Knowledge-Based Systems 248 (2022) 108842

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

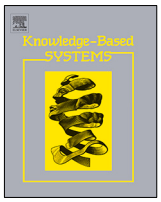

Chart GCN: Learning chart information with a graph convolutional network for stock movement prediction

Shangzhe Li [a,c], Junran Wu [c,*], Xin Jiang [b,d,e,**], Ke Xu [c]

[a] School of Mathematical Science, Beihang University, Beijing 100191, China
[b] LMIB and Institute of Artificial Intelligence, Beihang University, Beijing 100191, China
[c] State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
[d] Zhengzhou Aerotropolis Institute of Artificial Intelligence, Zhengzhou, Henan, 451162, China
[e] Peng Cheng Laboratory, Shenzhen, Guangdong 5180552, China

ARTICLE INFO

Article history:
Received 26 August 2021
Received in revised form 22 March 2022
Accepted 15 April 2022
Available online 25 April 2022

Keywords:
Technical charts and indicators
Graph convolutional network
Stock movement prediction

ABSTRACT

Advanced deep learning methods have been widely adopted in stock movement prediction with technical analysis (TA), while researchers prefer technical indicators to technical charts due to the divergence in quantification difficulty. In traditional TA, researchers usually utilize chart similarity to solve the quantifying problem, while chart similarity is often limited to specific charts as the templates for comparison, resulting in massive inadequate use of information. Accordingly, we propose a novel similarity framework to overcome the limitation of chart similarity to specific charts. Specifically, after extracting the key point sequence (i.e., the draft chart) from the stock price series, we transform it into a graph and ultimately employ an arbitrary graph kernel such as the Weisfeiler-Lehman graph kernel and graph convolutional network (GCN) to sufficiently mine the information in the chart for stock movement prediction. Our similarity framework is more robust than the chart similarity measures commonly used in traditional TA. Additionally, we further evaluate the effectiveness of our framework on real-world stock data and show that our framework achieves the best performance compared to several state-of-the-art baselines in stock movement prediction and obtains the highest average net values in a trading simulation. Our results complement the existing application of the chart similarity method in deep learning and provide support for the investing application of financial market decisions.

1. Introduction

Technical analysis (TA) plays an important role in quantitative investment; in this method, technical indicators and technical charts are adopted to analyze and predict the future movement of stock prices. The core assumption of TA is that all information related to investment decisions is contained in technical indicators and technical charts [1,2]. Recent studies show that many technical analysts have employed very advanced deep learning methods to predict future price movements and design profit strategies [3–5].

In the recent literature employing deep learning with TA [4–6], most technical analysts choose to use technical indicators rather than technical charts, although the effectiveness of technical charts has already been proven in stock price movement prediction without deep learning methods [7–11]. For charts with simple rules, the features generated are sparse Boolean values [12], which is unfavorable to deep learning input features. Additionally, the variation in the time scale and judgment rules of complex charts leads to difficulty in quantifying technical charts in the same way as quantifying technical indicators [13]. For example, the most famous charts, such as 'head-and-shoulders', 'bull flags' or 'rounding bottoms', have dozens of variations [14–16]. In traditional TA, researchers usually use chart similarity to analyze and identify charts [17], which can avoid the need for chart quantification. However, most of the existing studies focus on one or several specific charts, which leads to extensive inadequate use of the information contained in the technical charts [6].

At present, although there are various technical charts, we have not found any evidence or statement supporting that technical charts reported in the research by now are (or better, should be) the only charts. Moreover, the current technical charts known and used by industry and academia are not exhaustive [13]. In the actual stock market, no algorithm can find all chart information

* Corresponding author.
** Corresponding author at: LMIB and Institute of Artificial Intelligence, Beihang University, Beijing 100191, China.
E-mail addresses: wu_junran@buaa.edu.cn (J. Wu), jiangxin@buaa.edu.cn (X. Jiang).

https://doi.org/10.1016/j.knosys.2022.108842
0950-7051/

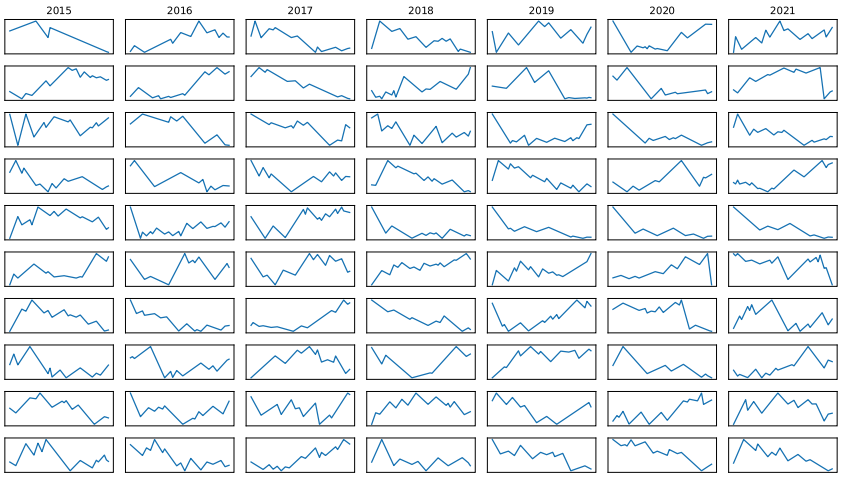

Fig. 1. The 10 best charts in the CSI-300 from 2015 to 2021.

by designing chart judgment rules. Even if some profitable charts are found, they are not necessarily the best. More importantly, these charts cannot be used all the time and may not be applicable after a period of time. To verify this, we rank 60-day price series in the CSI-300 from 2015 to 2021 with their returns in the next 10 days and finally simplify these price series to charts in Fig. 1, according to the method in [1,13], which assessed the top 10 charts every year according to their profitability. These charts are different from each other, and it is difficult to place them into one category or a classical chart, which means that the existing technical charts cannot be utilized to capture all possible regularities in the price path of financial instruments, and important information can be ignored because of the focus on a particular chart [13]. Considering the wide application of the graph similarity algorithm in deep learning [18], chart information can be efficiently captured by transforming charts into graphs [3]. A graph convolutional network (GCN), which is currently very popular, can be used to compare the similarity of transformed graphs [18], thus providing new insights into addressing the insufficient utilization of chart information.

To address the insufficient utilization of technical charts in stock movement prediction, we develop a novel framework to fully exploit the information contained in technical charts. Our framework is essentially a method for measuring chart similarity. We use the perceptually important points (PIP) [1] [19] algorithm to obtain the key point sequence from stock price series, which are also draft charts. Subsequently, the visibility graph (VG) [20] algorithm is applied to convert the extracted key point sequence into a graph. Then, we calculate the similarity between the transformed graphs through an arbitrary graph kernel according to different applications. Similar to other methods for measuring chart similarity, we do not need to quantify the charts. Finally, a GCN kernel is adopted in our framework to sufficiently mine the chart information for stock movement prediction.

Considering that robustness serves as an important criterion in measuring chart similarity methods [21], we first compare our framework with commonly used methods in chart similarity measurement, including dynamic time warping (DTW) and the Pearson correlation coefficient. Our framework achieves the best results under several chart variations. Additionally, our similarity framework does not need standardized preprocessing (e.g., normalization in DTW), which means faster online processing in stock trading. Second, based on our proposed similarity framework, we select the spatial GCN [22] as the last graph kernel to make it possible to predict stock price movement and sufficiently mine the chart information. Moreover, for the purposes of enriching the information contained in the price and meeting the needs of the spatial GCN, technical indicators of stock price are integrated into our framework. On the SZ-50 and CSI-300 stocks in 2018, our framework achieves the best prediction accuracies, 69.26% and 68.62%, compared with several state-of-the-art baselines. Based on the prediction results of our framework, we further design a trading strategy on the SZ-50 and CSI-300 stocks and gain the highest pure values, 1.24 and 1.19, in 2018.

To summarize, the main contributions of this paper are as follows:

- We propose a novel framework to sufficiently utilize technical charts in deep learning for stock movement prediction.
- By comparing the similarity of the graphs transformed from the key point series of charts with an appropriate graph kernel, we overcome the limitation of chart similarity to specific charts involved in chart use in deep learning.
- The similarity part of our framework exceeds the commonly used chart similarity in robustness. Additionally, our framework with GCN surpasses the baseline methods in prediction accuracy and achieves the highest excess return on real stock data.

The rest of this paper is organized as follows: In Section 2, related works and research are presented. Then, in Section 3, the proposed method is presented in detail, followed by the introduction of various utilized variables. In Section 4, the experimental settings are reported. The results are reported in Section 5. In Section 6, we discuss the results, and a conclusion is presented in Section 7.

2. Related work

2.1. Technical chart similarity

Technical charts are usually identified by connecting price points with lines; therefore, technical analysts usually employ pattern matching technology to find technical charts for trading decisions [23]. In stock movement prediction, many studies on pattern matching technology are based on examining the specific charts observed in past stock price charts and determining whether there are similar trends such as 'rounding bottoms', 'bull flags' or 'head-and-shoulders' in the current stock prices [7–11].

[1] For the convenience and simplicity of expression, we use PIP to represent the algorithm and PIPs to represent the result of the algorithm.

However, these methods are subjective, and the charts are difficult to quantify [13]; thus, there is a large body of literature investigating the use of similarity, with measures such as the Pearson correlation coefficient and DTW, to identify current stock price series charts; this is achieved by calculating the similarity between charts recognized in historical data or charts designed by experts and the current stock price series [1]. Nevertheless, the chart time length in TA is not fixed. Some charts may be formed in a few days, while others may span one or two months [13]. The similarity calculated by the Pearson correlation coefficient cannot accurately compare time series of different lengths, resulting in the inability to identify charts of price series with different lengths. Accordingly, some papers proposed calculating the similarity by DTW [24], which is an algorithm for finding the optimal alignment between two given (time-dependent) sequences that can handle unequal time lengths in chart matching. Nevertheless, most articles using DTW are still confined to one or bundles of specific charts [17,25] and susceptible to noise points. Consequently, the PIP algorithm is utilized to calculate the key points in the time-series chart to prevent the influence of noise [26], and DTW is applied to find the most similar sequences in the training set. Additionally, sequences are marked as rising, falling or fluctuating in the testing set [13], which is an attempt to use the information in the chart rather than rigidly identifying a specific chart.

Although the charts vary according to the training set, the charts used are limited in nature, and these methods are hard to utilize with deep learning methods [13]. Graph similarity can mine different structural information in a graph [27], and using a VG to convert a time series into a graph effectively utilizes the structural information in charts [3]; thus, the combination of these strategies overcomes the limitation to specific charts. The VG algorithm is a method that transforms time series into complex network representations and has aroused great interest in recent years [19,20,28,29]. Generally, a VG constitutes a series of geometric and ranking criteria of scalar, real-valued time series and provides a combined representation of the underlying dynamical system [29]. In previous studies, the VG of a time series remains invariant under several time-series transformations and can actually distinguish different types of fractality [20]. Therefore, a promising similarity method with PIP and the VG algorithm can potentially identify various charts in stock price series and inspires us to take advantage of technical charts for stock movement prediction.

2.2. Deep learning in stock movement prediction

Different methods in the field of stock movement prediction can be divided into two categories. The first algorithm attempts to improve the prediction performance by improving the prediction model, while the second algorithm focuses on improving the features used in the prediction.

In the first category of algorithm, researchers use a variety of methods, including naive Bayes classifiers, support vector machines (SVMs), recurrent neural networks (RNNs) and their variants [30–33]. The second category of algorithms focuses on improving the features. A network of fund stocks for obtaining the fund manager's preference for stocks was proposed, where this preference information is fused with the stock price indicator to obtain a stronger index [5]. Structured events can be extracted from financial news to obtain event embedding and then combined with price values to forecast stock trends [34]. Multiscale stock price information has been obtained through a convolutional neural network (CNN) and wavelet transform [6].

In the two categories of methods, few articles employ technical charts with deep learning for stock prediction for the reason mentioned above. Recently, the GCN has emerged as a powerful method for measuring graph similarity [18,22,35], and the work of [36–38] successfully addressed key problems in dynamical graph learning and achieved fantastic and valuable results. The way a spatial GCN processes graphs is similar to the way a CNN processes images [22]. In some papers on CNN visualization, the CNN can usually extract the local image features. For example, for an image of a dog, a CNN can recognize the dog's head and feet and then obtain a more advanced feature representation by further convolution of the local features. Inspired by this, we apply a graph convolution mask to the subgraphs of the transforming series graph (corresponding to the local image block). The combination of the distant price points mined from the transformed graph can form a low-level chart, such as the left shoulder of the head–shoulder chart. By further convoluting the convolution output, we can obtain a higher-level chart, such as the complete head–shoulder mode.

Therefore, we can utilize a spatial GCN as the final component of our framework to make full use of the extracted chart information in stock price movement prediction.

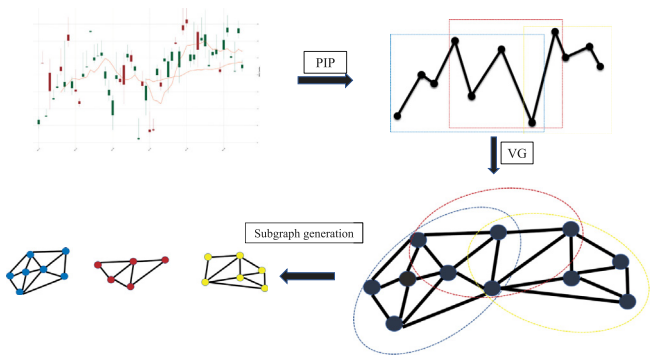

Fig. 2. An illustration of how our new chart similarity measurement could address the challenge of utilizing charts.

3. Method

In this paper, we present a novel framework that can address the main issue that limits the application of technical charts. Specifically, first, we propose a novel chart similarity method that can identify charts more robustly. The method consists of PIP, VG and arbitrary graph kernel. After extracting the key point sequence from a stock price series with PIP [13], we transform it into a graph by the VG method. Then, we can use an arbitrary graph kernel to measure the similarity of the transformed series graph. According to different application requirements, we can choose the final components of our similarity framework as different graph kernels, such as the Weisfeiler-Lehman (WL) graph kernel and GCN. To sufficiently mine the chart information, we propose a novel method, the chart spatial graph convolutional network (Chart GCN), to predict stock movement based on our similarity framework.

An illustration of how our new chart similarity measurement could address the challenge of utilizing charts is shown in Fig. 2. Given a stock price series, our new similarity method can remove the influence of noise points through PIP, avoid the influence of chart size and different judgment criteria by turning price series into graphs (the example of the above two advances can be seen in Section 3.1) and finally avoid direct comparison with a chart described by a specific rule through some graph kernel. Different from the previous method, which can only identify a specific chart, our method can recognize all three different charts contained in this series and then obtain a more advanced feature

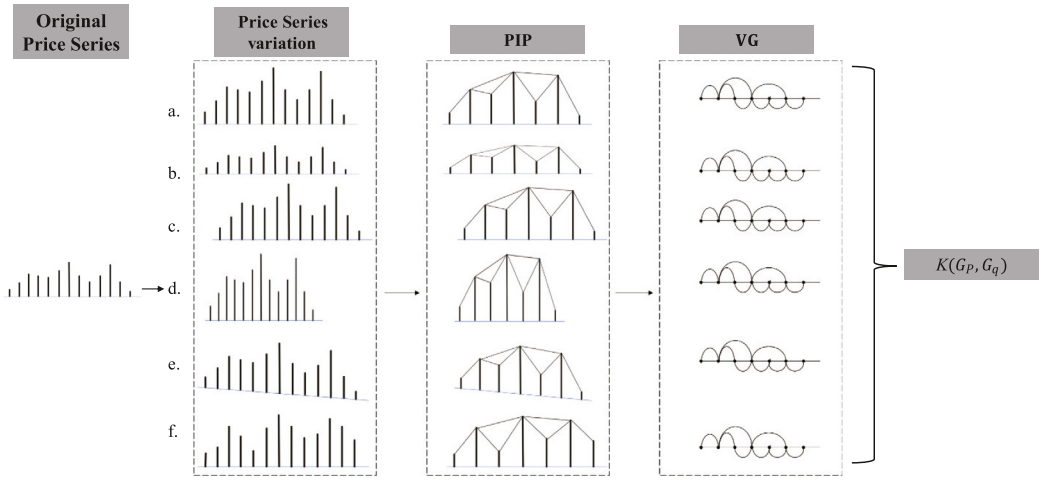

Fig. 3. The multistage design of the proposed similarity method and its superiority. The most important criterion in the evaluation of the chart similarity method is robustness, that is, whether the corresponding charts can still be identified under variations. The variations are shown in figure: (a.) Amplitude scaling, (b.) amplitude shift, (c.) time shift, (d.) time scaling, (e.) rotation, and (f.) WGN added. Therefore, we compare the robustness of our framework with the two most commonly used chart similarity methods on these six variations. The formulas of the variations and specific results are described in detail in the results section.

representation by further convolving the local features in the spatial GCN. In more detail, the normalized subgraph combined with the distant price points mined from the transformed graph can form a low-level chart, such as the left shoulder of the head–shoulder chart. By further convolving the convolution output, we can obtain a higher-level chart, such as the complete head–shoulder mode. Therefore, our method can deeply and fully mine the price series chart information.

3.1. Chart similarity

In this section, we propose a novel chart similarity measurement method based on PIP, VG and arbitrary graph kernel. Fig. 3 illustrates the multistage design of the proposed similarity method and its superior robustness.

3.1.1. PIP

Charts are characterized by a set of key points; for example, the head-and-shoulders chart is defined by a head point, two points for the shoulders and two more points for the neckline. These key points are the most relevant to the shape of the chart, and the other points are noise in the chart identification [39]. Accordingly, we obtain the key point set that is the draft chart and important for chart recognition based on PIP; this is a promising method for identifying the salient points and filtering noise points on a time series [26,39]. The PIPs calculation is described in Algorithm 1.

Formally, let $\{c_1, c_2, \ldots, c_n\}$ be the price series C_n of length n and $\{t_1, t_2, \ldots, t_n\}$ be the corresponding time stamp. We can obtain two draft PIPs $x_1 = (t_1, c_1)$ and $x_n = (t_n, c_n)$. The Euclidean distance d_E of each intermediate point $x_k = (t_k, c_k)$, $t_k \in \{t_2, \ldots, t_{n-1}\}$ to the two draft PIPs is defined as

$$d_E(x_k, x_1, x_n) = \sqrt{(t_k - t_1)^2 + (c_k - c_1)^2} + \sqrt{(t_n - t_k)^2 + (c_n - c_k)^2}. \quad (1)$$

The new PIPs, $x_k^* = \left(t_{k^*}, c_k^*\right)$, maximizes the distance d_E at t_{k^*},

$$k^* = \arg\max_k \left(d(x_k, x_1, x_n)\right), \quad (2)$$

where 'argmax' denotes the maximum function argument. If we set the user number of PIPs to $m < n$, we extract a key point sequence $\{x_1, x_{k^*}, \ldots, x_n\}$ of length m from C_n, denoted K_m.

Specifically, the fewer points we need, the more representative the points are. We define the chart importance score to describe the importance of points for identifying charts in time series according to the maximum distance from adjacent points when added to PIPs. In particular, the start point and stop point are reserved at the first step; therefore, their importance score is the highest. The pseudocode for calculating PIPs and the chart importance score is described in Algorithm 1.

Algorithm 1 Calculating PIPs and the chart importance score series

Input: Closing price series $X_i = (x_i, \ldots, x_{i+T})$, PIPs number m
Output: PIPs, chart importance score series $IS_i = (s_i, \ldots, s_{i+T})$

for $p = i \to i + T$ **do**
  $s_p = 0$
end for
add x_i, x_{i+T} to PIPs
m^* = the number of PIPs; denote PIPs timestamps as $p_1, \ldots, p_{m^*}$
while $m^* < m$ **do**
  for $p = i \to i + T$ **do**
    initial maximum distance $d_{max} = 0$
    for $j = 2 \to m^*$ **do**
      $k = j - 1$
      if $p > p_k$ and $p < p_j$ **then**
        p point distance $d_p = d_E(x_k, x_i, x_{i+T})$
        if $d_p > d_{max}$ **then**
          Candidate PIPs timestamp $p^* = p, d_{max} = d_p$
        end if
      end if
    end for
  end for
  add x_{p^*} to PIPs
  add p^* to PIPs timestamps
  $s_{p^*} = d_{max}$
end while
$s_i, s_{i+T} = max(IS_i) + 1$
return IS_i

3.1.2. VG algorithm

In this paper, we employ the VG algorithm to transform the key point sequences into graphs. In previous studies, the VG of a time series remains invariant under several time-series

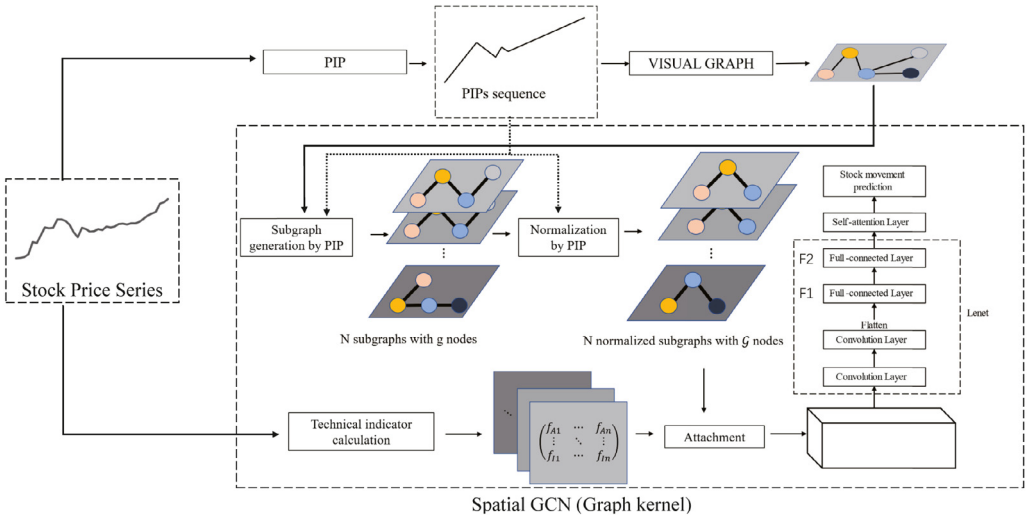

Fig. 4. Chart GCN detailed framework.

transformations and can actually distinguish different types of graphs [20]; in other words, the VG algorithm can identify charts and is robust against different variations. We first take all the data points in the key point sequence as the vertices of the transformed graph. Then, the connection between vertices is determined according to the following rules: we plot the vertical bar chart of the new time series according to the corresponding price. If we can draw a straight line that connects any two data points, provided that this 'visibility line' does not intersect any intermediate price height, we add an edge between the vertices corresponding to these two data points to the graph. More formally, a VG is obtained from a time series according to the following visibility criterion [5]: two arbitrary data points (t_a^*, c_a^*) and (t_b^*, c_b^*) where $t_a^*, t_b^* > 0$ in the time series have visibility and consequently become two vertices in the associated graph if any other data point (t_c^*, c_c^*) such that $t_a^* < t_c^* < t_b^*$ fulfills

$$c_c^* < c_a^* + \left(c_b^* - c_a^*\right) \frac{t_c^* - t_a^*}{t_b^* - t_a^*}. \tag{3}$$

3.1.3. Graph kernel

After we convert key point sequence K_m into graph G, we can calculate the similarity S between G_p and G_q with an arbitrary graph kernel $\mathcal{K}$ as in [40]:

$$S\left(G_p, G_q\right) = \mathcal{K}\left(G_p, G_q\right). \tag{4}$$

Due to the ability of the PIP algorithm to filter noise information, the ability of VG to distinguish different types of graphs, and the invariability under several transformations of the transformed series graph, we can identify various charts and completely utilize the chart information in the transformed series graph. Common charts usually have many variations, and these variations can be roughly classified into six types, as shown in Fig. 3 [11,13,16].

3.2. Stock movement prediction

In this section, we introduce our framework with GCN in detail. In the last step of our similarity framework, various graph kernels can be utilized for different applications. To sufficiently mine the chart information and address the combination with deep learning to predict stock price movement, we propose the novel framework of the Chart GCN based on our similarity and spatial GCN that not only captures nonconsecutive and long-distance information that is more consistent with the chart characteristics [41] but is also a good measure of the isomorphic similarity between graphs [22]. Therefore, our design can address the proposed problem fully inheriting the advantages of our similarity framework. We refer to [22] for the spatial GCN architecture. To assign nodes from two different graphs to a similar relative position if and only if their structural roles within the graphs are similar [22], a spatial GCN utilizes the betweenness centrality of nodes as the judgment in the receptive field point selection step and subgraph normalization, but betweenness centrality does not help identify charts in our framework. Accordingly, we change the judgment criteria for the importance of points from the betweenness centrality in the graph to the chart importance score. We show our framework in Fig. 4. The PIP and VG steps are the same as those in the similarity method. The main difference is that we input a 3D feature matrix into the convolution layer after attaching the technical indicator to the PIPs.

Given a transformed series graph, we use the following two steps to select all subgraphs in a complete graph and normalize them to create the receptive field of each PIPs and to have a consistent convolution method.

3.2.1. The generation of subgraphs

We sort all nodes in a graph based on their importance score. If the importance scores are the same, we sort the nodes by their degrees (number of neighborhoods of a node). If the degrees are the same, we sort them with the mean importance score of their neighbors. In this way, we select the top N nodes that are the most important for chart recognition through the similarity algorithm proposed previously. After selecting the nodes, we expand the subgraph for each selected node by the breadth-first search algorithm according to the preset minimum subgraph size g. If the size of the current depth subgraph is insufficient, we expand one layer to add more nodes and then check whether the subgraph size meets the requirements. After the subgraph size is satisfied, we obtain N subgraphs, each of which contains at most g nodes. For example, in the middle of Fig. 4, we generate N subgraphs.

3.2.2. The normalization of the subgraphs

Given a subgraph, we need a convolution mask to convolve the node order to make the node labels consistent with all timestamps and all subgraphs. We define the optimal label as follows. Let us assume that the subgraphs SG_i and SG_j contain g nodes.

Given the labeling s of subgraph SG, an adjacency matrix $A^s(SG)$ can be constructed. Then, we can define an optimal label as

$$s^* = \arg\min_s E\left[D_{\mathrm{A}}\left(\mathrm{A}^s(SG_i), \mathrm{A}^s\left(SG_j\right)\right) - D_G\left(SG_i, SG_j\right)\right], \tag{5}$$

where $D_{\mathrm{A}}(\cdot,\cdot)$ is the distance measure of two matrices, such as $\left\|\mathrm{A}-\mathrm{A}'\right\|_{l1}$, and $D_{\mathrm{G}}(\cdot,\cdot)$ is the distance measure of two graphs. Nevertheless, such labeling is NP-hard; thus, we follow [22] to obtain alternative labeling. First, we obtain a subgraph node as the root node in the above step. Then, we use the breadth-first search to sort the nodes based on depth. Next, in the same tier (depth) of the graph-based spanning tree, the nodes are sorted by the chart importance score. Finally, if two nodes in the same tier have the same score, we use the node degree to sort the tier. After this step, we obtain $\mathcal{G}$ nodes for each subgraph. By calculating the rank of the subgraph adjacency matrix in the previous step, we add dummy nodes that have no connection with any points in the graph to the subgraphs whose number of nodes number is fewer than $\mathcal{G}$ to meet the number of nodes requirement and filter out the subgraphs whose number of nodes is more than $\mathcal{G}$. We can easily see that the normalization applied to the subgraphs is consistent with the way a CNN's first layer is applied to images, which can extract the local features of the image. Accordingly, normalization can also extract a low-level chart from the transformed series graph. An example of graphic normalization is shown in the middle of Fig. 4.

To enrich the price information and meet the needs of the spatial GCN, instead of representing each node in the subgraph as an ID, we use a technical indicator to attach to each point in the subgraph. The selection and calculation of technical indicators are described in detail in the data section. For N subgraphs, there are $\mathcal{G}$ points in each graph. For each point, we attach the F technical indicators of the corresponding points; thus, we can obtain the 3D feature X of size $N \times \mathcal{G} \times F$.

3.2.3. Prediction architecture

The prediction architecture is designed with multiple convolution layers following LeNet [42]. Here, we use a typical configuration to illustrate the architecture, shown in the right panel of Fig. 4.

The 3D feature X of size $N \times \mathcal{G} \times F$ is used as the input of the first convolution layer, where N is the number of selected and normalized subgraphs, $\mathcal{G}$ is the size of the receptive field (the size of the normalized subgraphs), and F is the dimension of the features. To combine the charts of each input subgraph into a higher-level chart representation, we use a convolution kernel of $\mathcal{G} \times F$ to convolve the input tensor. Then, we convolve all N input subgraphs using the k_1 kernels to generate an $N \times k_1$ matrix in the same way. Then, we use a kernel of 5×1 to convolve N subgraphs to obtain a higher level of chart information (a combination of chart information in subgraphs). Here, we use k_2 5×1 kernels to generate the k_2 dimension. Thus, we have a $k_1 \times k_2$ matrix. Thus far, we successfully convolved different dimensions of features, different points in each subgraph and different subgraphs in the charts. Then, we can further convolve a higher-level chart. Accordingly, we use k_3 kernels to generate a $k_2 \times k_3$ matrix. We add two fully connected layers F_1, F_2 after the second convolution layer, as in LeNet. In the whole convolution layer and fully connected layer, the rectified linear unit (ReLU) activation function is used to avoid overfitting. The LeNet layer calculation can be simply expressed as:

$$l = f_{LeNet}(X). \tag{6}$$

To explore the inner connection between stocks [43,44] and to perform better classification, we add a self-attention layer between S samples. As shown in the right panel of Fig. 4, we have a self-attention layer for S samples of l.

As the first self-attention step, $l \in R^{S \times F2}$ formed by S samples of l is first transformed to three feature spaces (Q, K, and V) using Eq. (7).

$$Q(l) = l \cdot W_Q, \quad K(l) = l \cdot W_K, \quad V(l) = l \cdot W_V, \tag{7}$$

where $W_Q, W_K \in R^{F2 \times k_a}$ and $W_V \in R^{F2 \times k_v}$. The channel numbers k_a for $Q(l)$ and $K(l)$ need to be the same. The number of channels k_v for $V(l)$ can be different from k_a. Here, $Q(l) \in R^{S \times k_a}$ is generally called the query space, $K(l) \in R^{S \times k_a}$ is called the key space, and $V(l) \in R^{S \times k_v}$ is called the value space. The three feature spaces are named following the convention of [45], where the attention captures the mapping relationships between a possible query and the key–value pairs in the data.

In the second self-attention step, the self-attention for one variable, denoted $\alpha \in R^{S \times S}$, is calculated using the features in the query space and key space by two operations shown in Eqs. (7) and (8).

$$S = Q(l) \cdot K(l)^T. \tag{8}$$

The softmax function is applied to $S \in R^{S \times S}$ to normalize the attention, as shown in Eq. (9).

$$\alpha = softmax(S). \tag{9}$$

In the third self-attention step, the normalized attention α is applied to the features in the value space $V(l)$. This step obtains the attention hidden states $V_a = \alpha \cdot V(l)$, where $V_a \in R^{S \times k_v}$.

The last layer after the self-attention layer is further connected to a fully connected layer that corresponds to the movement prediction of the stock price. Given a set of M labeled graphs transformed from stock price series and $\{m = 1, \ldots, M\}$, our model utilizes cross entropy to measure model loss between the real label distribution y_n and the forecast distribution $\hat{y}_n$:

$$J = -\frac{1}{N}\sum_{n=1}^{N} y_n \log\left(\hat{y}_n\right). \tag{10}$$

4. Experiments

4.1. Data

The datasets collected in this paper contain daily quote data of component stocks in the SZ-50 and CSI-300 indices, which include most of the important stocks in the Chinese A-share market. We collect the daily prices (including the opening prices, high prices, low prices, closing prices, volumes and amounts) from 2010 to 2018 and calculate nine classes of trading indicators [46], as shown in Table 1, for the two indices. The source of our data is Tushare, a free, open source interface package of Python financial data supported by Sina Finance, Tencent Finance, the Shanghai Stock Exchange and the Shenzhen Stock Exchange.

4.2. Comparison method

Due to the popularity of deep neural networks, many recent studies on stock forecasting have focused on using long short-term memory (LSTM) [31], CNNs [4] and attention-based methods [47,48] with market price or index data as their features and have ultimately obtained competitive performance. Please note that none of these state-of-the-art prediction models, to the best of our knowledge, have taken advantage of technical charts extracted from price time series. To verify the effectiveness of our framework for investigating technical charts, we compare our proposed framework with the following baselines:

- **CNN:** the first baseline algorithm is a CNN with two-dimensional input [4].

Table 1
The trading indicator formulas.

Indicator name	Formula
Simple n-day moving average	$\frac{c_t+c_{t-1+\ldots}+c_{t-n+1}}{n}$
Weighted n-day moving average	$\frac{(n)\times c_1+(n-1)\times c_{t-1}+\cdots+c_{t-n+1}}{n+(n-1)+\cdots+1}$
Momentum	$c_t - c_{t-n+1}$
Moving Average Convergence Divergence (MACD)	$\mathrm{MACD}(n)_{t-1} + \frac{2}{n+1} \times (DIFF_t - \mathrm{MACD}(n)_{t-1})$
Larry Williams R%	$\frac{h_t-c_t}{h_t-l_t} \times 100$
Commodity Channel Index (CCI)	$\frac{M_t-SM_t}{0.015\times D_t}$
Stochastic K%	$\frac{c_t-ll_{t-(n-1)}}{hh_{t-(n-1)}-ll_{t-(n-1)}} \times 100$
Stochastic D%	$\frac{\sum_{i=0}^{n-1} K_{t-i}}{10}\%$
Relative Strength Index (RSI)	$100 - \frac{100}{1+\left(\sum_{i=0}^{n-1} UP_{t-i}/n\right)/\left(\sum_{i=0}^{n-1} DW_{t-i}/n\right)}$

c_t is the closing price, l_t is the low price and h_t is the high price at time t; $DIFF_t = EMA(12)_t - EMA(26)_t$, EMA is the exponential moving average, and $EMA(k)_t = EMA(k)_{t-1} + \alpha \times (c_t - EMA(k)_{t-1})$, α is a smoothing factor that is equal to $\frac{2}{k1}$, k is the time period of the k-day exponential moving average, ll_t and hh_t are the lowest low price and highest high price in the last t days, respectively. $M_t = \frac{h_t+l_t+c_t}{3}$, $SM_t = \frac{(\sum_{i-1}^{n} M_{t-i+1})}{n}$, and $D_t = \frac{(\sum_{i-1}^{n} |M_{t-i+1} - SM_t|)}{n}$. UP_t is the upward price change and DW_t is the downward price change at time t. n represents the time windows of all indicators.

- **LSTM:** the second baseline algorithm is a basic LSTM network for predicting the future trends of stock prices based on historical price data [31].
- **TCN:** the third baseline algorithm is a temporal convolution network (TCN) that includes multilayer convolution on the timestamp dimension [49].
- **DARNN:** the fourth baseline algorithm is a dual-stage attention-based recurrent neural network (DARNN) that can select relevant deriving series as well as temporal features and achieve time-series prediction performance superior to that of LSTM and attention-based LSTM [47].
- **CA-FSCN:** the last baseline is the newest method with a fully convolutional network (FCN) incorporating cross attention. In the cross attention stabilized fully convolutional neural network (CA-FSCN), dual-stage attention on the variable and timestamp dimensions is used to select the relevant variables at each time step [48].

In particular, all five baseline methods remain consistent with the original model and use technical indicator data as input to show the effectiveness of our framework in stock movement prediction with the actual data.

4.3. Parameter setting

To prevent data snooping, we strictly divide the training/validation/test set according to the timestamp of the sample data. In particular, we take the data from January 2010 to December 2017 as the training set and validation set and the rest of the data, which is the whole year of 2018, as the test set. To make full use of historical information, we randomly select 80% of the samples as the training set and the remaining 20% of the samples as the verification set in the training process. The statistics of the SZ-50 and CSI-300 indices datasets are presented in Table 2. Note that the proportion of the two trends in both the training and testing datasets is almost equal. For the purpose of actual trading, we choose the closing price to perform graph transformation. The target of our model is defined as follows:

$$y = \begin{cases} 1 & \text{if } c_{t+1} > c_t, \\ 0 & \text{else .} \end{cases} \tag{11}$$

In this paper, a grid search is used to select the optimal hyperparameters in terms of the best performance for all methods. Specifically, we tune the time windows n of all indicators within {100, 120, 130, 140}, the minimum number of PIPs within {30, 40, 60, 80}, the width of the top nodes N equal to the average number of nodes within {10, 15, 20, 25} [22], and the minimum subgraph size $\mathcal{G}$ within {3, 4, 5, 6}. We set $F_1 = 84$ and $F_2 = 32$ for the two fully connected layers [41]. All baselines mentioned above utilize the same empirically set common parameters; for example, the regularization weight decay = 0.00005 and the batch size = 128.

Table 2
Summary statistics of the SZ-50 and CSI-300 indices for the training, validation and test datasets.

Dataset		Training		Validation		Testing	
		(%)	Samples	(%)	Samples	(%)	Samples
SZ-50	↓	51.04	85,767	50.92	14,426	51.84	10,837
	↑	48.96		49.07		48.16	
CSI-300	↓	50.34	543,262	50.27	82,764	50.07	68,710
	↑	49.66		49.74		49.93	

5. Results

To verify the robustness of our proposed framework, we choose the WL graph kernel as the last part of our framework and compare it with the two most commonly used methods in chart similarity measuring. Then, employing a spatial GCN, we inspect our framework in stock movement prediction and share trading on real-world stock data to further verify the effectiveness of our framework.

5.1. The robustness of our framework in similarity measurement

To verify the robustness of our similarity method, the Pearson correlation coefficient, DTW distance and DTW distance without normalization are compared when several levels of small variations in time scaling and shift, amplitude scaling and shift, rotation and additive Gaussian noise, as in [21], are added to the close-price time series. We focus on the performance of the similarity methods in terms of their insensibility to small data variations. The time-series variations are defined as follows:

a. **Amplitude scaling:** $X_{AS}(t) = \beta * X(t)$
b. **Amplitude shift:** $X_{At}(t) = X(t) + \beta$
c. **Time scaling:** $X_{Ts}(t) = X(\beta * t)$
d. **Time shift:** $X_{Tt}(t) = X(t + \beta)$
e. **Rotation:** $X_B(t) = \mathrm{rotate}(X(t), \theta)$
f. **WGN added:** $X_{WGN}(t) = X(t) + Z_i \sim \mathcal{N}(X_i, N)$

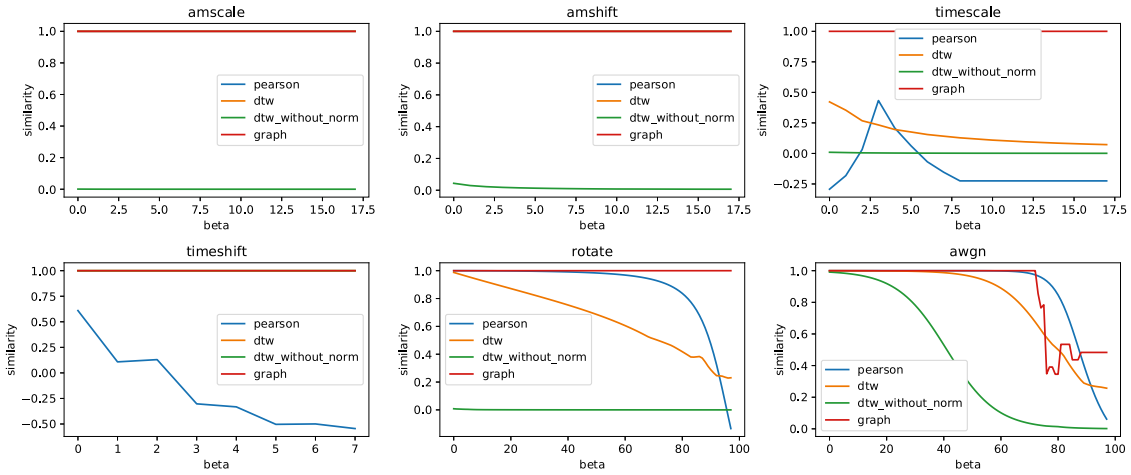

Fig. 5. Comparison of different similarity methods on different variations.

The corresponding diagrams of different variations are also shown in the corresponding serial numbers of Fig. 3, where β is a constant that represents the scaling or shift magnitude, θ is the rotation angle and WGN is white Gaussian noise. We perform 20–100 small incremental changes on β and θ values, whose initial values are experimentally determined. The horizontal ordinate of the figures presented in Fig. 5 indicates these 20–100 degrees of variation for each variation type (from (a) to (f)), and the vertical ordinate represents the similarity between the variation and the original time series. We select the stock closing price in the CSI-300 from 2015 to 2018 and take 20 trading days (almost one month) as an interval for intercepting the price series. Their similarity is calculated with the original price series under different degrees of variation. The average value of all intercepted price series under a certain degree of variation is displayed in Fig. 5. The larger the similarity is, the less the variation affects the similarity method.

As seen in Fig. 5, the results demonstrate that our similarity method is the most robust. Although our method deteriorates faster than the Pearson correlation coefficient method when WGN is added, the Pearson correlation coefficient method performs very poorly when the time scale changes or moves, while our method and the DTW method both perform very well and can even be unaffected. In particular, the similarity of our method remains at the highest level under high-level WGN variation, and it is stable. These results show the superiority of our method under the important robustness index, but note that the charts in real data often have multiple simultaneous variations; thus, we need to use real data to verify our framework.

5.2. The effectiveness of our framework on real stock data

To further verify the overall performance of our framework on real stock data, we compare our framework with six state-of-the-art stock movement prediction baselines. The overall performance in precision, accuracy and F1 score is shown in Table 3. Because of the stock trading characteristics, we can make profits as long as we ensure that every trading behavior is accurate. Obviously, from the perspective of all metrics, the performance of our proposed model based on chart information is the best on the two datasets. The performance on different datasets can reflect the different advantages of our model.

Note that the performance of the CA-FSCN and our model on SZ-50 differs more than that on the CSI-300. One possible explanation is that compared with the CSI-300, the SZ-50 has fewer stocks and more representative charts, which makes our method advantageous in distinguishing different charts by similarity. Specifically, compared with the CA-FSCN, the Chart GCN further considers the information contained in the technical chart. The Chart GCN can capture more complex information in stock price movements and is more suitable for chart recognition.

The precision of all methods on negative cases is higher than that on positive cases. This is because it is impossible to short in China's stock market, resulting in arbitrage space when stocks fall. According to the efficient market hypothesis [50], this is reasonable.

6. Discussion

6.1. Ablation

To further verify the effectiveness of the Chart GCN on real stock data, we compare our method with different variants. **Chart GCN-1:** this model excludes the chart importance selection component; in other words, we use node betweenness to sort nodes of the selected subgraph in normalization, which is equivalent to normalization in a spatial GCN. **Chart GCN-2:** The Chart GCN has no self-attention mechanism but a persistent chart importance selection component. **Chart GCN-3:** The Chart GCN has neither a chart importance selection component nor a self-attention mechanism. **Chart GCN-4:** The Chart GCN uses the original price as an attachment to each point in the subgraph. We show the precision results in Table 4 and obtain the following findings.

Chart GCN-3 is consistently inferior to Chart GCN-2. This illustrates the importance of the accurate extraction and explicit modeling of chart information, which suggests that the chart importance selection component is more capable of capturing chart information than the vanilla spatial GCN model. Our model performs better than Chart GCN-3 and Chart GCN-2, possibly because the chart importance selection component and self-attention mechanism can mutually reinforce each other: the chart importance selection component can be used to capture complex chart information, while dot-product attention can distinguish important historical technical indicator information.

The performance of Chart GCN-2 on SZ-50 is very good, but the performance on the CSI-300 is average because the charts in the stock price of SZ-50 are more representative while the charts in the CSI-300 contain more noise; thus, there is a large gap between the performance of Chart GCN-2 on the two indices.

Our model performs less than 2% better than Chart GCN-4 on the SZ-50 and CSI-300. This result verifies that the improvement of the technical indicators in the model is limited compared

Table 3
Results (%): our proposed framework and baselines. All models predict price trend labels at the next time step.

	SZ-50					CSI-300				
	Acc (%)	Pre_1 (%)	Pre_0 (%)	F1_1 (%)	F1_0 (%)	Acc (%)	Pre_1 (%)	Pre_0 (%)	F1_1 (%)	F1_0 (%)
CNN	54.78	50.09	58.85	50.72	58.22	54.54	49.84	58.62	50.44	58.02
LSTM	54.64	49.95	58.63	50.26	58.32	54.89	50.22	58.89	50.65	58.47
TCN	56.11	51.55	59.96	51.81	59.71	55.48	50.84	59.48	51.44	58.90
GRU	58.03	53.54	62.02	54.57	60.99	61.13	56.83	64.95	57.93	63.87
DARNN	60.68	56.42	64.41	57.24	63.62	61.93	57.80	65.52	58.50	64.85
CA-FSCN	63.03	58.81	66.82	60.09	65.58	64.74	60.76	68.23	61.67	67.36
Chart GCN	**69.26**	**65.76**	**72.26**	**66.40**	**71.67**	**68.62**	**64.87**	**71.91**	**65.93**	**70.91**

Table 4
Results (%): Ablation analysis, accuracy, precision and F1-score of each model variant (%).

	SZ-50					CSI-300				
	Acc (%)	Pre_1 (%)	Pre_0 (%)	F1_1 (%)	F1_0 (%)	Acc (%)	Pre_1 (%)	Pre_0 (%)	F1_1 (%)	F1_0 (%)
Chart GCN-1	58.69	54.27	62.57	55.13	61.72	57.66	53.23	61.42	53.57	61.09
Chart GCN-2	64.59	60.75	67.88	61.24	67.41	58.21	53.83	61.92	54.17	61.60
Chart GCN-3	54.35	49.63	58.39	50.04	57.98	54.94	50.27	58.98	50.84	58.42
Chart GCN-4	66.58	62.87	69.76	63.46	69.20	65.58	61.69	68.98	62.54	68.17
Chart GCN	**69.26**	**65.76**	**72.26**	**66.40**	**71.67**	**68.62**	**64.87**	**71.91**	**65.93**	**70.91**

with the original price. Furthermore, this shows that our model can effectively extract the chart information from the price and integrate it with the technical indicator.

6.2. Further illustration of the advantages of our framework

To further illustrate the advantages of our framework in chart information mining and utilization, we replaced the kernel part of our framework (spatial GCN) with the baseline models mentioned above. The results are shown in Table 5. We can see that our model improved for the CNN, TCN and CA-FSCN models but not for LSTM, GRU and DARNN. This may be because these models can only handle sequential data and lack the capability of capturing chart information in converted graphs. Nevertheless, CNN, TCN, CA-FSCN and spatial GCN models (the model we selected) can obtain a more advanced feature representation by further convolution of the local features when these models are combined with our similarity framework.

The most common and feasible method for utilizing chart information in deep learning is identifying charts as Boolean values and inputting them together with technical indicators. To explicitly show the disharmony between deep learning models and Boolean values of charts, we conduct an experiment with Boolean values of 61 charts [2] as model inputs. It can be seen in Table 5 that the performance of all models worsens after adding the Boolean values of the charts as features. The main reason for this result is that chart features are very sparse. As shown in Fig. 6, where we count the average annual rate of 61 charts in the CSI-300 from 2015 to 2021, the proportion of most charts is less than 1%. Additionally, the complex chart also has the problem of judgment standards and sizes, which further leads to difficulty in quantifying technical charts in the same way as quantifying technical indicators [13].

6.3. Trading simulation

Although the transaction cost affects the final profit, the relative position based on model profit does not change owing to the same trading strategy. We also calculate the average return on the component stocks of the SZ-50 and CSI-300 by evenly holding every stock as the benchmarks, indicating the overall market trend. Due to regulations in the Chinese stock market,

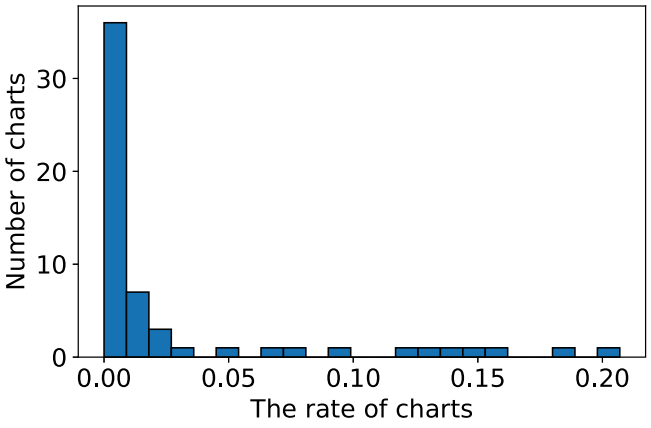

Fig. 6. Rate of charts in the CSI-300.

one can only execute long operations on the stocks. Specifically, the trading signals can be generated as follows: for the long signal, once our model prediction is positive, we will hold a long position before the close of the day. However, it should be noted that if there are previous long positions, we will not operate. Similarly, for the short signal, once our model prediction is negative, we will sell a long position before the close of the day. However, if there are previous short positions, we will not operate. To avoid fluctuation risk, another technical indicator, moving average convergence divergence (MACD), is also used. If the MACD value of the stock decreases than 0, the long-selling operation is also executed. Details of the models' net value curves during simulations are shown in Fig. 7. Our proposed framework can gain the best profit results of all the baseline methods, even when the market exhibited a downturn in 2018.

To further verify the prediction performance of our model on the actual data, we collect the minute price data of Rebar futures rb2010 in the Shanghai Stock Exchange [3] and take the data from 10 February 2020 to 2 April 2020 as the training set and validation set and the rest of the data from 3 April 2020 to 13 April 2020 as the test set. The trading strategy is similar to the strategy on stocks above. As seen in Fig. 8, our model can also achieve good returns on futures transactions.

[2] "https://www.ta-lib.org/".

[3] "https://www.myquant.cn/gm2/".

Table 5
Accuracies (%) of baselines with chart Boolean or similarity.

	SZ-50			CSI-300		
	Technical indicator (%)	+ Chart Boolean (%)	+ Chart similarity (%)	Technical indicator (%)	+ Chart Boolean (%)	+ Chart similarity (%)
CNN	54.78	48.21	**57.77**	54.54	50.41	**56.38**
LSTM	54.64	50.38	53.35	54.89	51.08	54.47
TCN	56.11	50.14	52.26	55.48	51.27	**56.00**
GRU	58.03	52.12	**58.33**	61.13	53.54	60.17
DARNN	60.68	52.47	53.23	61.93	57.80	**65.52**
CA-FSCN	63.03	54.84	**65.09**	64.74	56.55	**66.87**

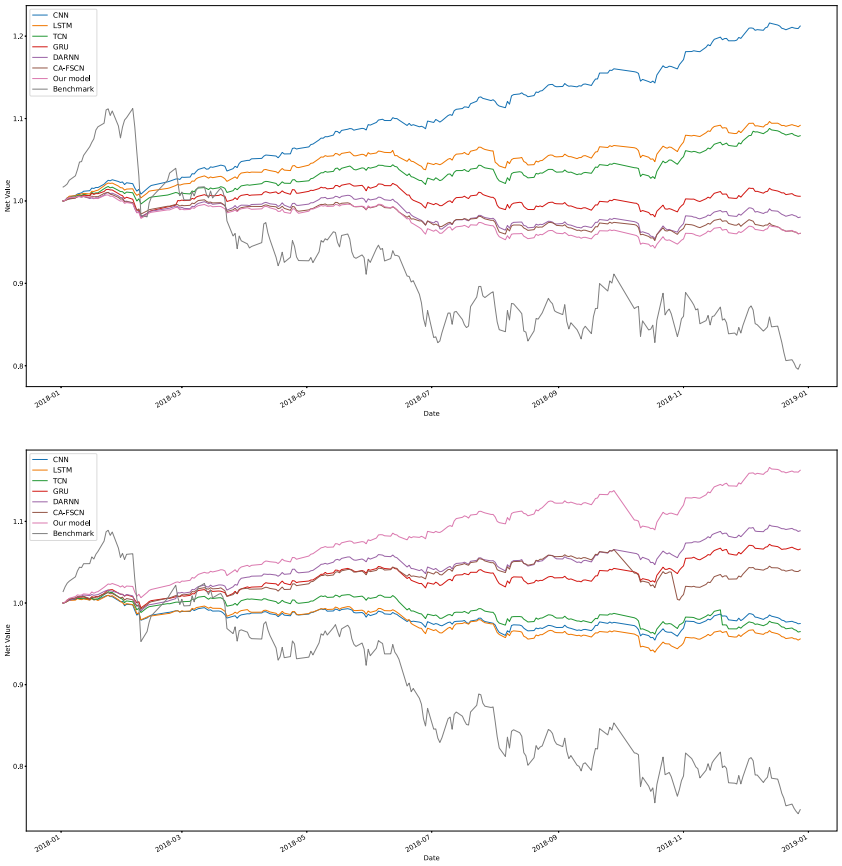

Fig. 7. The backtest results. **upper panel**: SZ-50 **lower panel**: CSI-300.

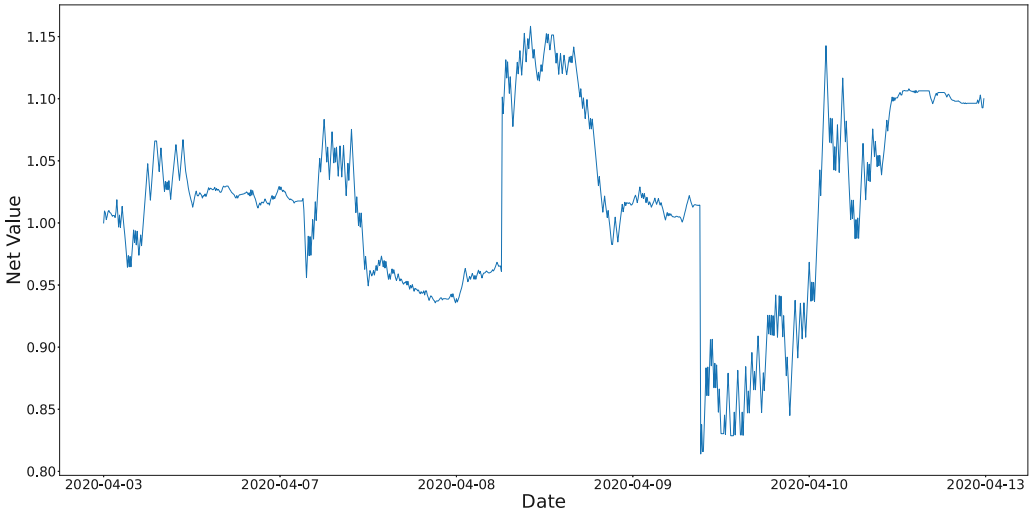

Fig. 8. The backtest result of futures transactions.

In summary, all these results indicate that our framework can effectively utilize charts. Moreover, the learned prediction model can help investors make more accurate trading decisions.

7. Conclusion

In this paper, we propose a novel framework to sufficiently utilize the information from technical charts for stock movement prediction. Compared with the chart similarity in traditional TA, our framework is more robust. Based on our framework, we can effectively utilize charts through the transformed graph to solve the basic stock movement prediction problems where charts are rarely employed by machine learning models. Through this framework, a deep learning model can make use of information contained in various charts rather than a couple of specific charts and has good performance in practical applications in stock forecasting and trading. The results of this paper highlight the role of charts in stock movement prediction. Compared with the use of original financial information such as market price data, the chart information of financial market series is well-matched with market price data on financial market tasks.

Despite the valid performance of our Chart GCN, there are some limitations to this research. For example, there may be multiple charts within a period of time, and the information between these charts may be mutual and conflicting. In addition, we used only one class of GCN model as the graph kernel. Given the universality of GCNs in various time-series problems, such as event prediction [51] and transportation elasticity prediction [52], there are more complex end-to-end models that may be preferred. These two limitations are promising directions for our future work.

CRediT authorship contribution statement

Shangzhe Li: Data curation, Investigation, Writing – original draft. **Junran Wu:** Conceptualization, Methodology, Writing – review & editing. **Xin Jiang:** Supervision, Conceptualization, Writing – review & editing. **Ke Xu:** Conceptualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work has been supported by Guangxi Province Science and Technology Program (2021AA11006), Beijing Natural Science Foundation (Z180005) and Natural Science Foundation of China (Grants No. 12171023).

References

[1] Prodromos Tsinaslanidis, Francisco Guijarro, What makes trading strategies based on chart pattern recognition profitable? Expert Syst. 38 (5) (2021) e12596.
[2] P.E. Tsinaslanidis, A.D. Zapranis, Technical analysis for algorithmic pattern recognition, in: Technical Analysis for Algorithmic Pattern Recognition, 2016.
[3] Junran Wu, Ke Xu, Xueyuan Chen, Shangzhe Li, Jichang Zhao, Price graphs: Utilizing the structural information of financial time series for stock prediction, Inform. Sci. 588 (2022) 405–424.
[4] Ehsan Hoseinzade, Saman Haratizadeh, CNNpred: CNN-based stock market prediction using a diverse set of variables, Expert Syst. Appl. 129 (2019) 273–285.
[5] Zhige Li, Derek Yang, Li Zhao, Jiang Bian, Tao Qin, Tie-Yan Liu, Individualized indicator for all: Stock-wise technical indicator optimization with stock embedding, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 894–902.
[6] Guang Liu, Yuzhao Mao, Qi Sun, Hailong Huang, Weiguo Gao, Xuan Li, JianPing Shen, Ruifan Li, Xiaojie Wang, Multi-scale two-way deep neural network for stock trend prediction.
[7] A.W. Lo, H. Mamaysky, J. Wang, Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation, Social Science Electronic Publishing.
[8] W. Leigh, N. Modani, R. Purvis, T. Roberts, Stock market trading rule discovery using technical charting heuristics, Expert Syst. Appl. (2) (2002) 155–159.
[9] W. Leigh, N. Modani, R. Hightower, A computational implementation of stock charting: abrupt volume increase as signal for movement in New York stock exchange composite index, Decis. Support Syst. 37 (4) (2004) 515–530.
[10] A. Tai Liang Chen, B. Feng Yu Chen, An intelligent pattern recognition model for supporting investment decisions in stock market, Inform. Sci. s 346–347 (2016) 261–274.
[11] Roberto Cervello-Royo, F. Guijarro, K. Michniuk, Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data, Expert Syst. Appl. 42 (14) (2015) 5963–5975.
[12] Yuechu Zheng, Yain-Whar Si, Raymond Wong, Feature extraction for chart pattern classification in financial time series, Knowl. Inf. Syst. 63 (7) (2021) 1807–1848.
[13] Prodromos E. Tsinaslanidis, Subsequence dynamic time warping for charting: Bullish and bearish class predictions for NYSE stocks, Expert Syst. Appl. 94 (2018) 193–204.
[14] Achilleas Zapranis, Prodromos E. Tsinaslanidis, A novel, rule-based technical pattern identification mechanism: Identifying and evaluating saucers and resistant levels in the US stock market, Expert Syst. Appl. 39 (7) (2012) 6301–6308.
[15] Carol L. Osler, P.H. Chang, Head and Shoulders: Not Just a Flaky Pattern, FRB of New York Staff Report (4), 1995.
[16] Rubén Arévalo, Jorge García, Francisco Guijarro, Alfred Peris, A dynamic trading rule based on filtered flag pattern recognition for stock market price forecasting, Expert Syst. Appl. 81 (2017) 177–192.
[17] Sang Hyuk Kim, Hee Soo Lee, Han Jun Ko, Seung Hwan Jeong, Hyun Woo Byun, Kyong Joo Oh, Pattern matching trading system based on the dynamic time warping algorithm, Sustainability 10 (12) (2018) 4641.
[18] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken-ichi Kawarabayashi, Stefanie Jegelka, What can neural networks reason about? 2019, arXiv preprint arXiv:1905.13211.
[19] Tomás Lozano-Pérez, Michael A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Commun. ACM 22 (10) (1979) 560–570.
[20] Lucas Lacasa, Bartolo Luque, Fernando Ballesteros, Jordi Luque, Juan Carlos Nuno, From time series to complex networks: The visibility graph, Proc. Natl. Acad. Sci. 105 (13) (2008) 4972–4975.
[21] A. Kianimajd, M.G. Ruano, P. Carvalho, J. Henriques, T. Rocha, S. Paredes, A.E. Ruano, Comparison of different methods of measuring similarity in physiologic time series, IFAC-PapersOnLine 50 (1) (2017) 11005–11010.
[22] Mathias Niepert, Mohamed Ahmed, Konstantin Kutzkov, Learning convolutional neural networks for graphs, in: International Conference on Machine Learning, PMLR, 2016, pp. 2014–2023.
[23] G.J. Deboeck, Trading on the edge: neural, genetic, and fuzzy systems for chaotic financial markets, in: Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets, 1994.
[24] Pavel Senin, Dynamic time warping algorithm review, Inf. Comput. Sci. Dep. Univ. Hawaii Manoa Honolulu USA 855 (1–23) (2008) 40.
[25] Suk Jun Lee, Kyong Joo Oh, Tae Yoon Kim, How many reference patterns can improve profitability for real-time trading in futures market? Expert Syst. Appl. 39 (8) (2012) 7458–7470.
[26] Prodromos E. Tsinaslanidis, Dimitris Kugiumtzis, A prediction scheme using perceptually important points and dynamic time warping, Expert Syst. Appl. 41 (15) (2014) 6848–6860.
[27] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? 2018.
[28] Na Wang, Dong Li, Qiwen Wang, Visibility graph analysis on quarterly macroeconomic series of China based on complex network theory, Physica A 391 (24) (2012) 6543–6555.
[29] Lucas Lacasa, Wolfram Just, Visibility graphs and symbolic dynamics, Physica D 374 (2018) 35–44.
[30] Luca Di Persio, Oleksandr Honchar, Artificial neural networks architectures for stock price prediction: Comparisons and applications, Int. J. Circuits Syst. Signal Process. 10 (2016) (2016) 403–413.
[31] David M.Q. Nelson, Adriano C.M. Pereira, Renato A. de Oliveira, Stock market's price movement prediction with LSTM neural networks, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 1419–1426.

[32] Jince Li, Bo Yang, Hongguang Li, Yongjian Wang, Chu Qi, Yi Liu, DTDR–ALSTM: Extracting dynamic time-delays to reconstruct multivariate data for improving attention-based LSTM industrial time series prediction models, Knowl.-Based Syst. 211 (2021) 106508.
[33] Hengxu Lin, Dong Zhou, Weiqing Liu, Jiang Bian, Learning multiple stock trading patterns with temporal routing adaptor and optimal transport, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1017–1026.
[34] Xin Du, Kumiko Tanaka-Ishii, Stock embeddings acquired from news articles and price history, and an application to portfolio optimization, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 3353–3363.
[35] Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, Xing Xie, Graph convolution machine for context-aware recommender system, Front. Comput. Sci. 16 (6) (2022) 1–12.
[36] Jia Chen, Ming Zhong, Jianxin Li, Dianhui Wang, Tieyun Qian, Hang Tu, Effective deep attributed network representation learning with topology adapted smoothing, IEEE Trans. Cybern. (2021).
[37] Zhao Li, Xin Wang, Jianxin Li, Qingpeng Zhang, Deep attributed network representation learning of complex coupling and interaction, Knowl.-Based Syst. 212 (2021) 106618.
[38] Guotong Xue, Ming Zhong, Jianxin Li, Jia Chen, Chengshuai Zhai, Ruochen Kong, Dynamic network embedding survey, Neurocomputing 472 (2022) 212–223.
[39] Joo Leito, F.N. Rui, N. Horta, Combining rules between PIPs and SAX to identify patterns in financial markets, Expert Syst. Appl. 65 (2016) 242–254.
[40] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, Karsten M. Borgwardt, Weisfeiler-lehman graph kernels, J. Mach. Learn. Res. 12 (9) (2011).
[41] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, Qiang Yang, Large-scale hierarchical text classification with recursively regularized deep graph-cnn, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1063–1072.
[42] Y. Lecun, L. Bottou, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[43] Jingyuan Wang, Yang Zhang, Ke Tang, Junjie Wu, Zhang Xiong, Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1900–1908.
[44] Genan Dai, Xiaoyang Hu, Youming Ge, Zhiqing Ning, Yubao Liu, Attention based simplified deep residual network for citywide crowd flows prediction, Front. Comput. Sci. 15 (2) (2021) 1–12.
[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[46] Jigar Patel, Sahil Shah, Priyank Thakkar, Ketan Kotecha, Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques, Expert Syst. Appl. 42 (1) (2015) 259–268.
[47] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, Garrison Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, 2017, arXiv preprint arXiv:1704.02971.
[48] Yifan Hao, Huiping Cao, A new attention mechanism to classify multivariate time series, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, 2020.
[49] Hongyan Hao, Yan Wang, Yudi Xia, Jian Zhao, Furao Shen, Temporal convolutional attention-based network for sequence modeling, 2020, arXiv preprint arXiv:2002.12530.
[50] T. Journal, The efficient market hypothesis and its critics, 2004.
[51] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, Tie-Yan Liu, Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 261–269.
[52] Hong-Wei Wang, Zhong-Ren Peng, Dongsheng Wang, Yuan Meng, Tianlong Wu, Weili Sun, Qing-Chang Lu, Evaluation and prediction of transportation resilience under extreme weather events: A diffusion graph convolutional approach, Transp. Res. C 115 (2020) 102619.