Computers and Electrical Engineering 108 (2023) 108687

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Conducting stock market index prediction via the localized spatial–temporal convolutional network☆,☆☆

Changhai Wang, Jiaxi Ren, Hui Liang *, Jingwenqi Gong, Bo Wang

Zhengzhou University of Light Industry, No.136 Kexue Avenue, Zhengzhou, 450000, Henan, China

ARTICLE INFO

Keywords:
Index prediction
Graph convolutional network
Index correlation
Chinese stock market
Market back-testing

ABSTRACT

The stock index movement prediction is a fundamental research issue in financial investment. The high-noise and dynamic of stock data make this problem challenging. Efficiently modeling the underlying spatial–temporal correlations between indices is a potential study area for raising the prediction performance. Existing methods usually use separate modules to capture spatial and temporal correlations, which lose sight of local spatial–temporal correlation. In this paper, we propose a brand-new prediction model named the localized graph convolutional network (LoGCN) for stock index forecasting. This model can represent the intricate local spatial–temporal connections via a well-designed convolution mechanism. First, the localized graph construction method is proposed to model the local spatial–temporal correlations. Then, the localized graph convolutional module is presented to fuse the impact of different indices. Finally, the fully connected network is put forward to transform the features into the expected index trends. 42 Chinese stock market indices were selected to conduct extensive experiments. Three evaluation metrics including the increase rate regression, further trends classification, and stock market back-testing were adopted to the experimental comparison. The experiments show that our method is 4.2%, 3.1%, and 40% better than conventional methods under these three metrics.

1. Introduction

The stock index trend prediction is one of the key foundational issues in financial investment and has attracted tremendous attention [1]. Accurately forecasting the index trend is always challenging due to the high-noise, dynamic and chaotic properties of the stock market. However, it is also important to pay enormous attention to improving the prediction results in order to make a profit in the stock market. Many methods have been developed to deal with stock trend prediction issues, mainly including technical analysis, machine learning, and deep learning. With the development of deep learning, powerful methods like graph convolutional networks (GCN) and its variants have been widely applied to stock market prediction tasks and have achieved promising performance. Different from other deep models, the GCN can fuse the impact of multiple independent time series data, and also reduce the overfitting that commonly exists in stock market trend prediction. Current prediction methods always take an independent approach to model the temporal and spatial correlations. For example, some classical works such as [2,3] utilize the recurrent neural network (RNN) to model the temporal correlation of each stock's time series data first, and then apply the GCN to

☆ This work was partially supported by the National Natural Science Foundation of China (61872439), the Key Science and Technology Program of Henan Province (212102210096, 222102210030).

☆☆ This paper is for regular issues of CAEE. Reviews were processed by Associate Editor Dr. Huimin Lu and recommended for publication.

* Corresponding author.
E-mail address: chw@zzuli.edu.cn (C. Wang).

https://doi.org/10.1016/j.compeleceng.2023.108687
Received 7 October 2022; Received in revised form 12 March 2023; Accepted 14 March 2023
Available online 28 March 2023

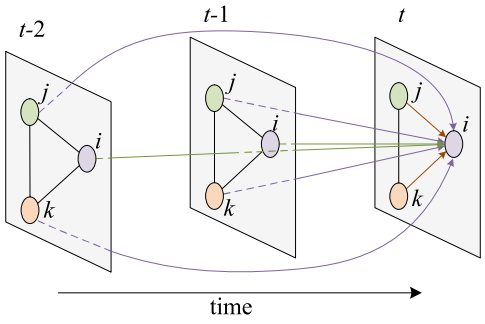

Fig. 1. A schematic diagram of the local spatial–temporal correlation. It contains three indices from time $t-2$ to t. This figure shows three relations that affect index i at time t. The first is the relationships with index i at times $t-1$ and $t-2$, which are marked with green lines. The second is relations with index j and k at time t, which are drawn with brown lines. The third is the correlations to index j and k at times $t-1$ and $t-2$, which are depicted in purple.

mine the spatial relations of multiple stocks. However, this framework cannot model the local spatial–temporal correlation, which is shown in Fig. 1.

In this figure, each node represents a stock market index. Take node i at time t for example, its further trends are affected by three types of relations. The first is the historical time series data, which is denoted with green lines. This relationship is regarded as the temporal correlation. Many time series models [4] have been proposed to handle this correlation. The second is the features of its neighbor indices at time t, which are linked with brown lines. This relationship is considered to be the spatial correlation. Current studies model this associate with GCN. But the input features for the graph convolutional model must be extracted from the historical transaction data of the other indices. They cannot focus on the spatial relationship at each timestamp. The third is the previous data of the other indices, which are connected with purple lines. There are no relevant studies on this relationship for stock prediction.

To sum up, these three relations at timestamp level are collectively referred to as local spatial–temporal correlation in this paper. Jointly modeling these relations is of great significance to improving the prediction performance. However, previous studies use independent modules to capture the temporal and spatial influences. They cannot model the local spatial–temporal correlation at the timestamp level. This paper proposes the LoGCN to model this correlation for stock index prediction issues. The main contributions of our work are as follows.

- A localized spatial–temporal convolutional framework named LoGCN is proposed for stock index forecasting.
- The localized index graph construction method is put forward.
- The localized graph convolution and pooling mechanisms are introduced.
- The effectiveness of the proposed method is proved through extensive experiments, and directions of future research are discussed.

The paper is organized as follows. The related work is presented in Section 2. The proposed prediction method is described in Section 3. The performance evaluations, including experimental design and comparison results, are presented in Section 4. Finally, further research directions are discussed in Section 5.

2. Related works

In recent years, the stock trend prediction has been an attractive topic in the financial field. Many studies are devoted to predicting stock movement or price value based on historical sequences and stock relations. The prediction of stock prices and stock market indexes has gone through three key stages: traditional machine learning methods, and deep learning approaches.

2.1. Classical machine learning

The time series model is a strong model group in the financial market from the very beginning. Technical analysis [5], autoregressive models [6], and other classical approaches to stock prediction are used. The time series approaches are still commonly utilized in stock market investing because they are simple and well explainable. Due to the non-linear correlation that actually exists in financial markets, some traditional machine learning models are used to predict the stock market, including generalized autoregressive conditional heteroskedasticity [7], decision trees [8] and hybrid approaches [9]. The traditional machine learning method, on the other hand, necessitates manually extracting features from transaction data in applications, and the feature extraction method directly impacts the model's prediction effect. At the moment, the artificial feature extraction method based on investment experience is unable to convey the deep correlation between time series data, and projecting the stock market trend has inherent limits.

2.2. Deep learning

With the success of deep learning in image recognition, natural language processing, and other domains [10], more and more academics are turning to stock market forecasting with deep learning models [11]. The deep learning model does not need to manually extract sample features. It just accepts the raw transaction data as input and establishes the mapping between time series data and future trends automatically. Multi-layer perceptron neural networks [12] and deep belief networks [13] are the early method. They utilize the multi-layer fully linked layers to create the link between the historical transaction data and the further trends. As the model becomes complex, the gradient vanishing will be an urgent problem faced by these two models. For improvement, the convolution and poling mechanisms are adopted in convolutional neural networks (CNN) [14]. The decrease of model parameters can effectively overcome the defects of multi-layer fully connected networks. Another most extensively used deep learning models are RNN [15], long short-term memory networks (LSTM) [16], and the hybrid models that combine CNN and RNN [17,18]. They can effectively model the temporal correlations with convolution and recurrent modules. In addition to these models, another time series model, Transformer [19], has also been applied to stock forecasting, and has achieved good results in recent years. However, the trends of stocks are always related, and different stocks share different correlation weights. Utilizing these correlations can significantly improve the prediction performance, and the classical deep learning methods cannot model this characteristic.

2.3. Graph convolution model

The GCN was put forward to model the non-Euclidean relations for multiple time series predictors, which are denoted as spatial correlation in this paper. It has been applied to many application fields, such as traffic forecasting [20], natural language processing [21], power systems [22], and has also been adopted for conducting the stock market predictions in some studies. In these works, each stock is represented as a node, and each edge is built in terms of some predefined stock relations.

Chen et al. [23] incorporate information about related corporations for stock price prediction. Feng et al. [2] contributed a deep learning solution called relational stock ranking, which tailors the LSTM and GCN for stock ranking. Chen et al. [24] model the stock market as a dynamic network, in which the Spearman rank-order correlation is employed to determine the stock correlation. Cheng et al. [3] put forward a dynamic relation graph to eliminate the noise in the predefined static relation and perform the attribute-mattered aggregator to improve the prediction effect. The multi-Hawkes process was used by Yin et al. [25] to learn the correlations between equities and predict their prices. Li et al. [26] generated the stock graph with the historical stock price. Feng et al. [27] put forward an attributed graph attention network model to recommend high return ratio stocks. Xu et al. [28] designed the hierarchical graph neural network to investigate the market state in hierarchical view for stock type prediction. Wu et al. [29] utilize the visibility graph to represent the associations among temporal points, and the node weights are used as a priori knowledge to enhance the learning of temporal attention.

However, the above methods model the temporal and spatial correlations separately. [2,3,23,27,28] first use the RNN model to get the historical data embedding, and then apply the GCN to model the stock correlations. [25,26] use hybrid thoughts to model spatial and temporal relationships. However, as the exhibition of Fig. 1, the index trends are related to not only their neighbors and the previous states but also the other indices at previous timestamp. Current methods cannot capture these correlations at a timestamp level. This paper focuses on this issue by presenting a localized spatial–temporal graph convolutional network, and this model would be a fundamental framework for stock market prediction in further studies.

3. Method

This section presents our index prediction methods. The overview of the prediction model will be introduced in Section 3.1. This model mainly contains the graph convolutional layer and the fully connected layer, which will be detailed in Sections 3.2 and 3.3.

3.1. Framework

The prediction framework is shown in Fig. 2. From this figure, we can see that the inputs of our proposed framework are the historical K-line data. Multiple indices are combined in the form of graph, and the graph node represents an index. The tth graph consists of the K-lines at time t. Several history K-lines of multiple indices make up the graph sequence, and this sequence is the input of the graph convolutional layer. The graph convolutional layer involves multiple convolution sub-layers, and each sub-layer consists of multiple spatial–temporal convolution modules along the time. Each convolution module takes several adjacent graphs as input, and produces one temporary graph. The convolution module includes two parts, which are convolution and pooling. The detailed design will be introduced in Section 3.2. The final output of the graph convolutional layer is a graph containing the convolution features of each index.

After the graph convolutional layer, we can get one graph which consists of the convolved deep features for each index. In addition to the convolution features, this method also extracts technical indicators from the historical transaction data. The convolution and technical features are combined as the input to a fully connected network. The extraction methods for technical features will be put forward in Section 3.3.

Some symbol definitions are first given before introducing the proposed localized spatial–temporal convolutional network. A basic index K-line is denoted as $\boldsymbol{h} = (h_1, \ldots, h_5)$, where five elements represent the opening point, highest point, lowest point,

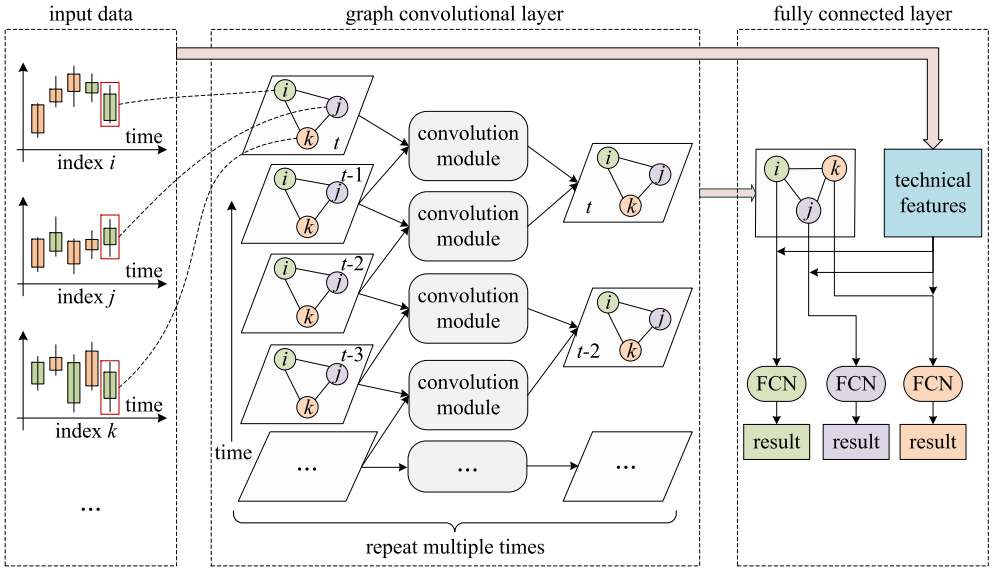

Fig. 2. The framework of the LoGCN. The input data is historical data. K-line data of all indices at the same time compose a graph. The convolution module takes several adjacent graphs as the input, and outputs one temporary graph. After multiple convolutions, the input data is converted to deep features. The deep features and technical features for each index are concatenated as the input of the fully connected network, and the prediction results are produced.

closing point, and trading volume, respectively. It should be known that, h_i is the normalized data, and the normalization method will be introduced in the experimental design section. The prediction sample of index i is a K-line sequence and denoted as

$$\boldsymbol{x}_{\mathcal{T}}^{(i)} = \{\boldsymbol{h}_1^{(i)}, \boldsymbol{h}_2^{(i)}, \dots, \boldsymbol{h}_n^{(i)}\} \tag{1}$$

where n is the sequence length. The K-lines of all indices at time t are denoted as

$$\boldsymbol{x}_{S}^{(t)} = \{\boldsymbol{h}_t^{(1)}, \boldsymbol{h}_t^{(2)}, \dots, \boldsymbol{h}_t^{(N)}\} \tag{2}$$

where N is the index number. $\boldsymbol{x}_S$ and $\boldsymbol{x}_{\mathcal{T}}$ represent spatial and temporal dependency, respectively. Besides, the input of the graph convolutional layer would be written as $\boldsymbol{X}_n^N = \{\boldsymbol{x}_{\mathcal{T}}^{(1)}, \boldsymbol{x}_{\mathcal{T}}^{(2)}, \dots, \boldsymbol{x}_{\mathcal{T}}^{(N)}\}$ or $\boldsymbol{X}_n^N = \{\boldsymbol{x}_S^{(1)}, \boldsymbol{x}_S^{(2)}, \dots, \boldsymbol{x}_S^{(n)}\}$.

3.2. Graph convolutional layer

This section details the graph convolutional layer in the prediction framework. This layer is the most important part of our method, and it includes three subsections. The graph construction section will describe how to build the local correlation for multiple indices. The localized graph convolution and max pooling subsections will give the method to obtain the convolution features.

3.2.1. Graph construction

A graph at time t is represented by $\boldsymbol{G}_t = \{\boldsymbol{V}, \boldsymbol{A}\}$, where $\boldsymbol{V}$ is the set of vertices and $\boldsymbol{A}$ is the set of edges represented by the local adjacency matrix. Each vertex in this paper represents an index, and N denotes the total number of vertices. The local adjacency matrix is expressed as $\boldsymbol{A} = (\boldsymbol{A}^t, \boldsymbol{A}^{t-1}, \dots, \boldsymbol{A}^{t-k+1})_{N\times kN}$, where $\boldsymbol{A}^t = (a_{ij}^t)_{N\times N}$, k is the maximum time interval that is considered in the localized graph convolution. From the definition, we can know that the local adjacency matrix consists of k $N \times N$ sub-matrices. Each sub-matrix represents the index correlations between the current timestamp and the previous timestamp. For example, the edge a_{ij}^{t-e} denotes the link between $\boldsymbol{h}_t^{(i)}$ and $\boldsymbol{h}_{t-e}^{(j)}$.

Fig. 3 shows the calculation method of correlations between index i and index j at time t. As $k = 3$ in this figure, a_{ij}^t, a_{ij}^{t-1}, a_{ij}^{t-2} should be calculated. Without loss of generality, a_{ij}^{t-e} is calculated as the Pearson correlation coefficient of sequence $\boldsymbol{x}^{(i)} = \{h_{(t-l+1)4}^{(i)}, \dots, h_{t4}^{(i)}\}$ and $\boldsymbol{x}^{(j)} = \{h_{(t-e-l+1)4}^{(j)}, \dots, h_{(t-e)4}^{(j)}\}$, where $h_{t4}^{(i)}$ is the closing point of K-line $\boldsymbol{h}_t^{(i)}$, l is the sequence length to calculate the correlation, and e is the time interval. The Pearson correlation coefficient can be obtained with

$$a_{ij}^{t-e} = \frac{\sum_f (h_f^{(i)} - \bar{h}^{(i)})(h_f^{(j)} - \bar{h}^{(j)})}{\sqrt{\sum_f (h_f^{(i)} - \bar{h}^{(i)})^2 \sum_f (h_f^{(j)} - \bar{h}^{(j)})^2}}, \tag{3}$$

where $\bar{h}^{(i)}$ is the average value of the sequence $h^{(i)}$. Take a_{ij}^{t-1} for example, it is the Pearson correlation coefficient of $\{h_{(t-l+1)4}^{(i)}, \dots, h_{t4}^{(i)}\}$ and $\{h_{(t-l)4}^{(j)}, \dots, h_{(t-1)4}^{(j)}\}$. The design of the correlation method is explained as follows. According to the definitions, the local adjacency matrix depicts the indices correlations between the current and previous timestamp. If the coefficient of $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$ is larger, the correlation between $\boldsymbol{h}_t^{(i)}$ and $\boldsymbol{h}_{t-e}^{(j)}$ would be strong. Thus, this value can be regarded as an element of the adjacency matrix.

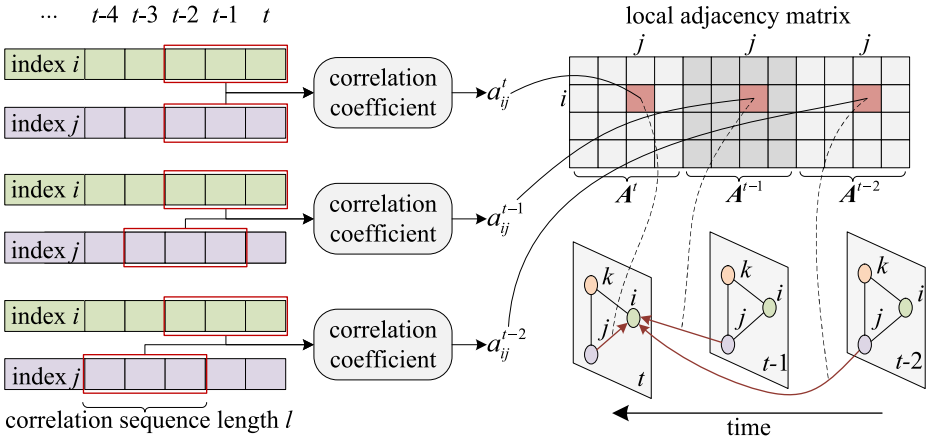

Fig. 3. The local adjacency matrix construction method. The inputs are the closing point sequence for index i and j. The adjacency matrix consists of three parts in this figure, corresponding to three time intervals. a_{ij}^{t-1} is the correlation coefficient of the sequences $\{h_{(t-l+1)4}^{(i)}, \ldots, h_{t4}^{(i)}\}$ and $\{h_{(t-l)4}^{(j)}, \ldots, h_{(t-1)4}^{(j)}\}$.

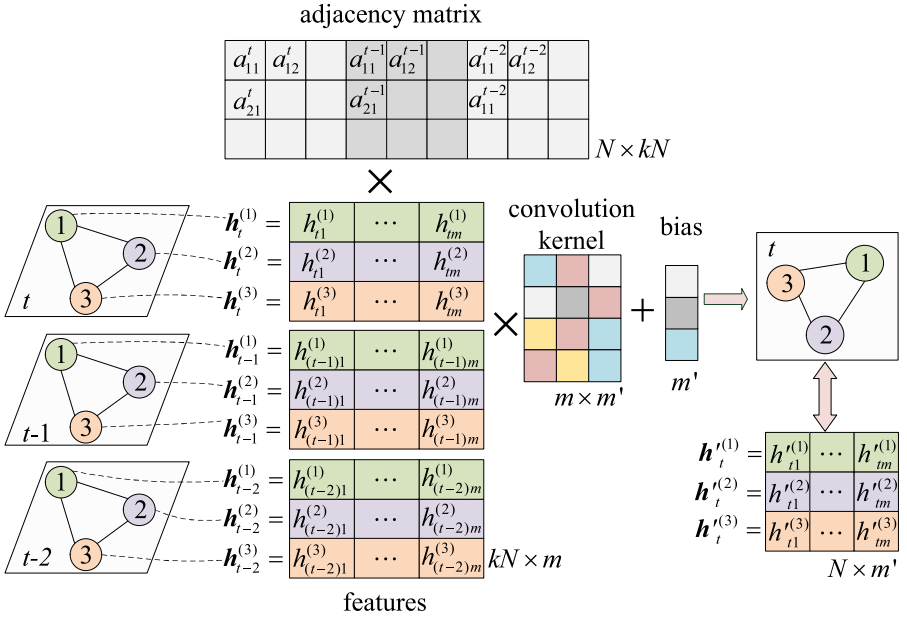

Fig. 4. The localized graph convolutional module. This module converts several adjacent graphs into one convolved graph. The local adjacency matrix $\boldsymbol{A}$ and feature sequence $\{\boldsymbol{x}_S^{(t-2)}, \boldsymbol{x}_S^{(t-1)}, \boldsymbol{x}_S^{(t)}\}$ are the inputs. The convolution kernel and bias are hyperparameters that need to be trained.

3.2.2. Graph convolution

The localized graph convolution is designed to capture the local spatial–temporal correlations. It includes two parts: the convolution and max pooling. This subsection details the convolution mechanism, which is used to fuse the features of current and previous nodes. Fig. 2 shows that the input for each convolution sub-layer is a graph sequence, and each sub-layer contains multiple localized graph convolutional modules along the time. Each module takes k adjacent graphs as input, and produces a temporary graph. The simplified localized graph convolutional module is shown in Fig. 4.

As shown in Fig. 4, the inputs for the convolution are feature sequence $\boldsymbol{X} = \{\boldsymbol{x}_S^{(t-k+1)}, \ldots, \boldsymbol{x}_S^{(t)}\} \in \mathbb{R}^{kN \times m}$ and its corresponding adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times kN}$, where $k = 3$ is the sequence length, $N = 3$ is the index number, and m is the input channel. The outputs are the fused features of all indices $\boldsymbol{x'}_S^{(t)} \in \mathbb{R}^{N \times m'}$ at this timestamp. Formula (4) shows the convolution operation.

$$\boldsymbol{x'}_S^{(t)} = \sigma(\boldsymbol{AXW} + \boldsymbol{b}) \tag{4}$$

where $\boldsymbol{W} \in \mathbb{R}^{m \times m'}$ is the convolution kernel, $\boldsymbol{b} \in \mathbb{R}^{m'}$ is the bias, and m' is the output channel. Formula (4) mainly consists of three steps. $\boldsymbol{A} \times \boldsymbol{X}$ is used to fuse the features with other local vertices, and the matrix with $N \times m$ dimension is obtained. The second step is multiplying the convolution kernel $\boldsymbol{W}$ to transform the feature dimension from m to m'. Finally, the m'-dimension bias is added

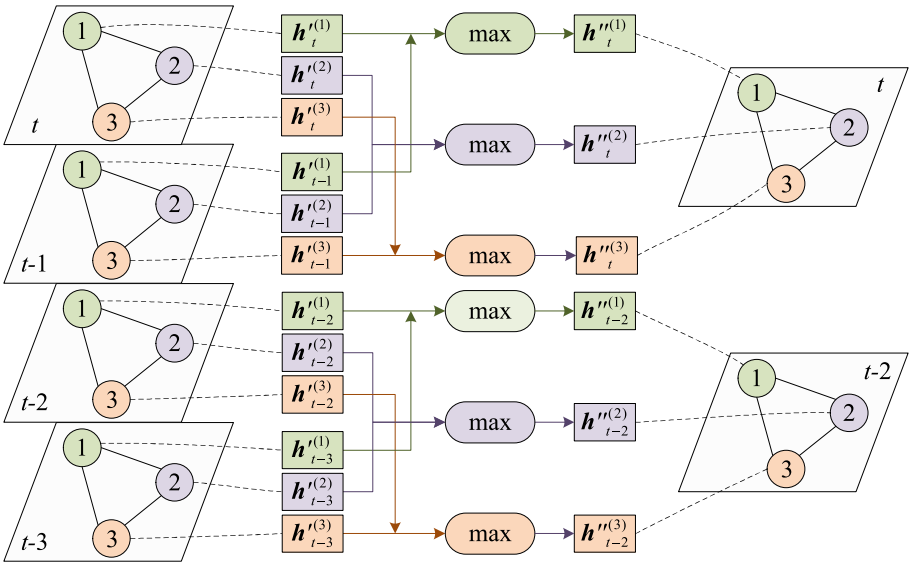

Fig. 5. The max pooling mechanism in the graph convolutional layer. Features of two history nodes are as input for each pooling cell, and the outputs are the maximum value.

to the output. As each convolution module produces a feature graph $\boldsymbol{x}'^{(t)}_{\mathcal{S}}$, all the modules in one sub-lay can bring out a feature graph sequence. To reduce the amount of data, the max pooling is devised.

3.2.3. Max pooling

The purpose of max pooling is to perform data reduction. It takes multiple temporary neighbor graphs as inputs, and gives birth to one graph. The number of neighbor graphs l would be dynamic, which is related to the data length of each graph convolutional sub-layer and should also facilitate the graph construction. For the index prediction issue, this value will be introduced in the experimental section, and can be found in Table 3. Similar to the classical convolutional neural network, the max pooling is designed to obtain the max values of multiple neighboring features. Suppose the input is $\boldsymbol{X}_l^N = \{\boldsymbol{x}_{\mathcal{T}}^{(1)}, \boldsymbol{x}_{\mathcal{T}}^{(2)}, \dots, \boldsymbol{x}_{\mathcal{T}}^{(N)}\}$, the max pooling is

$$\boldsymbol{x}_{\mathcal{S}} = \{max(\boldsymbol{x}_{\mathcal{T}}^{(1)}), max(\boldsymbol{x}_{\mathcal{T}}^{(2)}), \dots, max(\boldsymbol{x}_{\mathcal{T}}^{(N)})\} \tag{5}$$

where the output $\boldsymbol{x}_{\mathcal{S}}$ is the features of all vertices at this timestamp.

Fig. 5 shows the max pooling mechanism. In this figure, the number of neighbor graphs l is 2. The result for each node is the maximum of two nearby history nodes. It should be known that the max pooling results should be normalized before being input into the next sub-layer. After the max pooling, the sequence length would be n/l, and the output sequence would be used as the input of the next graph convolutional sub-layer. With multiple graph convolutional layers, only one feature graph would be produced, and it is named convolutional features, denoted as $\boldsymbol{f}_{conv} = \{\boldsymbol{f}_{conv}^{(1)}, \dots, \boldsymbol{f}_{conv}^{(N)}\}$. This feature will be one of the inputs to the fully connected layer.

3.3. Fully connected layer

The previous section lays out methods for the extraction of convolutional features. But the stock market also has some technical indicators which are widely used in financial investment. Our prediction method also employs these indicators and tries to improve the prediction performance. These technical indicators are named technical features, and they are extracted from the daily K-line transaction data. In this section, the input for each indicator should be $\boldsymbol{x}_{\mathcal{T}}^{(i)} = \{\boldsymbol{h}_1^{(i)}, \boldsymbol{h}_2^{(i)}, \dots, \boldsymbol{h}_\tau^{(i)}\}$, where i is the index and τ is the sample length. However, as the following formulas (6) to (13) are all calculated with a single index, we eliminate the symbols $\mathcal{T}$ and (i) in order to simplify the description. As this section uses the daily K-line as input, τ is not equal to n which is defined in Section 3.1. But it can be obtained with $\tau = n/n_d$, where n_d is the K-line number of each day. For example, if the K-line data used in graph convolution is 5-minute, the n_d will be 48. The detailed data description will be introduced in Section 4.1. The used technical indicators are shown in Table 1.

As shown in the description column, each indicator extracts two features, which represent different calculation periods. For the first three lines, the indicators can be obtained from the original transaction data. We regard the averages for the last 3 and 5 days as the features. For the following lines, the calculation period varies by indicator, which can be found in the description column. The rate of change (ROC) depicts the closing point change of the current day and the previous θ days, shown as

$$ROC(\theta) = \frac{h_{\tau 4} - h_{(\tau-\theta)4}}{h_{(\tau-\theta)4}}, \tag{6}$$

Table 1
The technical indicators are used in our model. The P/E, P/B and exchange ratio are obtained from the stock market. The other indicators can be calculated from the transaction data with Eqs. (6) to (13), and the description column gives the calculation parameter.

Abbreviation	Name	Description
P/E	price/earnings ratio	average of 3 and 5 days
P/B	price/book ratio	average of 3 and 5 days
–	exchange ratio	average of 3 and 5 days
ROC	rate of change	value of 3 and 5 days
WMA	weighted moving average	value of 3 and 5 days
HMA	hull moving average	value of 3 and 5 days
EMA	exponential moving average	value of 5 and 10 days
TEMA	triple exponential moving average	value of 5 and 10 days
CMO	Chande momentum oscillator	value of 9 and 14 days
WR	Williams indicator	value of 7 and 14 days
RSI	relative strength index	value of 3 and 5 days

where θ would be 3 and 5. The weighted moving average (WMA) and the hull moving average (HMA) are used to produce the weighted average closing point of the current θ days, shown as formula (7) and (8).

$$WMA(\theta) = \frac{\sum_{j=1}^{\theta} j h_{(\tau-j+1)4}}{\sum_{j=1}^{\theta} j} \tag{7}$$

$$HMA(\boldsymbol{x}, \theta) = WMA(\boldsymbol{x}, 2 \times WMA(\theta/2) - WMA(\boldsymbol{x}, \theta), \sqrt{\theta}) \tag{8}$$

The exponential moving average (EMA) and triple exponential moving average (TEMA) are also moving average indicators, and can be obtained with Formula (9) and (10).

$$EMA(\boldsymbol{x}, \theta) = \alpha \times h_{\tau 4} + (1-\alpha) \times EMA(\boldsymbol{x}, \theta - 1) \tag{9}$$

$$\begin{aligned} TEMA(x, \theta) =& 3 \times EMA(x, \theta) - 3 \times EMA(EMA(x, \theta), \theta) \\ &+ EMA(EMA(EMA(x, \theta), \theta), \theta) \end{aligned} \tag{10}$$

In these two formulas, θ would be 5 and 10, and $\alpha = 2/(\tau+1)$ is the moving weight. The Chande momentum oscillator (CMO) [5] is a technical indicator that uses momentum to identify relative strength or weakness in the market, shown as

$$CMO(\theta) = \frac{S_u - S_d}{S_u + S_d}, \tag{11}$$

where S_u and S_d are the sum of higher and lower closing points over θ periods. The Williams %R (WR) indicator [30] measures overbought and oversold levels, and it can be obtained with

$$WR(\theta) = \frac{\max\{\boldsymbol{H}_h\} - h_{\tau 4}}{\max\{\boldsymbol{H}_h\} - \min\{\boldsymbol{H}_l\}}, \tag{12}$$

where $\boldsymbol{H}_h$ and $\boldsymbol{H}_l$ are the highest and lowest points in the look back θ period. The relative strength index (RSI) measures the speed and magnitude of the price changes to evaluate overvalued or undervalued conditions, and it can be acquired with

$$RSI(\theta) = \frac{EMA(\boldsymbol{U}, \theta)}{EMA(\boldsymbol{U}, \theta) + EMA(\boldsymbol{D}, \theta)}, \tag{13}$$

where $\boldsymbol{U}$ and $\boldsymbol{D}$ are the upward and downward change sequences. $\boldsymbol{U}$ is characterized by the closing point being higher than the previous day, and $\boldsymbol{D}$ takes opposite. The total number of technical features is 22, and it is denoted as $\boldsymbol{f}_{tech} = \{\boldsymbol{f}_{tech}^{(1)}, \dots, \boldsymbol{f}_{tech}^{(N)}\}$.

After extracting the convolutional and technical features, the fully connected network is applied to get the prediction results. For each index, the fully connected network is represented as

$$\boldsymbol{o}^{(i)} = \boldsymbol{W}_f^{(i)} \boldsymbol{f}^{(i)} + \boldsymbol{b}^{(i)}, \tag{14}$$

where $\boldsymbol{f}^{(i)} = [\boldsymbol{f}_{conv}^{(i)} \parallel \boldsymbol{f}_{tech}^{(i)}]$, and $\parallel$ means concatenation of vectors. The prediction result is obtained for each index after the fully connected network. Two types of prediction tasks are defined in this article, which are gain regression and trend classification. The detailed prediction task will be introduced in the experimental section.

4. Performance evaluation

The Chinese stock market indices were adopted to evaluate our proposed method in this section. First, the experimental design will be introduced in Section 4.1. This section will lay out data preparation, model construction, and evaluation metrics. Then the performance comparison is given in Section 4.2, including the regression, the classification, and the market back-testing.

Table 2
Indices list for performance evaluation. The SSE, SZSE and CSI mean the indices are provided by Shanghai Stock Exchange, Shenzhen Stock Exchange and China Securities Index Company.

Code	Name	Code	Name
000001	SSE Composite Index	000002	SSE A Shares Index
000009	SSE 380 Index	000010	SSE 180 Index
000016	SSE 50 Index	000043	SSE MegaCap Index
000044	SSE MidCap Index	000045	SSE SmallCap Index
000046	SSE Mid&SmallCap Index	000300	CSI 300 Index
000901	CSI Well-Off Index	000922	CSI Dividend Index
000925	CSI RAFI 50 Index	000928	CSI Energy index
000929	CSI Construction Materials Index	000932	CSI Consumer Staples Index
000937	CSI Utilities Index	000965	CSI RAFI 200 Index
000926	CSI Central State-owned Enterprises Composite Index	000938	CSI Private-owned Enterprises Composite Index
000931	CSI Consumer Discretionary Index	000936	CSI Communication Services Index
000941	CSI CN Mainland New Energy Index	000949	CSI Agriculture Thematic Index
000953	CSI Local State-owned Enterprises Composite Index	000955	CSI State-owned Enterprises Composite Index
000958	CSI Innovative and Growing Enterprises Index	000961	CSI Upstream Resources Industry Index
000962	CSI Midstream Manufacturing Industry Index	000963	CSI Downstream Consumption and Services Industry Index
000964	CSI Emerging Industries Index	399934	CSI Financials and Real Estate Index
399001	SZSE Component Index	399005	SSE SME Composite Index
399100	SZSE New Component Index	399330	SZSE 100 Index
399812	CSI Old-Age Industry Index	399903	CSI 100 Index
399905	CSI 500 Index	399933	CSI Health Care Index
399935	CSI Information Technology Index	399998	CSI Coal & Consumable Fuels Index

4.1. Experimental design

4.1.1. Data preparation

There are over 500 indices in the Chinese stock market, distributed across various sectors of the economy. As some indices have been developed just a few years or others lack a certain amount of transaction data, we selected 42 indices for our experiment. These indices cover typical composite and industry indices from the Shanghai and Shenzhen Stock Exchanges. The index list is shown in Table 2.

The 5-minute K-line data ranges from 2009 to 2022 are collected as the input to the graph convolutional layer, and the corresponding 1-day K-line data is adopted to obtain the technical features. The K-line consists of basic transaction data including the opening point, the highest point, the lowest point, the closing point, and trading volume. For index i at time t, the K-line data is written as $\boldsymbol{h}_t^{(i)} = (h_{t1}^{(i)}, \ldots, h_{t5}^{(i)})$. Before performing the graph convolution, data normalization is applied to eliminate the value discrepancies.

The first step in data normalization is converting the index points to the scale of rise and fall. The conversion method for the K-line at time t is shown as

$$h_{tj}{}'^{(i)} = \frac{h_{tj}^{(i)} - h_{(t-1)4}^{(i)}}{h_{(t-1)4}^{(i)}}, \tag{15}$$

where $h_{tj}^{(i)}$ is the jth element at time t, and $h_{tj}^{\prime(i)}$ is the corresponding fluctuation. As illustrated in Formula (15), the element i at time t is calculated with the closing point of time $t-1$. The second step in normalization is applying the linear and sigmod transformations to make the sample data meet the normal distribution, which will facilitate model training. This transformation is shown as

$$\hat{h}_{tj}^{(i)} = 2Sigmod\left(\frac{h_{tj}{}'^{(i)} - \bar{h}_j^{(i)}}{pre(\boldsymbol{H}_j^{(i)}, \beta) - pre(\boldsymbol{H}_j^{(i)}, 1-\beta)}\right) - 1, \tag{16}$$

where $\bar{h}_j^{(i)}$ is the average of the jth element sequence $\boldsymbol{H}_j^{(i)}$. $pre(\boldsymbol{H}_j^{(i)}, \beta)$ is a value in $\boldsymbol{H}_j^{(i)}$ that satisfies $min(abs(P(h < pre(\boldsymbol{H}_j^{(i)}, \beta)) - \beta))$ for each element h in $\boldsymbol{H}_j^{(i)}$. After normalization, the sequence data are divided into prediction samples with a one-day step sliding window. Following the previous studies [24,29], historical data of 16 days were adopted as the prediction sample, and the corresponding label varies by prediction task. For the regression task, the label is the fluctuation of the next day. For the classification task, it would be 0 or 1, which corresponds to the downward and upward, respectively. The total sample number is 3146. These samples were divided into training and testing sets in a 7:3 ratio.

4.1.2. Model construction

This model takes the 16-day historical transaction data as the input. The initial input would be a 768×5 matrix as each day contains 48 5-minute K-lines. The history sequence length k for the localized spatial–temporal convolution is set to 4. The maximum pooling number and the input length for each convolution layer are shown in Table 3.

From this table, we can see that our LoGCN contains 9 layers. The max pooling number l for most layers is 2, except the second layer. This design is fully based on the construction of the localized adjacency matrix. For the first layer, the 5-minute K-line history

Table 3
The localized graph construction parameters.

Layer	1	2	3	4	5	6	7	8	9
Max pooling number	2	3	2	2	2	2	2	2	2
Data scale to construct graph	5 m	10 m	30 m	1h	2h	1d	2d	4d	8d
Input length	768	384	128	64	32	16	8	4	2
Output length	384	128	64	32	16	8	4	2	1

data is utilized to construct the adjacency matrix. As the max pooling number is 2 for the first layer, the output length would be shortened by half. This operation is equivalent to merging the adjacent two K-lines into one, which means the graph feature in the second layer corresponds to two in the first layer. Thus, the graph construction data in the second layer would be using the 10-minute K-line data, which can be synthesized with the 5-minute K-line data. The other layers follow the same principle. The third row in Table 3 shows the data to construct the localized adjacency matrix for each layer. Based on the second and the third rows, the input and the output length for each layer can be obtained. In the experiment, the pytorch framework was applied to build this model, and the optimized stochastic gradient descent technique Adam was utilized to train the model parameters. In model training, the batch size is set to 128 and the learning rate is set to 0.001.

4.1.3. Evaluation metrics

This section introduces assessment metrics so that they can be used to compare the effectiveness of various prediction methods. The suggested model is evaluated from three points of view: regression, classification, and market back-testing.

Regression makes accurate predictions about future scales of rise and fall. The mean square error (MSE) and the mean absolute error (MAE) are commonly used evaluation metrics, shown as

$$\delta_{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_i - y_i\right| \tag{17}$$

and

$$\delta_{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2, \tag{18}$$

where $\hat{y}_i$ and y_i are the predicted and actual fluctuation values. These two metrics evaluate prediction errors, and smaller errors indicate better performance.

The classification predicts the ups and downs of further days. In this paper, the up trends of the next day are denoted as positive, and down trends are labeled negative. Following the previous studies [24,28], the evaluation metrics include Accuracy (Acc) and F1-score ($F_{1,pos}$, $F_{1,neg}$) shown as

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \tag{19}$$

$$F_{1,pos} = \frac{2TP}{2TP + FP + FN} \tag{20}$$

and

$$F_{1,neg} = \frac{2TN}{2TN + FP + FN} \tag{21}$$

where TP, TN, FP and FN stand for true positive, true negative, false positive and false negative, respectively. The greater Acc and F_1 represent better performance, according to the metric definition.

For the market back-testing evaluation, some specified initial money was given and the trading was simulated based on the classification result. The dates ranged from 20180702 to 20220630 were selected for the back-testing, and contained 971 transaction days. For each trading day, the prediction model was re-trained with the historical transaction data before the current day, and the trained model was applied to predict further trends. A simple trading strategy was adopted for our back-testing. When the prediction result for the next day was up, we executed a 'buy' operation with all the money. Otherwise, 'sell' behavior was carried out. It should be noted that this part of the experiment assumes the stock market was liquid enough, and the index point was treated as the trading price instead of the corresponding net asset value of the exchange traded fund (ETF).

Following investor concerns and previous studies [24,28], the trading return ratio (Trr) is regarded as the evaluation metric, which is shown as formula (22).

$$Trr = \frac{total - initial}{initial} \times 100\% \tag{22}$$

As each trade should be charged a certain percentage of commission by the stock exchange, we bring in three expended Trr metrics with different commission ratios. Trr_0 means there is no trading commission, and Trr_2 and Trr_5 set the trading commission ratio to $2e^{-4}$ and $5e^{-4}$, respectively. For the Chinese stock market, $0.5e^{-4}$ to $5e^{-4}$ would be the common commission rate in recent years.

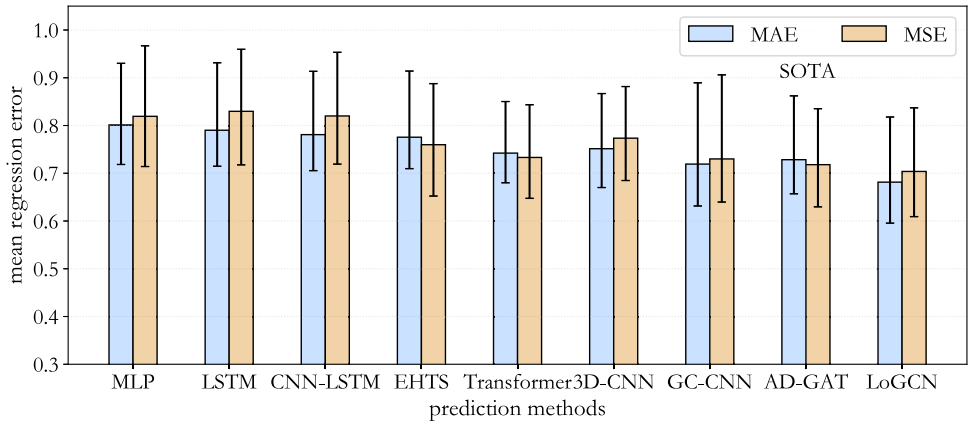

Fig. 6. The regression results for different prediction methods. Each result represents the mean error of 42 indices, and the corresponding error bar shows the error range of these indices. This figure indicates that the regression error of the current state-of-the-art method AD-GAT is 0.723, and our method is 0.693.

4.2. Performance comparison

This section first introduces comparative baselines, and then gives the experimental results. According to the evaluation metrics introduced in Section 4.1.3, the results of regression, classification, and back-testing will be put forward accordingly in Section 4.2.2 to Section 4.2.4. Finally, the result discussion will be held in Section 4.2.5.

4.2.1. Baselines

There are many methods proposed to predict the stock market's further trends. Some classical models include MLP [12], LSTM [4], CNN-LSTM [17], ETHS [18], Transformer [19], 3D-CNN [14], GC-CNN [24], AD-GAT [3], etc. Most of these models were published in the last two years. We reimplement these methods on the dataset introduced in Section 4.1.1 and make a comparison with our methods. As the prediction results always contain randomness, the simulation for each method repeats 100 times.

As the MLP and LSTM models need to manually extract the sample features, 22 technical indicators introduced in Section 3.3 were chosen as input for these two methods in the experiment. The number of hidden layers in the MLP model was 2, and the number of neurons in the output layer was 1 and 2 for regression and classification tasks, respectively. The output of the LSTM unit was input to a fully connected layer in the LSTM model, and then the prediction results were obtained through the output layer. The input size for the CNN-LSTM and ETHS was 768×5. For CNN-LSTM, the CNN module was followed by an LSTM unit. For ETHS, the results of the CNN and LSTM models were first extracted respectively, and then concatenated into one feature vector. For the Transformer, the daily K-lines of the previous 16 days are as the input, according to the original research. The embedding matrix size of 16×32 is first obtained through a fully connected network. The number of encoder and decoder layers is 3, respectively. The input size for the 3D-CNN was 768×5× 42, and the output was prediction results for 42 indices. We calculated the adjacency matrix and performed graph convolution for the GC-CNN using the Spearman rank-order correlation coefficient. The inputs of the AD-GAT were the one-dimensional convolution results. The adjacency matrix was trained with the historical transaction data, and the attribute-mattered aggregator was added to the graph convolution.

4.2.2. The regression results

The regression errors of different methods are shown in Fig. 6 and Table 4. It should be known that each result is the mean error of all 42 indices. As the performances of 42 indices are different, the error bar is plotted to show the error range of all results. The top of the error bar means the worst prediction result, and the bottom is the best. This figure reveals the following results. First, our model outperforms the current state-of-the-art method AD-GAT. It decreases the regression error from 0.723 to 0.693, and reduces it by about 4.2%. Second, the worst result of our model is also better than that of other methods. It means our method performs stably. Finally, the results of two loss functions are substantially the same, which reveals that the loss function has little impact on the prediction results.

4.2.3. The classification results

In addition to the regression results, Fig. 7 and Table 5 shows the classification results with three evaluation metrics. The prediction task in this figure is the ups and downs of the following day. The upward trend is denoted as positive, and the downward trend is regarded as negative. We can draw the following conclusions from this figure.

First, our method stably outperforms all the other methods with the metrics of Acc and $F_{1,pos}$. For the metric of $F_{1,neg}$, our method is basically equal to several other methods. However, this figure demonstrates that our method is the state-of-the-art method. Second, the error bar for each result reveals that the performance of different indices varied significantly from one another. The best result would be higher than 65%, and the worst would be just lower than 45%. This means accurately predicting the further trends of the stock market has a long way to go.

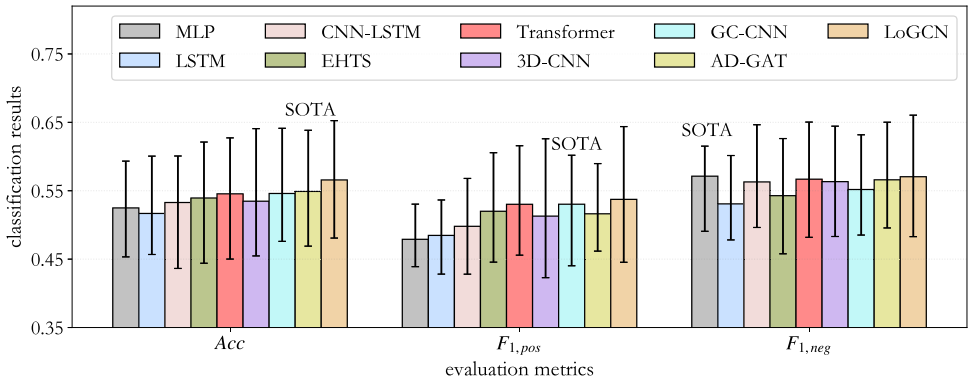

Fig. 7. The classification results for different prediction methods with three evaluation metrics. The SOTA indicates the current best methods. For metrics of prediction accuracy and $F_{1,pos}$, our method performs better than all other methods.

Table 4
The regression results correspond to Fig. 6. The bold texts indicate the current best results.

		MLP	LSTM	CNN-LSTM	EHTS	Transformer	3D-CNN	GC-CNN	AD-GAT	LoGCN
MAE	mean	0.801	0.790	0.781	0.776	0.742	0.751	**0.719**	0.728	0.681
	max	0.930	0.931	0.914	0.914	**0.850**	0.867	0.889	0.862	0.818
	min	0.718	0.715	0.706	0.710	0.680	0.670	**0.632**	0.657	0.596
MSE	mean	0.819	0.830	0.820	0.760	0.733	0.774	0.730	**0.718**	0.704
	max	0.967	0.960	0.953	0.888	0.844	0.882	0.905	**0.835**	0.837
	min	0.714	0.718	0.719	0.652	0.648	0.685	0.639	**0.630**	0.609

Table 5
The classification results correspond to Fig. 7. The bold texts indicate the current best results.

		MLP	LSTM	CNN-LSTM	EHTS	Transformer	3D-CNN	GC-CNN	AD-GAT	LoGCN
Acc	mean	0.525	0.517	0.533	0.539	0.546	0.535	0.546	**0.549**	0.566
	max	0.593	0.601	0.601	0.621	0.623	0.641	**0.641**	0.638	0.653
	min	0.453	0.457	0.436	0.444	0.464	0.455	**0.476**	0.469	0.481
$F_{1,pos}$	mean	0.479	0.485	0.498	0.520	0.530	0.513	**0.530**	0.516	0.537
	max	0.530	0.537	0.568	0.606	0.602	**0.626**	0.602	0.590	0.644
	min	0.439	0.428	0.428	0.446	0.461	0.423	0.440	**0.462**	0.445
$F_{1,neg}$	mean	**0.571**	0.531	0.563	0.543	0.567	0.563	0.552	0.566	0.570
	max	0.611	0.602	0.647	0.626	0.647	0.644	0.632	**0.650**	0.661
	min	0.487	0.478	0.496	0.458	0.484	0.483	0.485	**0.496**	0.483

Table 6
The average return ratios correspond to Fig. 8. The bold texts indicate the current best results.

	Hold	Random	MLP	LSTM	CNN-LSTM	EHTS	Transformer	3D-CNN	GC-CNN	AD-GAT	LoGCN
Trr_0	0.365	0.194	0.569	0.433	0.703	0.836	0.942	0.737	0.958	**1.031**	1.430
Trr_2	0.365	0.091	0.419	0.298	0.538	0.662	0.720	0.572	0.782	**0.831**	1.201
Trr_5	0.365	−0.067	0.224	0.120	0.332	0.442	0.508	0.367	0.538	**0.601**	0.906

4.2.4. The market back-testing results

The mean trading return ratios for different methods are shown in Fig. 8 and Table 6. The results of two other investment strategies are added to this figure, which are Hold and Random. The Hold means the investor buys the indices with the closing point of 2018-07-02 and sells them at 2022-06-30. This strategy represents the inherent income of the stock market. The Random means buying and selling the index at random, and this strategy imitates the investor with no investment experience. As the stock exchange charges a certain percentage of commission for each trading operation, Fig. 8 displays three trading return ratios with different commission rates. Trr_0, Trr_2 and Trr_5 indicate that the commission ratio is 0, $2e^{-4}$ and $5e^{-4}$, respectively.

Concise results can be obtained from Fig. 8. As the Hold strategy has no trading operation, the income is the same with different commissions. It also indicates the average inherent income for the 42 indices is 36%. In all these methods, the Random gives the worst result, and the income is even negative as a result of the increased commission rate. For the following methods, our approach yields the best performance. For the Trr_0, our method gets a return rate of about 143%, and outperforms the current best method by 39%. As the commission rate increases to $5e^{-4}$, the income of our method is still capable of reaching 90%.

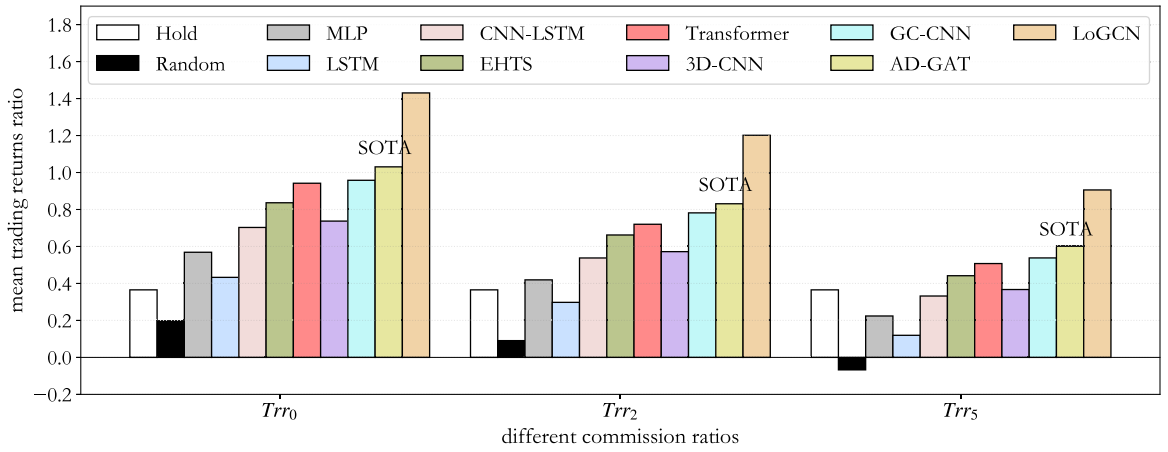

Fig. 8. The average trading return ratios for different prediction methods with different commission ratios. The other two investment strategies were included in this figure. The Hold means buying the index at the beginning and holding it until the end, while the Random conducts the 'buy' and 'sell' operations randomly. Our method significantly outperforms other methods.

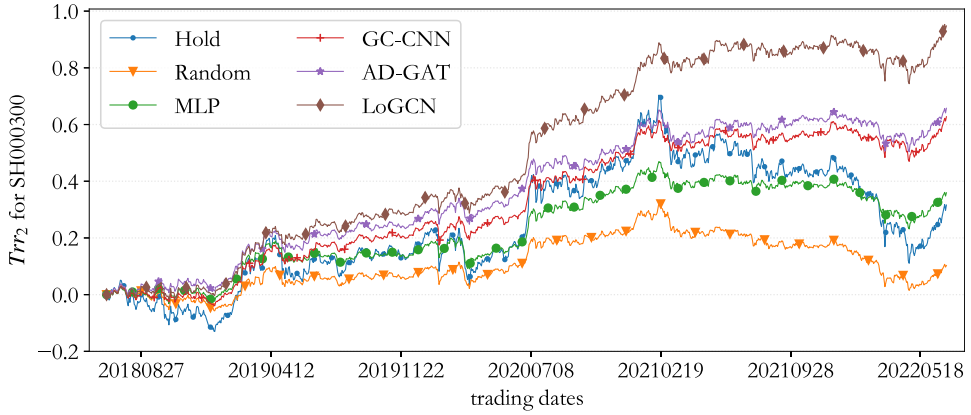

Fig. 9. The income curves for SH000300. Our method obtains 94.77% investment returns, and the current best method produces 65.73%.

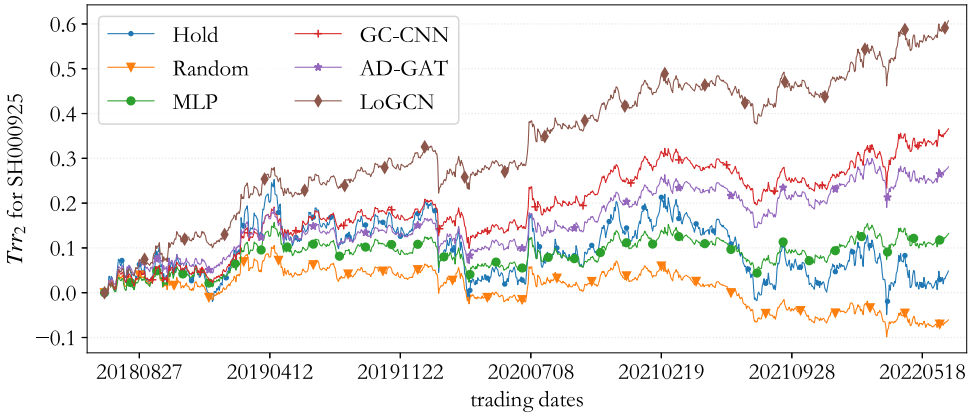

Fig. 10. The income curves for SH000925. Our method obtains 60.73% investment returns, and the current best method gives 36.63%.

To further analyze the predicted effect, Figs. 9 to 12 display the income curves for different methods. In order to make the figure clear, we only keep the curves of 6 methods, and the curves of other methods are similar. These figures show the income curves of four indices, including SH000300, SH000925, SH000928, and SH000932. These four indices take different obtains in the past

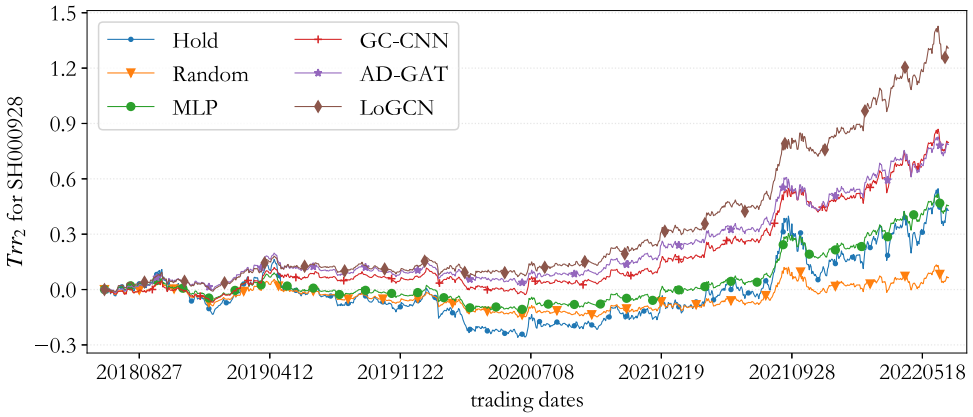

Fig. 11. The income curves for SH000928. Our method yields 130.9% of income, and the current best method provides 79.74%.

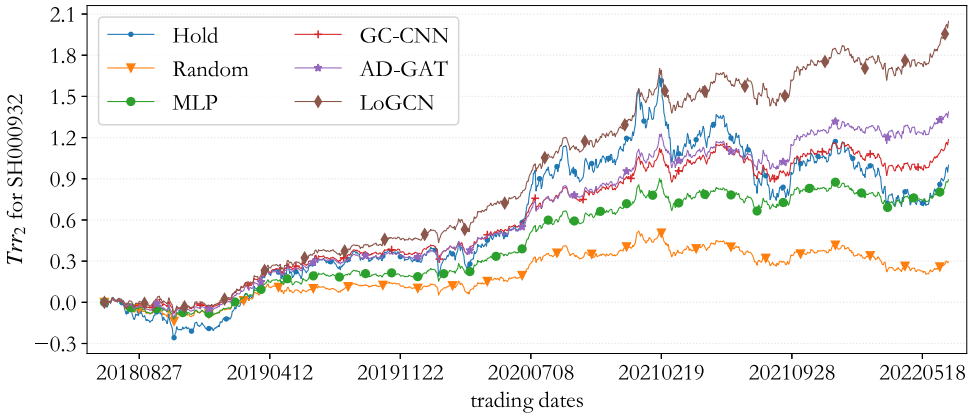

Fig. 12. The income curves for SH000932. Our approach generates 204.67% of income, while the current best method produces 138.73%.

four years. The SH000925 offers just a 5% increase, while the SH000932 takes 100% returns. These four indices were selected to display the performance of our method on different index trends. Clearly results can be obtained from these figures that the proposed LoGCN obviously performs better than other methods. For the worst-performing index SH000925, our approach has yielded 61% returns, and it is far beyond the inherent benefits and other prediction methods. Observing the best-performing index SH000932, our method obtains 205% income and is also far beyond the state-of-the-art methods.

4.2.5. Results discussion

It can be inferred from the summary of the aforementioned trials that our approach produces the best outcomes across all evaluation measures, including regression, classification, and trading back-testing. The following explains why our solution performs better than other approaches. For the first 5 methods, from MLP to Transformer, they are classical time series models, and they do not consider the correlations between indices. Thus, their performance is relatively poor. Although 3D-CNN used all index data to predict the indices further trends, the non-Euclidean relation between indices was also not taken into account. So it does not perform well either. The last three methods all model the prediction with graphs. However, GC-CNN and AD-GAT do not factor in the local correlation between the trends of different indices, which is a significant characteristic for improving the prediction performance. Our proposed model, LoGCN, introduces the localized graph convolution mechanism, which incorporates the effects of spatial and temporal correlations into a short time period. It can effectively capture the short-term index effects. Thus, our method obtains state-of-the-art results.

5. Conclusions

This article proposes the LoGCN to tackle the issue that current methods cannot consider the spatial and temporal relationship in a short time period. Multiple sub-modules of this method are detailed, including adjacency matrix construction, localized graph convolution, max pooling, etc. 42 commonly used indices in the Chinese stock market are regarded as experimental data. Our method obtains state-of-the-art results in evaluation metrics of regression, classification, and market back-testing. In stock market

investment, our method can be regarded as a decision strategy to improve the investment return. However, our experimental results are the average of multiple experiments. If a single model is used for forecasting, the investment returns may fluctuate significantly.

Our proposed LoGCN can be regarded as a basic framework for stock market prediction. Further research directions could be in the following two aspects. First, improving the classification accuracy is still an ongoing concern as current accuracy is less than 60%. Applying some robustness tricks such as adversarial training or multi-graph convolution to the model would be an efficient way to improve prediction accuracy. Another direction would be designing new investment strategies based on our prediction models. The back-testing strategy in our experiments executes the trading operation based on the classified result of the following day. The improved strategy would use the prediction confidence or the regression error as the trading reference. On the other hand, properly reducing the trading frequency is also an improvement direction.

CRediT authorship contribution statement

Changhai Wang: Conceptualization, Methodology, Writing – original draft. **Jiaxi Ren:** Methodology, Writing – review & editing. **Hui Liang:** Funding acquisition, Resources, Supervision. **Jingwenqi Gong:** Data curation. **Bo Wang:** Data curation, Visualization.

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.

Data availability

The authors do not have permission to share data.

References

[1] Thakkar A, Chaudhari K. Fusion in stock market prediction: a decade survey on the necessity, recent developments, and potential future directions. Inf Fusion 2021;65:95–107.
[2] Feng F, He X, Wang X, Luo C, Liu Y, Chua T-S. Temporal relational ranking for stock prediction. ACM Trans Inf Syst 2019;37(2):1–30.
[3] Cheng R, Li Q. Modeling the momentum spillover effect for stock prediction via attribute-driven graph attention networks. In: Proceedings of the AAAI conference on artificial intelligence. Vol. 35. No. 1. 2021, p. 55–62.
[4] Luo J, Zhu G, Xiang H. Artificial Intelligent based day-ahead stock market profit forecasting. Comput Electr Eng 2022;99:107837.
[5] Tushar C, Stanley K. The new technical trader. John Wiley & Sons; 1994.
[6] Wang Y, Guo Y. Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost. China Commun 2020;17(3):205–21.
[7] Wang L, Ma F, Liu J, Yang L. Forecasting stock price volatility: New evidence from the GARCH-MIDAS model. Int J Forecast 2020;36(2):684–94.
[8] Kamble A. Short and long term stock trend prediction using decision tree. In: 2017 International conference on intelligent computing and control systems. IEEE; 2017, p. 1371–5.
[9] Zolfaghari M, Gholami S. A hybrid approach of adaptive wavelet transform, long short-term memory and ARIMA-GARCH family models for the stock index prediction. Expert Syst Appl 2021;182:115149.
[10] Dong S, Wang P, Abbas K. A survey on deep learning and its applications. Comp Sci Rev 2021;40:100379.
[11] Jiang W. Applications of deep learning in stock market prediction: recent progress. Expert Syst Appl 2021;184:115537.
[12] Bose A, Hsu C-H, Roy SS, Lee KC, Mohammadi-ivatloo B, Abimannan S. Forecasting stock price by hybrid model of cascading Multivariate Adaptive Regression Splines and Deep Neural Network. Comput Electr Eng 2021;95:107405.
[13] Chen J-H, Hao Y-H, Wang H, Wang T, Zheng D-W. Futures price prediction modeling and decision-making based on DBN deep learning. Intell Data Anal 2019;23(S1):53–65.
[14] Ehsan H, Saman H. CNNpred: CNN-based stock market prediction using a diverse set of variables. Expert Syst Appl 2019;129:273–85.
[15] Zhang X, Gu N, Chang J, Ye H. Predicting stock price movement using a DBN-RNN. Appl Artif Intell 2021;35(12):876–92.
[16] Fister D, Perc M, Jagrič T. Two robust long short-term memory frameworks for trading stocks. Appl Intell 2021;51(10):7177–95.
[17] Lu W, Li J, Wang J, Qin L. A CNN-BiLSTM-AM method for stock price prediction. Neural Comput Appl 2021;33(10):4741–53.
[18] Kamara AF, Chen E, Pan Z. An ensemble of a boosted hybrid of deep learning models and technical analysis for forecasting stock prices. Inform Sci 2022;594:1–19.
[19] Wang C, Chen Y, Zhang S, Zhang Q. Stock market index prediction using deep Transformer model. Expert Syst Appl 2022;208:118128.
[20] Jiang W, Luo J. Graph neural network for traffic forecasting: A survey. Expert Syst Appl 2022;117921.
[21] Wu L, Chen Y, Shen K, Guo X, Gao H, Li S. Graph neural networks for natural language processing: A survey. Found Trends® Mach Learn 2023;16(2):119–328.
[22] Liao W, Bak-Jensen B, Pillai JR, Wang Y, Wang Y. A review of graph neural networks and their applications in power systems. J Mod Power Syst Clean Energy 2021.
[23] Chen Y, Wei Z, Huang X. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In: Proceedings of the 27th ACM international conference on information and knowledge management. 2018, p. 1655–8.
[24] Chen W, Jiang M, Zhang W-G, Chen Z. A novel graph convolutional feature based convolutional neural network for stock trend prediction. Inform Sci 2021;556:67–94.
[25] Yin T, Liu C, Ding F, Feng Z, Yuan B, Zhang N. Graph-based stock correlation and prediction for high-frequency trading systems. Pattern Recognit 2022;122:108209.
[26] Li W, Bao R, Harimoto K, Chen D, Xu J, Su Q. Modeling the stock relation with graph network for overnight stock movement prediction. In: Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence. 2021, p. 4541–7.
[27] Feng S, Xu C, Zuo Y, Chen G, Lin F, XiaHou J. Relation-aware dynamic attributed graph attention network for stocks recommendation. Pattern Recognit 2022;121:108119.

[28] Xu C, Huang H, Ying X, Gao J, Li Z, Zhang P, et al. HGNN: Hierarchical graph neural network for predicting the classification of price-limit-hitting stocks. Inform Sci 2022;607:783–98.
[29] Wu J, Xu K, Chen X, Li S, Zhao J. Price graphs: Utilizing the structural information of financial time series for stock prediction. Inform Sci 2022;588:405–24.
[30] Williams LR. How I made one million dollars... last year... trading commodities. Windsor Books; 1979.

Changhai Wang, was born in Liaocheng, Shandong Province, China, in 1987. He received the M.S. degree from Jiangsu University in 2012 and the Ph.D. degree from Nankai University in 2016. He has been working at the Zhengzhou University of Light Industry since 2017. His research includes deep learning, convolutional neural networks, and financial data analysis.

Jiaxi Ren, was born in Pingdingshan, Henan Province, China, in 2000. She is now a M.S. student at Zhengzhou University of Light Industry. Her research interests include graph convolutional networks and anomaly detection.

Hui Liang, is currently a professor at Zhengzhou University of Light Industry. He was a Marie Curie Research Fellow at the National Centre for Computer Animation, Bournemouth University, UK. He received his Ph.D. degree from the Communication University of China. His current research interests include big data, deep learning, and intelligent data management.

Jingwenqi Gong, was born in Xuchang, Henan Province, China, in 2002. He is now an undergraduate student at Zhengzhou University of Light Industry. His research interests include time-series analysis.

Bo Wang, was born in Zhoukou, Henan Province, China, in 1987. He received his Ph.D. degree from Xi'an Jiaotong University in 2017. He has been working at the Zhengzhou University of Light Industry since 2017. His research interests include time series data analysis, machine learning, and cloud computing.