{T} R_t}{T} \times 252, \tag{31}$$

$$\text{SR} = \frac{\text{AnnualR} - r_f}{\hat{\sigma}}, \tag{32}$$

$$\text{MDD} = \max_{1 \le t < s \le T} \frac{V_t - V_s}{V_t}, \tag{33}$$

where r_f represents the annualized risk-free interest rate, which is set to be 2% in this paper, and $\hat{\sigma} = \sqrt{\sum_{t=1}^{T} (R_t - \text{AnnualR}/252)^2 / T \times 252}$ is the sample standard deviation of the realized returns.

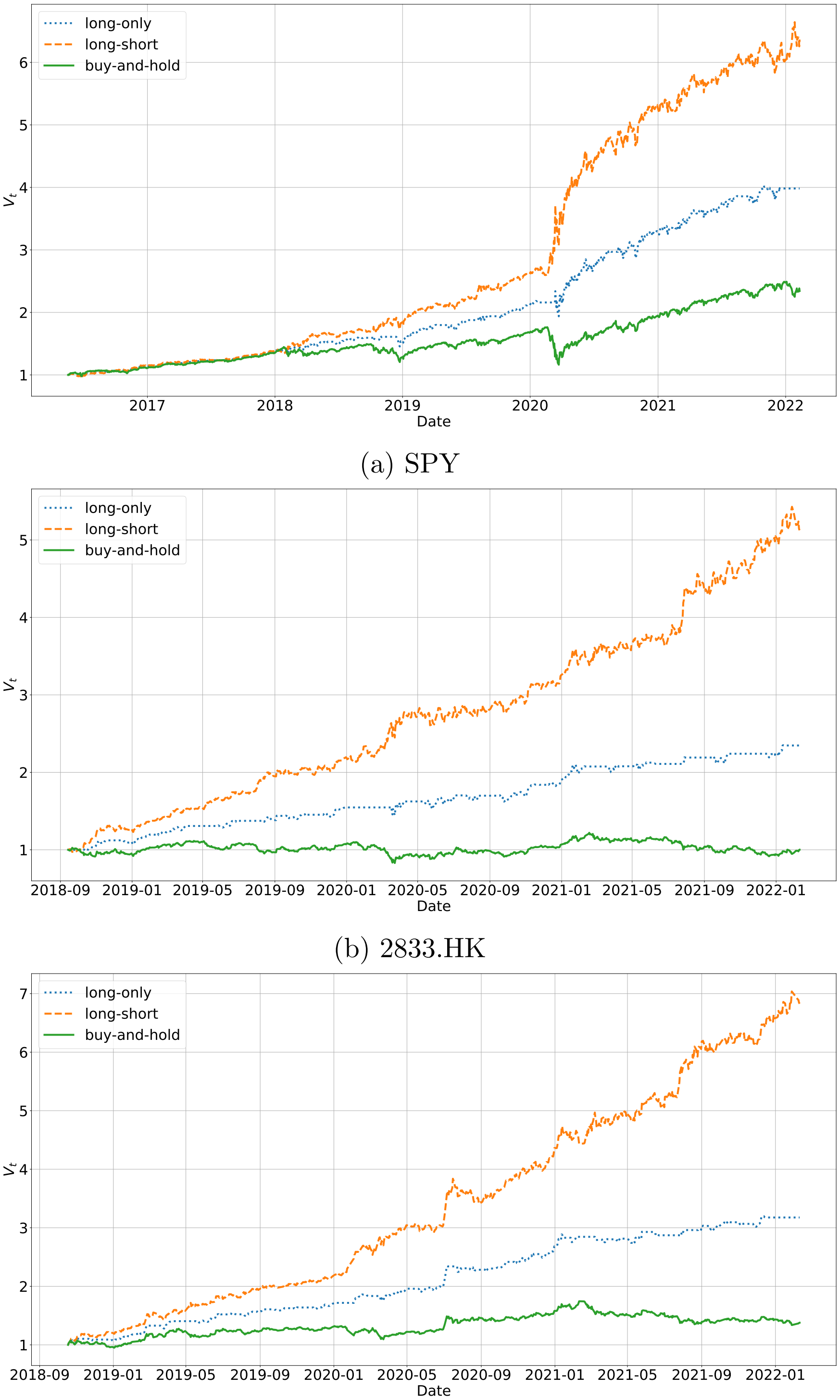
Table 4 reports the out-of-sample metrics across a wide range of models that use different input data. The CS-ACNN model with candlestick charts as input generally records the best performance, which is closely followed by the CS-ACNN model with GAF images as input. The advantage of using these models is particularly consistent for long-short strategies. For example, the long-short strategies based on the best CS-ACNN models reach an annualized return of 33.95%, 51.24%, and 62.12%, for SPY, 2833.HK, and 510050.SS respectively. The Sharpe ratios for these scenarios reach as high as 1.78, 2.46, and 3.03, respectively. These performance metrics are obtained after accounting for transaction costs, which highlights the potential of our model in terms of delivering superior risk-adjusted returns.

The only exception to the superior performance of CS-ACNN is the Sharpe ratio and maximum drawdown of the long-only strategy for SPY. The LSTM and 1D-CNN deliver very low volatilities and, therefore, high Sharpe ratios and low maximum drawdowns. However, this is because they rarely predict upward price movements and therefore trade very little, which is also reflected by their low recalls, as shown in Panel A of Table 2. This is usually not preferred by investors because of the low potential for returns. Overall, our CS-ACNN model with image inputs demonstrates the most robust financial performance.

To visualize the profitability of our model, Figure 9 illustrates the time series of the total portfolio wealth, V_t, of the investment strategies based on the CS-ACNN model with candlestick charts as input. The portfolio wealth for both the long-only and long-short strategies rises steadily with limited drawdowns, and they both reach significant excess returns beyond the buy-and-hold strategy. This further demonstrates the superior out-of-sample performance of our model.

Table 4: Profitability of investment strategies based on each model. In each panel, the best performance for each metric across different configurations is highlighted in bold.

Model	Input	AnnualR		SR		MDD	
		long-only	long-short	long-only	long-short	long-only	long-short
Panel A: SPY							
CS-ACNN	Time Series	18.24%	19.94%	1.48	0.99	9.70%	28.60%
	GAF	24.14%	31.72%	1.44	1.65	20.13%	19.43%
	Candlestick	**25.25%**	**33.95%**	1.57	**1.78**	17.22%	**17.22%**
SVM	Time Series	15.38%	14.23%	0.74	0.67	34.10%	34.10%
	GAF	12.15%	7.77%	0.59	0.32	33.08%	43.21%
	Candlestick	16.53%	16.53%	0.80	0.80	34.10%	34.10%
CNN-TA	GAF	15.49%	14.45%	0.78	0.69	28.74%	28.74%
	Candlestick	16.53%	16.53%	0.80	0.80	34.10%	34.10%
LSTM	Time Series	12.00%	7.45%	**1.97**	0.30	8.47%	38.48%
1D-CNN	Time Series	10.72%	4.90%	1.96	0.16	**3.97**%	56.48%
Buy-and-hold		17.81%		0.87		34.10%	
Panel B: 2833.HK							
CS-ACNN	Time Series	10.53%	18.88%	0.58	0.83	14.65%	14.64%
	GAF	22.12%	42.05%	1.32	2.00	10.82%	11.01%
	Candlestick	**26.71%**	**51.24%**	**1.75**	**2.46**	**7.13**%	**7.93%**
SVM	Time Series	2.14%	2.14%	0.01	0.01	25.43%	25.43%
	GAF	8.50%	14.84%	0.45	0.63	16.92%	15.96%
	Candlestick	2.15%	2.15%	0.01	0.01	25.67%	25.67%
CNN-TA	GAF	12.84%	23.51%	0.76	1.06	12.63%	13.99%
	Candlestick	2.15%	2.15%	0.01	0.01	25.67%	25.67%
LSTM	Time Series	12.30%	22.42%	0.25	1.01	14.18%	14.42%
1D-CNN	Time Series	14.64%	27.12%	1.28	1.24	10.82%	23.17%
Buy-and-hold		0.27%		-0.08		25.67%	
Panel C: 510050.SS							
CS-ACNN	Time Series	10.82%	9.49%	0.55	0.36	21.31%	27.51%
	GAF	35.09%	58.00%	2.05	2.81	8.58%	**10.34%**
	Candlestick	**37.16%**	**62.12%**	**2.52**	**3.03**	5.97%	11.14%
SVM	Time Series	4.17%	-3.80%	0.16	-0.28	20.21%	42.17%
	GAF	17.14%	22.12%	0.91	0.98	15.02%	16.78%
	Candlestick	12.15%	12.15%	0.49	0.49	23.35%	23.35%
CNN-TA	GAF	26.09%	39.98%	1.58	1.89	9.44%	10.88%
	Candlestick	12.15%	12.15%	0.49	0.49	23.35%	23.35%
LSTM	Time Series	17.24%	22.31%	0.96	0.98	15.02%	17.29%
1D-CNN	Time Series	11.84%	11.53%	1.36	0.46	**3.30%**	32.33%
Buy-and-hold		13.75%		0.57		22.54%	



(a) SPY

(b) 2833.HK

(c) 510050.SS

Figure 9: The time series of total portfolio wealth for investment strategies based on the CS-ACNN model with candlestick charts as input.

6 Model Interpretability

For financial applications, it is important to understand the patterns that the model learns. This helps investors verify whether the model indeed learns patterns that conform to economic intuitions, and diagnose whether this performance is likely to persist in the future. To that end, we visualize the intermediate output of the CS-ACNN model using the gradient-weighted class activation mapping (Grad-CAM) (Selvaraju et al., 2017).

6.1 Visualization Methodology

To explore how the CS-ACNN model activates different regions of the input image, we use the candlestick charts as an example. Figure 10 demonstrates how Grad-CAM (Selvaraju et al., 2017) generates a heatmap of class activations for the last convolutional layer of the CS-ACNN model. For any label y ("up" or "down"), Grad-CAM first obtains the weight $w_1^y, w_2^y, \cdots, w_n^y$ of each channel of the feature map using back propagation:

$$w_k^y = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{Global Ave. Pool.}} \qquad \underbrace{\frac{\partial u^y}{\partial F_{ij}^k}}_{\text{Backprop Grad.}} \tag{34}$$

where u^y represents the value corresponding to category y immediately before the Softmax operation, F represents the feature map obtained by convolution, k represents the channel of the feature map, i, j represent the horizontal and vertical coordinates of the feature map, respectively, and Z represents the dimensions of the feature map (i.e., length times width). This process is equivalent to finding the mean value of the gradients on the feature map, or a global average-pooling operation.

After that, each channel of the feature map is linearly weighted to obtain a heat map of the same size as the convolution feature map. Grad-CAM adds a ReLU operation to the fused heat map, the goal of which is to activate the regions with a positive effect on category y:

$$L_{\text{Grad-CAM}}^y = \text{ReLU}\left(\sum_k w_k^y F^k\right). \tag{35}$$

6.2 Learning Technical Patterns

We apply the visualization methodology discussed in Section 6.1 and demonstrate how the CS-ACNN model captures well-known technical patterns in financial markets. In particular,

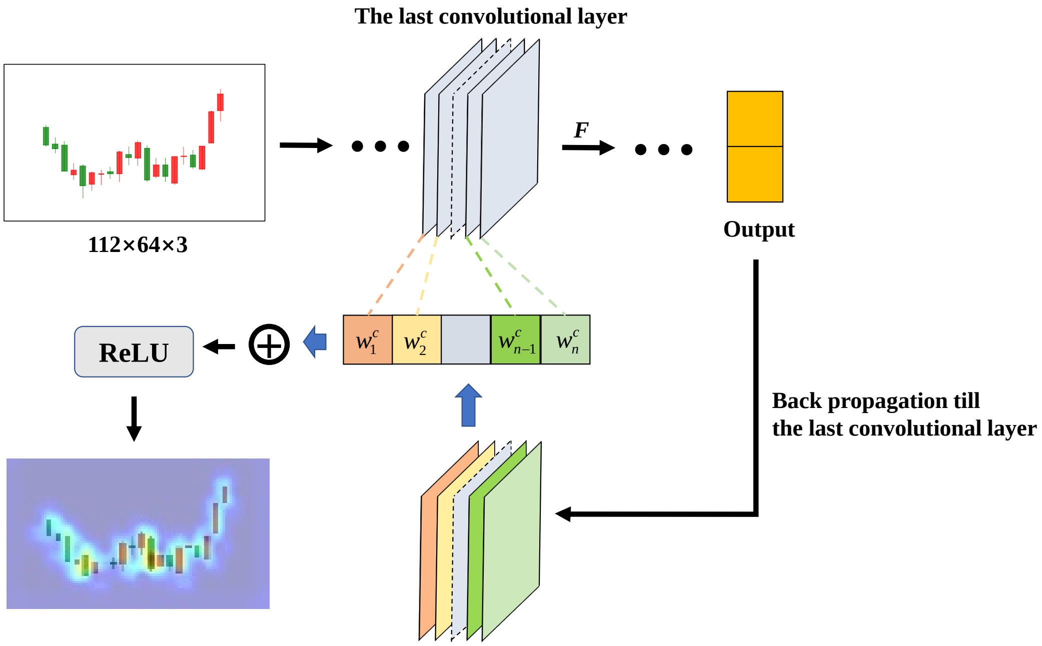


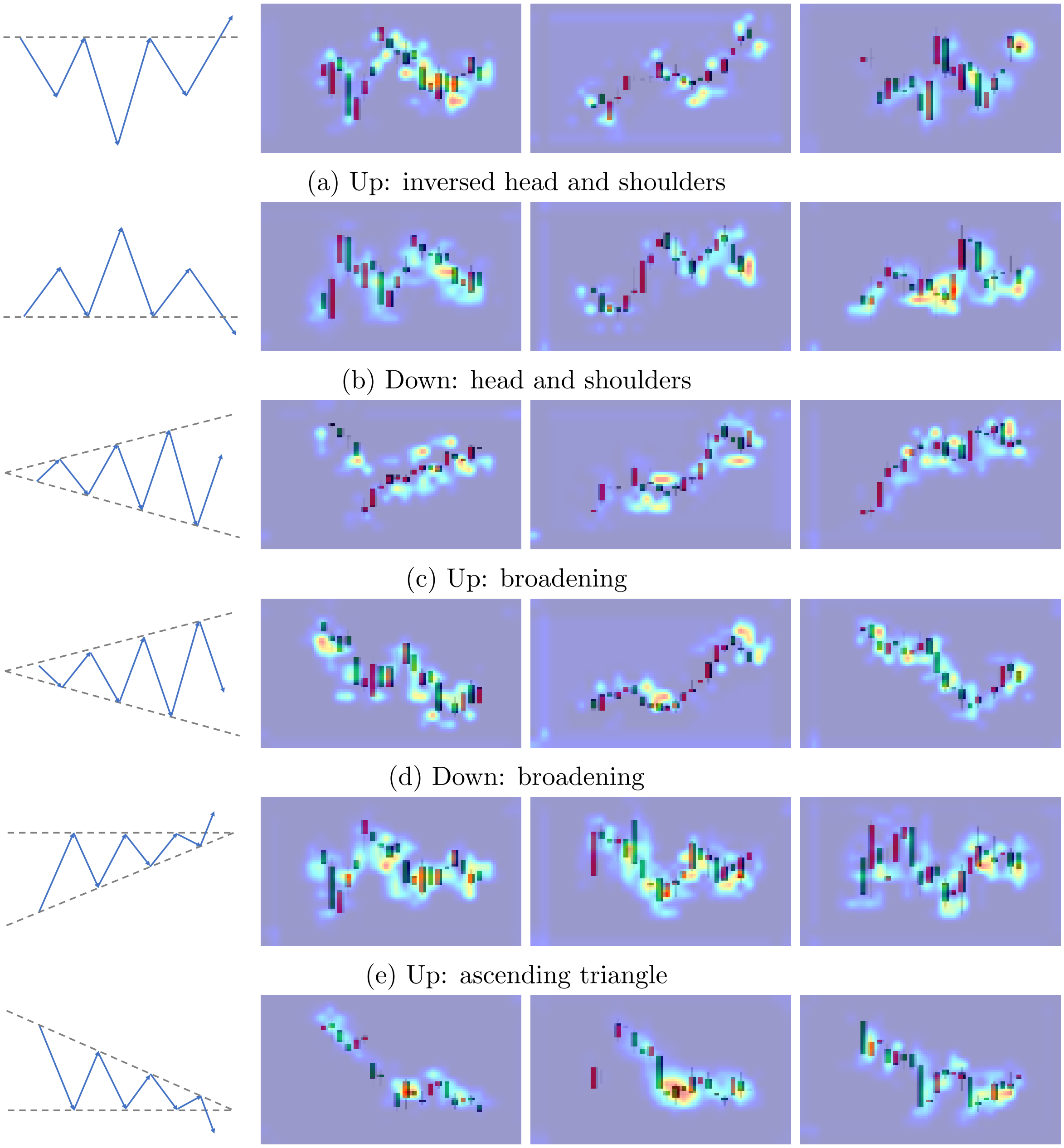
Figure 10: A schematic illustration of Grad-CAM.

for samples that are correctly classified as “up” or “down”, we apply the Grad-CAM technique to generate heatmaps corresponding to the last convolutional block of the CS-ACNN. We then explore the regions highlighted by our model and demonstrate that they match several classical technical patterns. We discuss results for 510050.SS as an example.

Based on Lo, Mamaysky, and Wang (2000), we choose the three most popular technical patterns: head and shoulder, broadening, and triangle. The left-most image of each row in Figure 11 shows the toy diagrams of these patterns. We then include three examples from our dataset that resemble each of these patterns. In each example, the highlighted regions are those that the CS-ACNN model focuses on according to the Grad-CAM methodology. Most of these highlighted regions are located in the right half of the input image, which corresponds to recent price information.

In particular, Figures 11a and 11b demonstrate sample images that resemble the head-and-shoulder technical pattern, for “up” and “down” samples, respectively. The highlighted regions focus mostly on the turning points, i.e., the knots of the “heads” and “shoulders”, while the trends between two turning points are generally focused on to a lesser degree. This is consistent with the fact that head-and-shoulder patterns are characterized by these turning points.

Figures 11c and 11d demonstrate sample images that resemble the broadening technical pattern, for “up” and “down” samples, respectively. In contrast with the results of the head


(a) Up: inversed head and shoulders

(b) Down: head and shoulders

(c) Up: broadening

(d) Down: broadening

(e) Up: ascending triangle

(f) Down: descending triangle

Figure 11: Toy diagrams for several technical analysis patterns, and heatmaps generated by the Grad-CAM for the last convolutional layer of CS-ACNN, based on 20-day candlesticks of 510050.SS.

and shoulders, the CS-ACNN model typically highlights the two edges (upper and lower) of the entire trend of the candlesticks. This is also consistent with the fact that broadening patterns are defined as vibrations within a wedge, which is the envelope around the trend.

Figures 11e and 11f demonstrate sample images that resemble the triangle technical pattern, for "up" and "down" samples, respectively. In this case, the CS-ACNN model focuses on almost the entire trend, especially on the right-hand side where the most recent price information is contained. This can be rationalized because the direction of the triangle is determined by the last few candlesticks.

It is striking that, by visualizing images using Grad-CAM, we can open the black box for CS-ACNN and interpret its learned patterns by matching many of its predictions to classical technical patterns. This process demonstrates that deep learning models based on image data can help investors uncover visual patterns that resemble traditional technical analysis. In addition, it allows investors to gain more confidence in the model by understanding its underlying economic rationale, thereby making the CS-ACNN model more widely applicable in practice.

7 Conclusion

In this paper, we construct two types of images based on widely-available financial time series data for price trend prediction. We propose the CS-ACNN model, which introduces channel and spatial attention mechanisms between convolutional layers to enhance the model's ability to extract information from images relevant to price trends. Through empirical analysis of three widely-traded ETFs in the US, Hong Kong, and mainland China, respectively, we demonstrate the effectiveness of our framework in terms of both classification metrics and investment profitability.

Our model achieves impressive out-of-sample profitability outcomes using only the price and volume information of the underlying asset. For example, the annualized returns range from 25.25% to 37.16% for the long-only strategy, and from 33.95% to 62.12% for the long-short strategy, after account for transaction costs. These returns stay strong after adjusting for risks, as demonstrated by Sharpe ratios ranging from 1.57 to 2.52 for the long-only strategy, and from 1.78 to 3.03 for the long-short strategy. These return characteristics can provide tremendous value for investors in practice.

In addition, we demonstrate that the images constructed based on our methodology can lead to better performance compared to models based on the original time series data. Furthermore, our CS-ACNN model outperforms existing image-based deep learning models for price trend prediction due to our novel attention mechanism. These results provide a promising direction for further research, in which more information can be incorporated into the images—such as technical indicators, firm fundamentals, and investor sentiment—and more sophisticated deep learning architectures can be explored for performing financial tasks.

Model interpretability is a particularly important requirement for machine learning applications in finance. We further develop a methodology to interpret our black-box model by visualizing the last convolutional layers of the CS-ACNN model using Grad-CAM. We find that the model activates certain regions in the images that are the most relevant for future price trends. In addition, the learned patterns closely resemble technical patterns in traditional technical analysis. This striking result demonstrates the ability of our model to learn meaningful patterns with appropriate economic rationale, which provides additional confidence in the model for investors in real-world applications.

Acknowledgements

Funding support from Peking University is gratefully acknowledged. We thank Xiuli Shao for very helpful comments and discussion.

References

Barroso, P., and K. Saxena, 2022, Lest we forget: Learn from out-of-sample forecast errors when optimizing portfolios, *The Review of Financial Studies* 35, 1222–1278.

Bianchi, D., M. Büchner, and A. Tamoni, 2021, Bond risk premiums with machine learning, *The Review of Financial Studies* 34, 1046–1089.

Borgefors, G., 1986, Distance transformations in digital images, *Computer Vision, Graphics, and Image Processing* 34, 344–371.

Chatigny, P., R. Goyenko, and C. Zhang, 2021, Asset pricing with attention guided deep learning, Available at SSRN 3971876.

Chen, L., M. Pelger, and J. Zhu, 2022, Deep learning in asset pricing, *Management Science* Forthcoming.

Chen, S., and L. Ge, 2019, Exploring the attention mechanism in LSTM-based Hong Kong stock price movement prediction, *Quantitative Finance* 19, 1507–1515.

Chen, Y., J. Wu, and Z. Wu, 2022, China's commercial bank stock price prediction using a novel K-means-LSTM hybrid approach, *Expert Systems with Applications* 202, 117370.

Cong, L. W., K. Tang, J. Wang, and Y. Zhang, 2021a, AlphaPortfolio: Direct construction through deep reinforcement learning and interpretable AI, Available at SSRN 3554486.

Cong, L. W., K. Tang, J. Wang, and Y. Zhang, 2021b, Deep sequence modeling: Development and applications in asset pricing, *The Journal of Financial Data Science* 3, 28–42.

Culkin, R., and S. R. Das, 2017, Machine learning in finance: The case of deep learning for option pricing, *Journal of Investment Management* 15, 92–100.

Fan, J., Y. Ke, and Y. Liao, 2021, Augmented factor models with applications to validating market risk factors and forecasting bond risk premia, *Journal of Econometrics* 222, 269–294.

Fang, J., J. Lin, S. Xia, Z. Xia, S. Hu, X. Liu, and Y. Jiang, 2020, Neural network-based automatic factor construction, *Quantitative Finance* 20, 2101–2114.

Fang, X., J. Zhu, R. Zhang, X. Shao, and H. Wang, 2021, IBNet: Interactive branch network for salient object detection, *Neurocomputing* 465, 574–583.

Fischer, T., and C. Krauss, 2018, Deep learning with long short-term memory networks for financial market predictions, *European Journal of Operational Research* 270, 654–669.

Freyberger, J., A. Neuhierl, and M. Weber, 2020, Dissecting characteristics nonparametrically, *The Review of Financial Studies* 33, 2326–2377.

Gao, R., X. Zhang, H. Zhang, Q. Zhao, and Y. Wang, 2022, Forecasting the overnight return direction of stock market index combining global market indices: A multiple-branch deep learning approach, *Expert Systems with Applications* 194, 116506.

Ghoshal, S., and S. Roberts, 2020, Thresholded ConvNet ensembles: Neural networks for technical forecasting, *Neural Computing and Applications* 32, 15249–15262.

Giglio, S., B. Kelly, and D. Xiu, 2021, Factor models, machine learning, and asset pricing, *Annual Review of Financial Economics* Forthcoming.

Gu, S., B. Kelly, and D. Xiu, 2020, Empirical asset pricing via machine learning, *The Review of Financial Studies* 33, 2223–2273.

Gu, S., B. Kelly, and D. Xiu, 2021, Autoencoder asset pricing models, *Journal of Econometrics* 222, 429–450.

Guo, S.-J., F.-C. Hsu, and C.-C. Hung, 2018, Deep candlestick predictor: a framework toward forecasting the price movement from candlestick charts, in *2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, 219–226, IEEE.

Jerez, T., and W. Kristjanpoller, 2020, Effects of the validation set on stock returns forecasting, *Expert Systems with Applications* 150, 113271.

Jiang, J., B. T. Kelly, and D. Xiu, 2022, (Re-) imag (in) ing price trends, Chicago Booth Research Paper.

Jmour, N., S. Zayen, and A. Abdelkrim, 2018, Convolutional neural networks for image classification, in *2018 international Conference on Advanced Systems and Electric Technologies (IC_ASET)*, 397–402, IEEE.

Kim, S., 2019, Enhancing the momentum strategy through deep regression, *Quantitative Finance* 19, 1121–1133.

Kingma, D. P., and J. Ba, 2015, Adam: A method for stochastic optimization, in *International Conference on Learning Representations (ICLR)*.

Koshiyama, A., N. Firoozye, and P. Treleaven, 2021, Generative adversarial networks for financial trading strategies fine-tuning and combination, *Quantitative Finance* 21, 797–813.

Kumbure, M. M., C. Lohrmann, P. Luukka, and J. Porras, 2022, Machine learning techniques and data for stock market forecasting: A literature review, *Expert Systems with Applications* 116659.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner, 1998, Gradient-based learning applied to document recognition, *Proceedings of the IEEE* 86, 2278–2324.

Lee, K., and G. Jo, 1999, Expert system for predicting stock market timing using a candlestick chart, *Expert Systems with Applications* 16, 357–364.

Lo, A. W., H. Mamaysky, and J. Wang, 2000, Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation, *The Journal of Finance* 55, 1705–1765.

Long, W., Z. Lu, and L. Cui, 2019, Deep learning-based feature engineering for stock price movement prediction, *Knowledge-Based Systems* 164, 163–173.

Ma, T., and Y. Tan, 2022, Stock ranking with multi-task learning, *Expert Systems with Applications* 199, 116886.

Moews, B., J. M. Herrmann, and G. Ibikunle, 2019, Lagged correlation-based deep learning for directional trend change prediction in financial time series, *Expert Systems with Applications* 120, 197–206.

Nagel, S., 2021, *Machine Learning in Asset Pricing* (Princeton University Press).

Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, 2017, Grad-CAM: Visual explanations from deep networks via gradient-based localization, in *Proceedings of the IEEE International Conference on Computer Vision*, 618–626.

Sezer, O. B., and A. M. Ozbayoglu, 2018, Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach, *Applied Soft Computing* 70, 525–538.

Shu, L., F. Shi, and G. Tian, 2020, High-dimensional index tracking based on the adaptive elastic net, *Quantitative Finance* 20, 1513–1530.

Sim, H. S., H. I. Kim, and J. J. Ahn, 2019, Is deep learning for image recognition applicable to stock market prediction?, *Complexity* 2019.

Simonyan, K., and A. Zisserman, 2014, Very deep convolutional networks for large-scale image recognition, arXiv Preprint arXiv:1409.1556.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, 2017, Attention is all you need, *Advances in Neural Information Processing Systems* 30.

Vrontos, S. D., J. Galakis, and I. D. Vrontos, 2021, Implied volatility directional forecasting: A machine learning approach, *Quantitative Finance* 1–20.

Wang, H., and X. Y. Zhou, 2020, Continuous-time mean-variance portfolio selection: A reinforcement learning framework, *Mathematical Finance* 30, 1273–1308.

Wang, Q., B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, 2020, ECA-Net: Efficient channel attention for deep convolutional neural networks, in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11531–11539.

Wang, Z., and T. Oates, 2015, Imaging time-series to improve classification and imputation, in *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Woo, S., J. Park, J.-Y. Lee, and I. S. Kweon, 2018, CBAM: Convolutional block attention module, in *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19.

Zhang, Q., C. Qin, Y. Zhang, F. Bao, C. Zhang, and P. Liu, 2022, Transformer-based attention network for stock movement prediction, *Expert Systems with Applications* 202, 117239.