Computers and Electrical Engineering 103 (2022) 108285

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

MG-Conv: A spatiotemporal multi-graph convolutional neural network for stock market index trend prediction☆,☆☆

Changhai Wang [a], Hui Liang [a,*], Bo Wang [a], Xiaoxu Cui [b], Yuwei Xu [c]

[a] Zhengzhou University of Light Industry, No. 136 Kexue Avenue, Zhengzhou, 450000, Henan, China
[b] China University of Political Science and Law, No. 25 Xitucheng Road, Beijing, 102249, Beijing, China
[c] Purple Mountain Laboratories for Network and Communication Security, No. 9 Mozhou East Road, Nanjing, 211111, Jiangsu, China

ARTICLE INFO

Keywords:
Stock index prediction
Index constituent stocks
Graph convolutional neural network
One-dimensional convolutional neural network
Data normalization

ABSTRACT

Index trend prediction is a critical topic in the sphere of financial investment. An index trend prediction model based on a multi-graph convolutional neural network termed MG-Conv is suggested in this paper. First, the data normalization and one-dimensional convolutional neural network are proposed to extract the deep features of historical transaction data. Then, two types of correlation graphs named static and dynamic graphs are defined. Finally, the multi-graph convolution is performed on these two graphs, and the results of graph convolution are transferred to anticipated values with fully connected networks. 42 Chinese stock market indices were selected as experimental data. Classic approaches including LSTM, 3D-CNN, GC-CNN, and AD-GAT were chosen as comparison benchmarks. The results show that the method can reduce the average prediction error by 5.11% and performs strong robustness.

1. Introduction

Predicting the future trend of the stock market via historical transaction data in the stock market is an eternal topic in the field of financial investment [1]. Accurate stock forecasting is the most important factor affecting investment returns. Due to the dynamic, non-stationary nature of stock prices and the influence of sporadic events like company operational circumstances and public opinion, accurately predicting the price movement of a single stock is challenging. The stock market index is an indicator that reflects the overall stock price trend of specific industries or companies in the stock market [2]. Compared to a single stock, the stock market index is less affected by contingency variables such as a single company's operating conditions and is well predictable.

Hundreds of stock market indices have been created since the beginning of the stock market, including composite indices that represent the general market trend, industry indices that reflect the trends of various sectors, scale indices that track the movement of various-sized businesses, etc. At the same time, varieties of exchange-traded funds (ETF) are developed to invest in indices by stock exchanges. The purchase of exchange-traded funds is a popular way for many individuals to gain the asset appreciation. As a result, predicting the future trend of various types of indexes is critical for increasing investment performance. Additionally, the forecast of the index trend can serve as a guide for the formulation of national economic policies since the strengths of different index trends represent the characteristics of various phases of economic growth.

☆ This paper is for regular issues of CAEE. Reviews processed and recommended for publication to the Editor-in-Chief by Huimin Lu.

☆☆ 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).

* Corresponding author.
E-mail address: hliang@zzuli.edu.cn (H. Liang).

https://doi.org/10.1016/j.compeleceng.2022.108285
Received 9 May 2022; Received in revised form 30 July 2022; Accepted 30 July 2022
Available online 27 August 2022

We focus on the Chinese A-share market ETF investment topic in this paper and give investors references by forecasting future trends of several indices. Our purpose is that, given the index point data of the last few days, the model can output the scale of rise and fall for the next several days. According to recent studies, the graph convolutional network (GCN) is employed in this study to fuse index correlations in the trend prediction. The following issues come up when doing graph convolution. The first challenge is data normalization during index sequence data preparation. Converting historical K-line data into a scale of rise and decline is the main effort in the data processing. Current normalization techniques [3], such as linear function normalization and 0-mean normalization, will lead to severely unbalanced data distribution. The result of this issue is that most training samples will have modest gradients, and parameter optimization will be dominated by a small number of examples with big gradients.

The second challenge is how to model and fuse the correlation between indices, which is regarded as the concept of a graph in graph convolutional networks. To the best of investors' knowledge, indices with the same constituent stocks usually follow the same trend. This intrinsic index relationship can be regarded as a static graph that is formed when indexes are developed. The issue of constructing a static graph between indexes based on constituent stocks data must be overcome. In addition to the static graph, there are trendy correlations between indices of different composite strategies. For example, some industry indexes always hold negative correlations, a rise in one index is often accompanied by a decline in the other. This correlation is derived from historical transaction data, which is termed as the dynamic graph. The design of this dynamic graph also needs to be solved. However, how to perform the convolution operations on both graphs is also challenging.

In order to overcome the existing challenges, we proposed MG-Conv, a model integrated one-dimensional and multi-graph convolutional network. The key contributions of this article are summarized as follows.

- An index prediction framework integrates the one-dimensional convolutional network and the multi-graph convolutional network is put forward.
- A transaction data normalization method based on the linear and Sigmod transformation is proposed.
- The construction methods of the static and dynamic graph are given, and the multi-graph convolution operations are defined.
- 42 A-share indices are selected to evaluate the performance of MG-Conv, and directions of future research are discussed.

The paper is organized as follows. The related work is presented in Section 2. The proposed prediction framework is shown in Section 3. The presented data normalization method and one-dimensional convolutional network are introduced in Section 4. The graph construction and convolution are defined in Section 5. Finally, Sections 6 and 7 illustrate the performance evaluation and conclusions of the proposed method.

2. Related work

The prediction of stock prices and stock market indices has gone through three major stages: time series analysis methods, traditional machine learning methods, and deep learning approaches [4]. The autoregressive integrated moving average model [5] is the most well-known time series analysis method. These time series models are gradually replaced by machine learning techniques since they are unable to capture the nonlinear relationship between time series data, and there are few studies in this area in recent years. Decision trees [6], neural networks [7] and boost-based methods [8] are examples of machine learning methods, and they are still commonly used in investment. But these techniques require manually extracting features from transaction data, and the extracted features directly affect the model's performance. Additionally, the manually extracted features are unable to convey the deep correlation between time series data, which led to inherent limits for stock index prediction.

With the success of deep learning in image recognition, natural language processing, and other domains [9], more and more academics are turning to stock market forecasting with deep learning models. 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 network [10], convolutional neural network [11], recurrent neural network [12], and graph convolutional neural network [13] were extensively employed. Multi-layer perceptron neural network uses multi-layer fully linked layers to create the mapping of transaction data to the further trends. The gradient vanishing is the most serious issue they encounter. The convolutional neural network takes the pooling mechanism to decrease model parameters. But it requires the input data to retain a stable order locally, which is not suitable for modeling the correction of stock market indexes. The recurrent neural network is another widely used model, which is expert in modeling time series data. But it is also unable to simulate multi-index sequences with non-Euclidean links [14].

The graph convolutional network (GCN) was introduced to model graph data in recent years. It has been successfully applied to many research fields, such as traffic forecasting [15], recommend system [16], and it also has been used to predict stock market trends in a few studies. Feng et al. [13] contributed a deep learning solution called Relational Stock Ranking, which outperforms state-of-the-art methods in back-testing. Chen et al. [17] employed the information of stockholders to construct the stock graph. [18] constructs graph through converting their industry categories into multi-hot binary vectors. These three solutions utilize explicit domain knowledge such as sector-industry relation, supplier-consumer relation and stockholders to model influence between different stocks. But this graph construction method cannot apply to the index prediction as the index did not have such domain knowledge. Besides, the graph obtained from domain knowledge cannot reflect the non-intuitive latent correlation between indices.

Zhong et al. [19] and Chen et al. [20] employed the Spearman rank-order correlation to dynamically determine stock correlations, and then converted stock transaction data into picture data for convolution. The multi-Hawkes process was used by Yin et al. [21] to

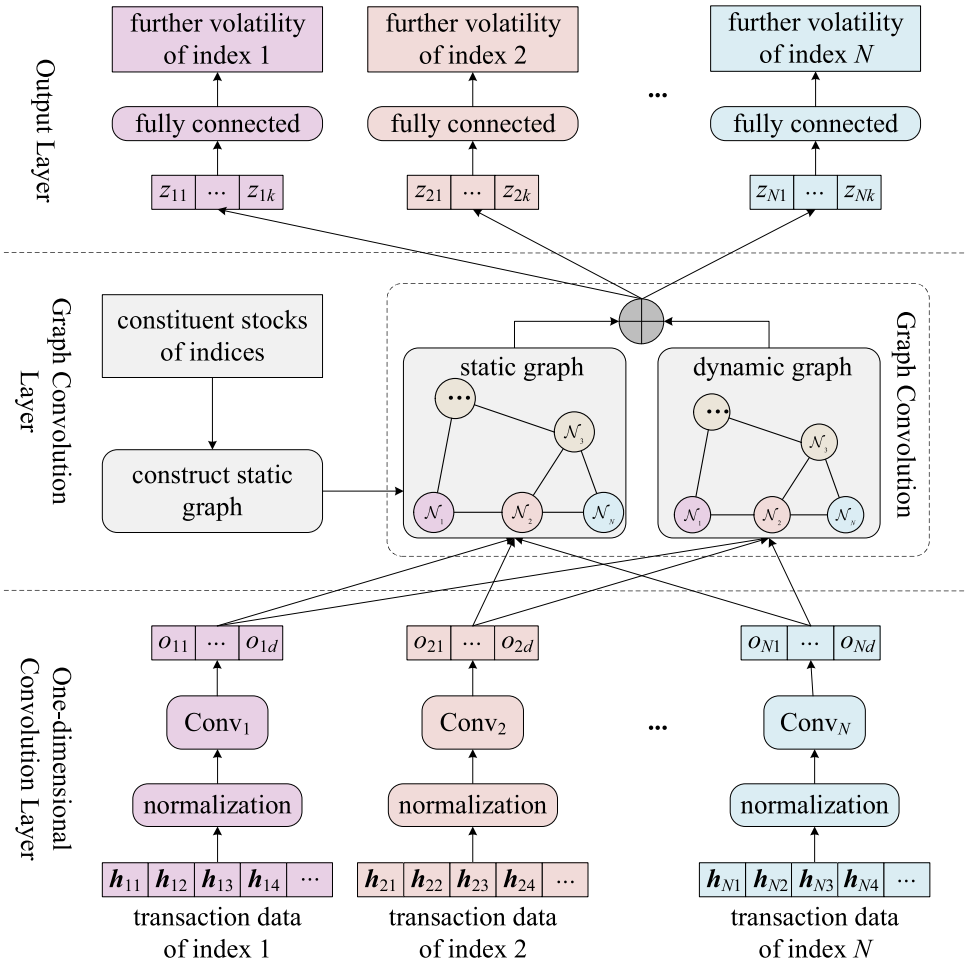


Fig. 1. The framework of MG-Conv. The inputs are the transaction data of indices. Then the one-dimensional convolution and the multi-graph convolution are utilized to extract and fuse the index features. Finally, further fluctuation for each index is obtained with a fully connected network.

learn the correlations between stocks and predict their prices. Cheng et al. [22] put forward a dynamic relation graph to eliminate the noise in the predefined static relation and utilizes the attribute-mattered aggregator to improve the prediction performance. Li et al. [23] generated the stock graph with the historical stock price. [24] generates the graph structure with the mutual information and cross-correlation coefficient. These methods, on the other hand, are all dynamic ways of obtaining the stock graph. As the stock market is an extremely dynamic system, past statistical correlations between indices are insufficient to further prediction.

To sum up, the static graph structure calculated with domain knowledge cannot represent the non-intuitive latent correlation. The dynamic graph structure is the opposite. Conducting the static and dynamic graph convolutions at the same time can consider both intuitive and latent correlations. As current methods cannot exploit both graphs, a spatiotemporal convolutional neural network involving one-dimensional and multi-graph layers termed MG-Conv is offered as a solution. The one-dimensional convolutional neural network is employed to extract deep features, and the multi-graph convolutional network incorporating static and dynamic graphs is applied for combining index relationships. The findings show that the strategy described in this paper has a considerable impact.

3. Framework

The framework of the prediction model in this paper is shown in Fig. 1. The framework of indices prediction in this paper consists of three layers: one-dimensional Convolutional Layer, graph convolutional layer, and output layer. The K-line historical data is fed into the one-dimensional convolutional layer, which performs normalization and windowing to create training and testing samples. This step will be introduced in Section 4.1. For each sample, the convolutional deep features are first extracted using a one-dimensional convolutional network. This step will be put forward in Section 4.2. The graph convolutional layer takes the convolutional features of all samples as input, and each index corresponds to a vertex in the graph.

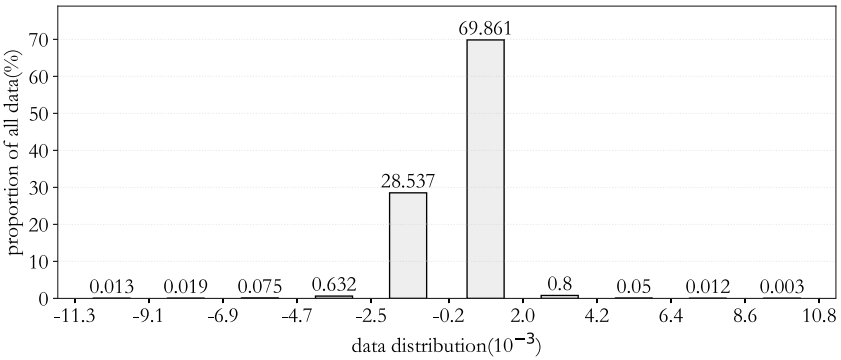


Fig. 2. Data distribution after converting index point to the scale of rise and fall. Most of the data is concentrated between -2.5×10^{-3} and 2.0×10^{-3}.

The graph convolutional layer creates two graphs: one is the static graph based on the constituent stocks of indices, and the other is the dynamic graph built dynamically based on historical transaction data. The adjacency matrices and the results of the one-dimensional convolutional layer are fed into the graph convolutional layer, which produces a prediction vector that fuses the correlation between distinct indices. In Sections 5.1 and 5.2, the graph construction and multi-graph convolution will be presented respectively. The output layer builds fully connected neural networks for prediction vectors of each index. After the output layer, the initial input of transaction data is mapped into the scale of rise and fall for the further days. The details of this framework will be introduced in the following sections.

4. One-dimensional convolutional layer

The one-dimensional convolutional layer consists of two primary steps. The first step is data normalization which converts the index point to volatility and divides the sequence data into prediction samples. This step will be introduced in Section 4.1. The second step is extracting the deep features for each prediction sample through the one-dimensional convolutional network. This part will be considered in Section 4.2.

4.1. Data normalization

The stock market's K-line data is used as the model's input in this article. A tuple $\boldsymbol{h} = (h_1, \ldots, h_5)$ represents the data of a K-line, with five items representing the starting point, highest point, lowest point, close point, and trade volume respectively. The sequence $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots \boldsymbol{h}_T\}$ represents all historical K-line data for an index, while $\boldsymbol{H}_i = \{h_{1i}, h_{2i}, \ldots h_{Ti}\}$ represents the sequence consisting of the ith element of all K-line data. It is required to translate the index points into a scale of rise and fall in order to eliminate the effects of dimensional discrepancies. The conversion procedure for the open point is as given in formula (1).

$$h_{t1}{}' = \frac{h_{t1} - h_{(t-1)4}}{h_{(t-1)4}} \tag{1}$$

Formula (2) is provided as the conversion technique for trading volume.

$$h_{t5}{}' = \frac{h_{t5} - h_{(t-1)5}}{h_{(t-1)5}} \tag{2}$$

Formula (3) represents the conversion procedure for the highest point, lowest point, and close point.

$$h_{ti}{}' = \frac{h_{ti} - h_{(t-1)i}}{h_{(t-1)i}} \tag{3}$$

Different components in the K-line have different conversion methods, as illustrated in the formulas above. The previous K-line's close point is used as the datum for the current open point. The preceding K-trading line's volume is used as the foundation for calculating the current trading volume. The current open point serves as a standard for the highest point, lowest point, and close point. The data distribution is displayed in Fig. 2 after converting all the index transaction data to the scale of rise and fall.

$$h_{ti}{}'' = \frac{h_{ti}{}' - \bar{h}_i}{pre\left(\boldsymbol{H}_i, \alpha\right) - pre\left(\boldsymbol{H}_i, 1-\alpha\right)} \tag{4}$$

As illustrated in Fig. 2, the scale of rise and fall at each K-line is quite small compared to the index point, where most data range from -2.5×10^{-3} to 2.0×10^{-3}. As a result of this property, the deep model will be difficult to train. In order to address this problem, the linear normalization approach is used to extend the data to the range of -1 to 1, as indicated in formula (4), where $\bar{h}_i = \frac{1}{T}\sum_{t=1}^{T} h_{ti}{}'$ denotes the mean of the sequence $\boldsymbol{H}_i$. $pre\left(\boldsymbol{H}_i, \alpha\right)$ is an element in $\boldsymbol{H}_i$ that satisfies $P(h_{ji} < pre\left(\boldsymbol{H}_i, \alpha\right)) = \alpha$ for each

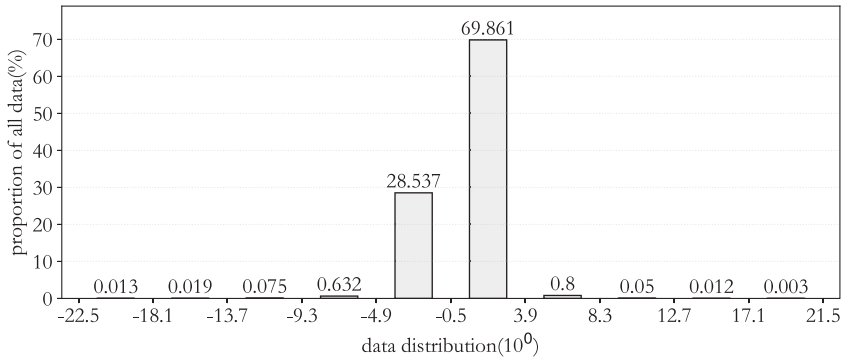


Fig. 3. Data distribution after linear normalization. Most of the data is concentrated between −4.9 and 3.9.

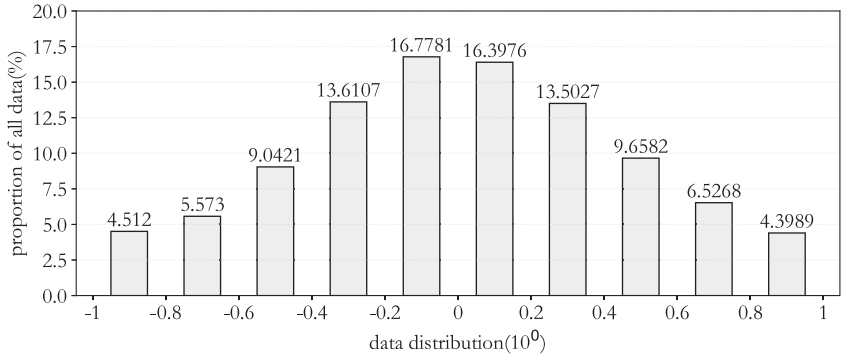


Fig. 4. Data distribution after Sigmod transformation. All data is between −1 and 1, and satisfies the normal distribution.

element h_{ji} in $\boldsymbol{H}_i$. The parameter $0.5 < \alpha < 1$ is used to control the data's normalization range. After normalization, the larger α is, the more dispersed the data will be. The data, on the other hand, is more concentrated to 0. Fig. 3 depicts the data distribution after using formula (4), where α is set to 0.7.

As illustrated in Fig. 3, over 98 percent of data is centered between −4.9 and 3.9, and a small amount of data is well outside this range. These little pieces of data will give substantial gradients during model training. A Sigmod function-based transform strategy is proposed to lessen the effect of this component data on the gradient in model training, illustrated in Formula (5),

$$\hat{h}_{ti} = 2Sigmod\left(h_{ti}''\right) - 1 = \frac{e^{h_{ti}''} - 1}{e^{h_{ti}''} + 1} \tag{5}$$

where h_{ti}'' is the output of formula (4). Following the application of formula (5), all data are evenly distributed and concentrated between −1 and 1, and the data distribution is shown as Fig. 4.

The time series data are then divided into samples using a one-day step sliding window. The average scale of rise and fall in the future a_2 days is predicted with the data of past a_1 days. The sample data $\boldsymbol{x}_t$ and related label y_t may be determined using formulas (6) and (7), where k is the K-line number of each day.

$$\boldsymbol{x}_t = (\boldsymbol{h}_{(t-1)\times k+1}, \ldots, \boldsymbol{h}_{(t-1+a_1)\times k}) \tag{6}$$

$$y_t = \frac{1}{a_2}\sum_{i=0}^{a_2-1} \boldsymbol{h}_{((t+a_1+i)\times k)4} \tag{7}$$

The sample set $\boldsymbol{S} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots (\boldsymbol{x}_\lambda, y_\lambda)\}$ is obtained when the sample division is done. The one-dimensional convolutional layer will be used to extract the deep features for each sample.

4.2. Deep feature extraction

The second step of the one-dimensional convolutional layer is extracting the convolution features for each sample. This layer's inputs are the normalized sample data, and its outputs are the extracted features. The structure of a one-dimensional convolutional network is shown in Fig. 5, which contains numerous convolution and pooling layers. The input to the jth convolutional layer is matrix $\boldsymbol{x}_{j-1}$, whose size is $\tau_{j-1} \times {n_i}^{j-1}$, where τ_{j-1} is the data length and ${n_i}^{j-1}$ is the number of input channels. The jth convolutional

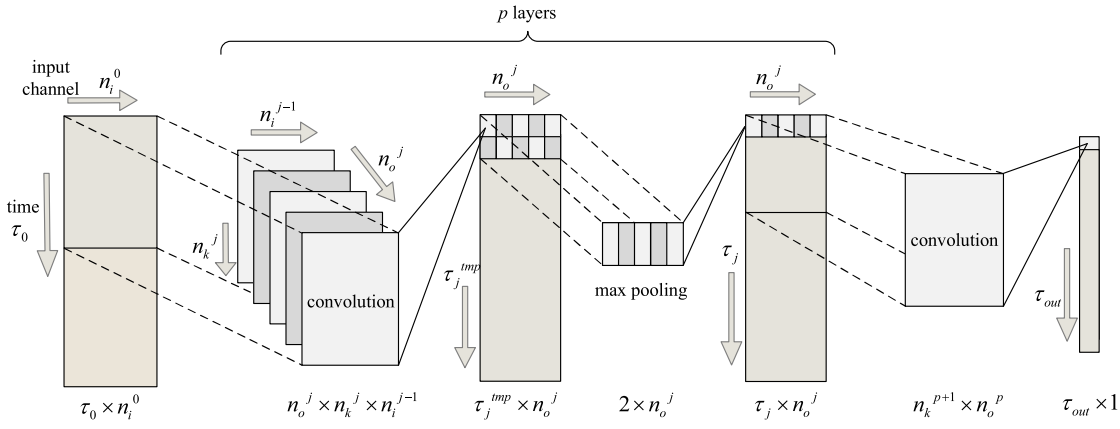


Fig. 5. The one-dimensional convolutional network. The input is transaction sequence data. p layers of convolution and max pooling are applied to extract features. The output is a τ_{out}-dimensional vector.

layer's filter is matrix $\boldsymbol{c}_j$ with the dimensions ${n_o}^j \times {n_k}^j \times {n_i}^{j-1}$, where ${n_o}^j$ is the number of output channels and ${n_k}^j$ is the filter size. During the convolutions, the convolutional kernel moves along the time direction. For each step, the convolution calculation method is shown as Formula (8),

$$\left(\boldsymbol{x}_j^{tmp}\right)_e^f = \sum_{b=e}^{e+n_k^j} \sum_{d=1}^{n_i^{j-1}} \left(\boldsymbol{x}_{j-1}\right)_b^d \left({\boldsymbol{c}_j}^f\right)_{b-e+1}^d \tag{8}$$

where ${\boldsymbol{c}_j}^f$ is the fth convolution kernel of the jth layer. The jth convolutional layer's output is ${\boldsymbol{x}_j}^{tmp}$, with a size of ${\tau_j}^{tmp} \times {n_o}^j$, where ${\tau_j}^{tmp} = \tau_{j-1} - {n_k}^j + 1$ and $\tau_0 = a_1 \times k$. Then, a maximum pooling layer is applied, shown as Formula (9).

$$\left(\boldsymbol{x}_j\right)_e^f = \max\left[\left(\boldsymbol{x}_j^{tmp}\right)_{2e-1}^f, \left(\boldsymbol{x}_j^{tmp}\right)_{2e}^f\right] \tag{9}$$

The output $\boldsymbol{x}_j$ is the matrix of size $\tau_j \times {n_o}^j$, where $\tau_j = \left\lceil {\tau_j}^{tmp}/2 \right\rceil$. The max pooling and convolutional layers are repeated p times respectively. Finally, a filter size of $1 \times {n_k}^{p+1} \times {n_o}^p$ is used to produce a one-dimensional convolution output. This output vector represents the features of transaction data for a prediction sample. N indices will produce N output vectors at time t. In order to fuse the trends of different indices, these outputs will serve as the input to the graph convolutional layer, which will be described in the next section.

5. Graph convolutional layer

The one-dimensional convolutional network was used to obtain the convolution features of different indices in the previous section. But there are trends correlations between different indices, and how to model these correlations to improve the prediction performance is a problem that needs to be solved. The multi-graph convolutional neural network is proposed in this section, and the fused features are generated. Section 5.1 introduces the construction of graph structures, including static and dynamic graphs. Section 5.2 discusses the multi-graph convolution method.

5.1. Graph construction

Building the graph structure between indices is the primary step in employing the graph convolutional neural network for index prediction. A graph is represented by $\boldsymbol{G} = \{\boldsymbol{V}, \boldsymbol{A}\}$, where $\boldsymbol{V}$ is the set of vertices and $\boldsymbol{A}$ is the set of edges represented by the adjacency matrix. Each vertex in this paper represents an index, and N denotes the total number of vertices. The adjacency matrix is expressed as $\boldsymbol{A} = \left(a_{ij}\right)_{N \times N}$ and the edges denote the link between indices. To the best of investors' knowledge, the index is a weighted sum of multiple stocks, and there are trends correlations between two indices with the same constituent stocks. The weights of the same elements between two indices can be used to build the initial adjacency matrix, commonly known as the static index graph. Assume that the set of common constituents between indexes i and j is $\boldsymbol{C}_{ij} = \left\{\left(c_{ij1}, w_{i1}, w_{j1}\right), \left(c_{ij2}, w_{i2}, w_{j2}\right), \ldots \left(c_{ijn}, w_{in}, w_{jn}\right)\right\}$, with c_{ijk} is the kth common constituent stock for indexes i and j, and w_{ik} and w_{jk} are the weight in indexes i and j, respectively. The static adjacency matrix calculation method is shown as Formula (10).

$$a_{ij} = \sum_{k=1}^{n} w_{jk} \tag{10}$$

As indicated in Formula (10), the edge a_{ij} in adjacency matrix is the proportion sum of common constituent stocks in index j. The following is an explanation of this computation method. According to the definition of the adjacency matrix, a_{ij} represents the

influence of index i on index j. When the common constituent stocks account for a substantial fraction of j, the influence of index i on j is great, and the correlation created by the same convolution kernel parameters is also stronger. As a result, the weights sum of common elements in j can be used to determine this edge. This method produces an asymmetric adjacency matrix that satisfies $0 \leq a_{ij} \leq 1$.

The static graph is determined during index compilation, and can be represented by the adjacency matrix calculated with formula (10). However, there are usually trend correlations between indices with different constituent stocks in the real stock market. For example, some industry indices usually hold negative associations, and when a certain industry index is strong, another tends to be weak. This negative association between indices must also be modeled with a graph, which in this work is referred to as the dynamic graph. The dynamic graph must be built dynamically based on historical transaction data for various indexes. Following the insight of the previous work [25], we randomly initial the graph adjacency matrix, and dynamically update it as graph convolutional model training. To reduce the training time, we decompose the dynamic matrix into multiplication of two low order matrices, which is defined as Formula (11).

$$\tilde{\boldsymbol{A}} = Softmax\left(ReLU\left(\boldsymbol{A}_1 \boldsymbol{A}_2{}^T\right)\right) \tag{11}$$

In this definition, $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are $N \times M$ matrices, and M is an integer parameter satisfying $M \ll N$. The two matrices' initial values are created randomly and will be dynamically modified during the training of the graph convolution model. The adjacency matrix sized $N \times N$ is obtained by multiplying $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. Two functions of Softmax and RuLU are used to normalize the adjacency matrix.

5.2. Graph convolution

It is necessary to perform graph convolution on the outputs of one-dimensional convolutional layer in order to fuse the relationship between trends of various indices. The calculation method for generalized graph convolution is shown as Formula (12),

$$\boldsymbol{Z} = \boldsymbol{A}^T \boldsymbol{O} \boldsymbol{W} \tag{12}$$

where $\boldsymbol{A}^T$ is the transpose of adjacency matrix, which can be either the static adjacency matrix determined in formula (10) or the dynamic adjacency matrix specified in formula (11). $\boldsymbol{W}$ is a $d \times k$ dimension convolution kernel matrix, and $\boldsymbol{O}$ is the output of the one-dimensional convolutional layer with N indices.

According to the definition of adjacency matrix, the ith row represents the influence of index i on other indices. Then the convolution of transposed $\boldsymbol{A}$ means $\boldsymbol{o}_i$ incorporates the influence of other indices. Formula (12) just denotes the basic graph convolution. But most of the time, a single graph convolution is not enough to represent the effect of various indices. To deeply fuse the correlations, we define the multiple convolutions on one graph which is denoted as Formula (13),

$$\boldsymbol{Z} = \sum_{i=0}^{K} \boldsymbol{P}^i \boldsymbol{O} \boldsymbol{W} \tag{13}$$

where $\boldsymbol{P} = \boldsymbol{A}^T / rowsum\left(\boldsymbol{A}^T\right)$ represents the transposed adjacency matrix after normalization. After these multiple convolutions, the outputs of the one-dimensional layer are deeply fused on one graph. As there are two types of graphs between indices, we should perform the graph convolution on both static and dynamic graphs. Therefore, the multi-graph convolution operation is defined as Formula (14).

$$\boldsymbol{Z} = \sum_{i=0}^{K} \boldsymbol{P}^i \boldsymbol{O} \boldsymbol{W}_1 + \tilde{\boldsymbol{P}}^i \boldsymbol{O} \boldsymbol{W}_2 \tag{14}$$

The first part of Formula (14) is graph convolution on the static graph, and the second portion is on the dynamic graph. In this formula, $\boldsymbol{W}_1$, $\tilde{\boldsymbol{P}}$ and $\boldsymbol{W}_2$ are hyperparameters that should be updated dynamically during model training. These two convolutions are undertaken respectively, and the results are added to the final output. The final output $\boldsymbol{Z}$ is a $N \times k$ matrix, where N is the number of indices and k is the feature dimension. In this matrix, each row corresponds to a certain index at the current timestamp. In order to convert this k-dimensional vector to the further volatility, N fully connected networks are built, and the outputs are the volatility of further days.

In the following sections, the prediction model is first trained with the gradient descent technique, and the trained model is saved to perform further predictions. Given the samples of all indices at time t, the prediction method would be shown in Algorithm 1.

6. Performance evaluation

The performance of the proposed MG-Conv for index trend prediction was evaluated with commonly used indices of China stock market. The experimental design is presented first, followed by comparison results of various methodologies, and finally the impact of model training iteration time outcomes.

Algorithm 1 The index prediction method.

Input:

The normalized historical transaction data $\boldsymbol{S}_t = \{\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)}, \cdots, \boldsymbol{x}_t^{(N)}\}$;

Output:

The prediction value $\hat{\boldsymbol{y}}_t = \{\hat{y}_t^{(1)}, \hat{y}_t^{(2)}, \cdots, \hat{y}_t^{(N)}\}$;

$\boldsymbol{F} = \emptyset$
for $i = 1, 2, \cdots, N$ **do**
  $\boldsymbol{f}_t^{(i)} = \mathrm{Conv}_i(\boldsymbol{x}_t^{(i)})$; // one-dimension convolution for each index
  $\boldsymbol{F} = \boldsymbol{F} \cup \boldsymbol{f}_t^{(i)}$;
end for
$\boldsymbol{F}_S = \boldsymbol{F}_D = \boldsymbol{F}$;
for $i = 1, 2, \cdots, K$ **do**
  $\boldsymbol{F}_S = \mathrm{Graph_S}(\boldsymbol{F}_S)$; // graph convolution with static graph
  $\boldsymbol{F}_D = \mathrm{Graph_D}(\boldsymbol{F}_D)$; // graph convolution with dynamic graph
end for
$\boldsymbol{F} = \boldsymbol{F}_D + \boldsymbol{F}_D$;
for $i = 1, 2, \cdots, N$ **do**
  $\hat{y}_t^{(i)} = \mathrm{FCN}_i(\boldsymbol{F}_i)$; // fully connected network
end for
return $\hat{\boldsymbol{y}}_t$

6.1. Experimental design

In the stock market of China, there are over 500 indices. Not all of these indexes are suitable for our experiment. Some indexes were only created for a brief period of time, and others were lacking a significant amount of data. After filtering, 42 commonly used indices were selected, and the 5-minute K-line transaction data ranges from 2009 to 2021 of these indices were chosen as experimental data. The index lists are shown as Table 1.

Formula (1) to (3) were used to convert the K-line data to the scale of rise and fall. Formula (4) and (5) normalized the volatility to normal distribution, and the normalization parameters $\bar{h}_i$, $pre\left(\boldsymbol{H}_i, \alpha\right)$ and $pre\left(\boldsymbol{H}_i, 1-\alpha\right)$ in formulas (4) were stored. The data was divided into samples using the one-day step sliding window. The average volatility of each index in the next a_2 days was predicted using K-line data from the previous five days. In our experiments, a_2 will take 1 to 4, and the label can be obtained using formula (7). Since one day of trading data contains 48 sets of 5-minute K-line point, the model was fed data contain 240 K-line points. The total number of samples for each index was 3012, and they were divided into training and testing sets in a 7:3 ratio.

The deep learning model was built using the pytorch framework in this article, and the optimized stochastic gradient descent technique Adam was utilized to train the model parameters. The parameters settings for our method are presented in Table 2. We employed a step-by-step training technique during model training. To begin, leave the graph convolutional layer's parameters alone and independently train the hyperparameters of each one-dimensional convolutional model. Then, keep the parameters of the one-dimensional convolutional layer intact, and train the parameters of the graph's convolutional layer. The prediction results were obtained on the testing set once the model training was done.

6.2. Comparison results of different methods

The prediction task in this work falls within the category of regression problems, which indicates the outputs of prediction models are the value of fluctuation. Small prediction errors would indicate better model performance. We chose two error criteria as the training loss function, which are mean absolute error (MAE) and mean square error (MSE). Eq. (15) to (16) include the related formulas.

$$\delta_{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_i - y_i\right| \tag{15}$$

$$\delta_{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2 \tag{16}$$

The popular methods currently used for stock and index prediction involve the fully connected neural network (ANN) [7], recurrent neural network (LSTM) [12], convolutional neural network (3D-CNN) [26], hybrid neural network (CNN-LSTM) [27],

Table 1
Indices list for performance evaluation.

Code	Name	Code	Name
000001	SSE Composite Index	000928	CSI Energy index
000002	SSE A Shares Index	000929	CSI Construction Materials Index
000009	SSE 380 Index	000931	CSI Consumer Discretionary Index
000010	SSE 180 Index	000936	CSI Communication Services Index
000016	SSE 50 Index	000937	CSI Utilities Index
000300	CSI 300 Index	000941	CSI CN Mainland New Energy Index
000925	CSI RAFI 50 Index	000949	CSI Agriculture Thematic Index
000965	CSI RAFI 200 Index	000953	CSI Local State-owned Enterprises Composite Index
399001	SZSE Component Index	000806	CSI Leading Consumption and Services Index
399005	SSE SME Composite Index	399933	CSI Health Care Index
399100	SZSE New Component Index	399934	CSI Financials and Real Estate Index
399330	SZSE 100 Index	399935	CSI Information Technology Index
399903	CSI 100 Index	399998	CSI Coal & Consumable Fuels Index
399905	CSI 500 Index	399812	CSI Old-Age Industry Index
000926	CSI Central State-owned Enterprises Composite Index	000044	SSE MidCap Index
000938	CSI Private-owned Enterprises Composite Index	000045	SSE SmallCap Index
000955	CSI State-owned Enterprises Composite Index	000046	SSE Mid&SmallCap Index
000961	CSI Upstream Resources Industry Index	000964	CSI Emerging Industries Index
000962	CSI Midstream Manufacturing Industry Index	000958	CSI Innovative and Growing Enterprises Index
000963	CSI Downstream Consumption and Services Industry Index	000922	CSI Dividend Index
000043	SSE MegaCap Index	000901	CSI Well-Off Index

Table 2
The parameter settings.

Parameter	Description	Value
a_1	History days of prediction samples	5
a_2	Further days to predict	1,2,3,4
α	Normalization parameter	0.7
N	The number of indices	42
τ_0	Input length of one-dimensional convolutional network	240
n_i^0	The initial input channel of one-dimensional convolutional network	5
n_k	The kernel size of one-dimensional convolutional network	11
n_o	The output channel of one-dimensional convolutional network	100
p	The number of convolution and max pooling layers	3
τ_{out}	The dimension of deep features	22
K	The number of graph convolution	3
b	Batch size	128
r	Learning rate	0.001
–	Max iterations	90

graph convolutional network (GC-CNN) [20], attribute-driven graph attention network (AD-GAT) [22]. This study contrasts our approach with the ones mentioned above. 22 technical indicators were chosen as input features for ANN and LSTM in the experimental design, including the rate of change, moving average, and relative strength index etc. The number of hidden layers in the ANN model was 2, and the number of neurons in the output layer was 1, which matched the predicted result. The output of the LSTM unit was input to a fully connected layer in the LSTM model, and the final output layer had one neuron. The input size for the CNN-LSTM model was 240, and the CNN module was followed by an LSTM unit. The input size for the 3D-CNN was $240 \times 5 \times 42$, and the output was a 1×42 vector corresponding to 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 are the one-dimensional convolution results, and the adjacency matrix was trained with the historical transaction data and the attribute-mattered aggregator was added to the graph convolution.

Besides, some additional methods such as One-Conv, SG-Conv, and DG-Conv were evaluated in order to deeply analyze details of our recommended method MG-Conv. The One-Conv was the one-dimensional convolution method, which excluded the graph convolutional layer from our proposed method. SG-Conv and DG-Conv added the static graph convolution and the dynamic convolution to One-Conv respectively. Figs. 6 to 9 show the average regression error for various prediction tasks, where 1 denotes predicting the next 1-day volatility of the close point, and so on. Prediction labels for different tasks can be calculated with formula (7). It is worth noting that, while the loss functions on training the model differ, the methods used to calculate the mean regression error for the test samples are both MAE.

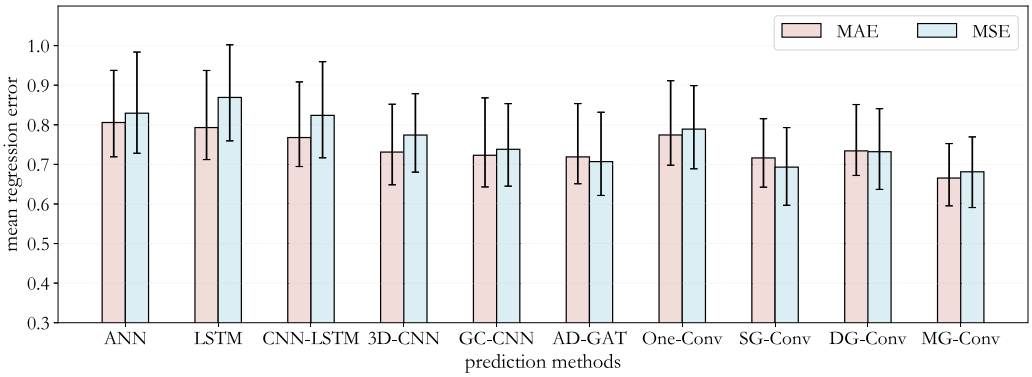


Fig. 6. Prediction results for the next 1 day. The prediction error of the state-of-the-art method AD-GAT is 0.71, and our method is 0.67.

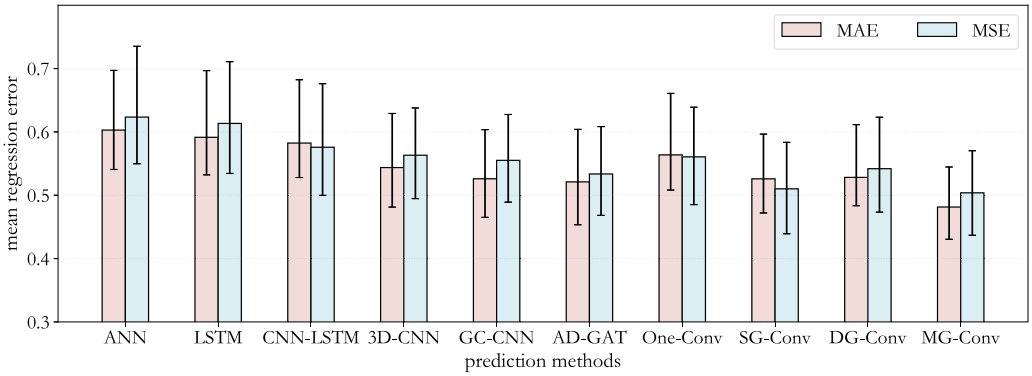


Fig. 7. Prediction results for the next 2 days. The prediction error of the state-of-the-art method AD-GAT is 0.53, and our method is 0.49.

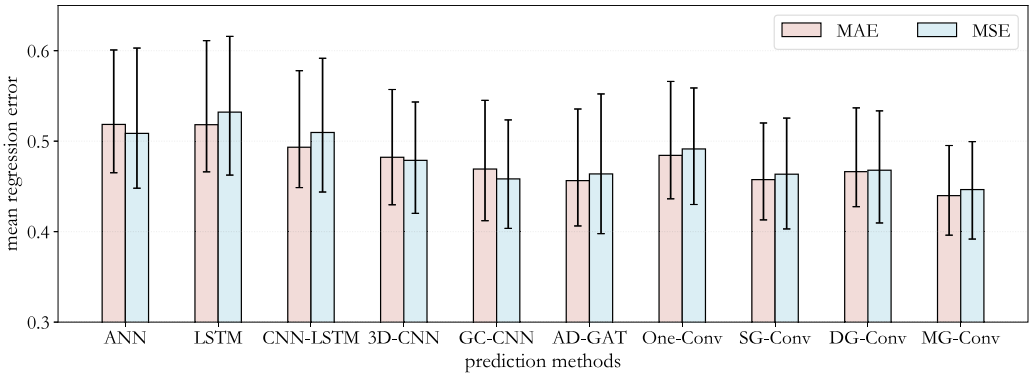


Fig. 8. Prediction results for the next 3 days. The prediction error of the state-of-the-art method AD-GAT is 0.46, and our method is 0.44.

It can be seen from these figures that the differences in loss functions have little impact on prediction results. The average prediction errors of several approaches may be observed in these figures, and they are significantly varied. The methods of 3D-CNN, GC-CNN, AD-GAT and MG-Conv outperform the first three methods significantly. The following are the key reasons. The first three algorithms use a single index as the prediction task and ignore the correlations between indices. As a result, the overall performance is subpar. On the other hand, 3D-CNN, GC-CNN, AD-GAT and MG-Conv use historical data of different indices to jointly train the model. They can balance the errors of different indices in training to minimize the total loss, which can effectively avoid the overfitting problem in training of single index historical data. As a result, these three strategies outperform the first three methods.

Although 3D-CNN evaluates the influence of various indices during training, it can only consider the local relationship between nearby indices and cannot address non-Euclidean relationships. The graph convolution is used by GC-CNN, AD-GAT and MG-Conv to depict the relationship between indices. Instead of merely evaluating the correlation between a few adjacent indexes, these three techniques can consider the association between a specific index and all other indexes along the graph. They take into account

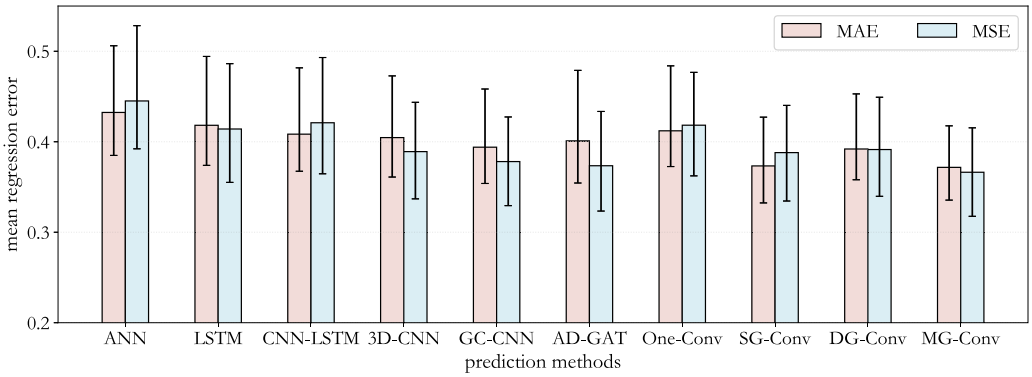


Fig. 9. Prediction results for the next 4 days. The prediction error of the state-of-the-art method AD-GAT is 0.39, and our method is 0.37.

Table 3
Prediction error reduction ratio compared with AD-GAT.

	The average results		The best results		The worst results		Overall
	MAE(%)	MSE(%)	MAE(%)	MSE(%)	MAE(%)	MSE(%)	
1 day	7.42	3.64	8.57	4.91	11.83	7.49	7.31
2 days	7.62	5.58	5.05	6.71	9.83	6.26	6.84
3 days	3.64	3.73	2.51	1.51	7.53	9.54	4.74
4 days	7.31	1.93	5.33	1.79	12.8	4.18	5.56
Overall	6.50	3.72	5.37	3.73	10.50	6.87	6.11
	5.11		4.55		8.68		

not only the correlation between indices, but also the correlation weight. As a result, they marginally outperform 3D-CNN. When comparing the following methods, it is clear that MG-Conv outperforms them all. The fundamental distinction between MG-Conv and the other two methods is the graph construction approach. The GC-CNN creates the graph using Spearman rank-order correlation, and dynamic training for the AD-GAT. Although the AD-GAT performs slightly better than GC-CNN, they both fall into dynamic graph insight. As a result, they produce similar outcomes to DG-Conv. Because the stock market is such a dynamic system, graph structures derived from previous data are insufficient to forecast future patterns. As MG-Conv uses both dynamic and static graphs, it has the best performance.

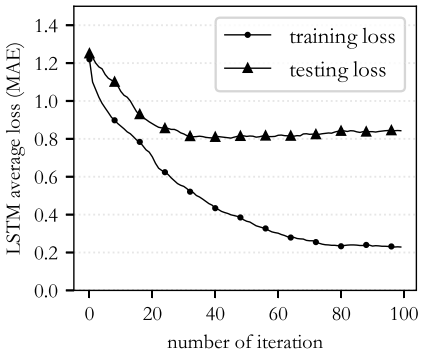
In terms of several prediction tasks, Table 3 displays the prediction error reduction ratio of our method versus the state-of-the-art AD-GAT. The average prediction error for all indexes falls by 5.11 percent, with MAE and MSE loss functions decreasing by 6.50 percent and 3.72 percent respectively. When we look at the best and worst results, we can see that our method reduces the prediction error ratio by 4.55 and 8.68 percent respectively, indicating that MG-Conv also provides improvement for poor performance samples.

6.3. The effect of iterations

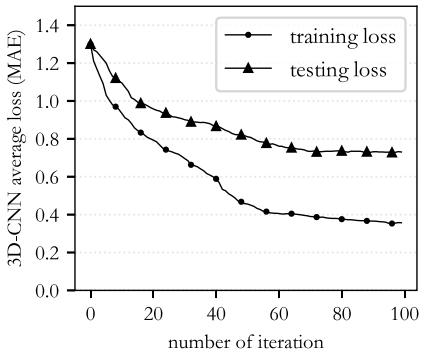
Fig. 10 depicts the curves of testing errors as the training iteration increases. It just shows the trends of four approaches as the ANN and LSTM, CNN-LSTM and 3D-CNN show similar patterns, as well as the GC-CNN and AD-GAT. As seen in Fig. 10, the training and testing errors gradually decrease as the iterations increase. However, the testing error is much higher than the training error for all four approaches. This phenomenon reveals that overfitting is a common difficulty in predicting the trend of stock movement. The key reason is that the stock market's variability makes the trend less predictable. The findings also demonstrate that enhancing the generalization ability of models is still the pressing issue in the stock market trend prediction.

These curves show that different methods require different iterations to get the lowest test error, ranging from 30 to 80 for four methods. The LSTM holds less iteration because the model just predicts a single index and the input features are extracted before model training, and it needs only a few parameters to be trained. With fewer iterations, the best prediction performance can be reached. The following three algorithms, on the other hand, must optimize convolution parameters for several indices at the same time, and the number of parameters is greater than LSTM, resulting in a slow convergence speed.

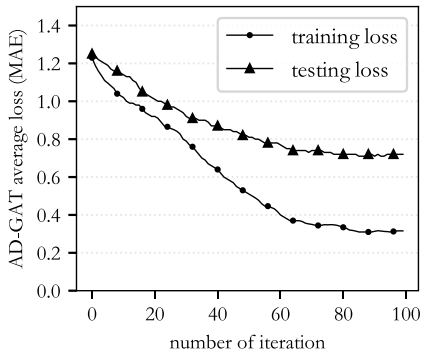
After 100 iterations, the LSTM has smaller training errors but larger testing errors than other approaches. The key reason is that this method trains the prediction model for a single index with a small amount of training data. The generalization ability of the trained model is reduced when the amount of data is limited. Instead, the other three techniques train the model with all index data, which is a significant quantity of data. Thus, they can overcome the overfitting properly. Comparing MG-Conv with 3D-CNN and AD-GAT, we can see that MG-Conv has smaller training errors than the other two approaches. It is because MG-Conv has two graph convolutions, which can more fully integrate the characteristics of various indices and reduce overfitting.



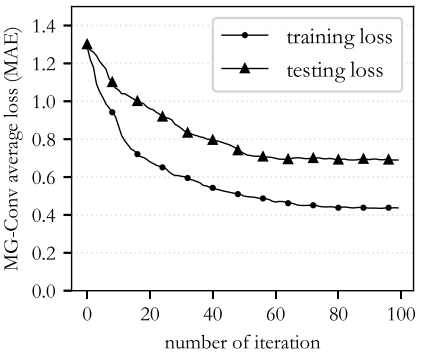
(a) LSTM. After convergence, the training loss is about 0.23 and the testing loss is about 0.8.



(b) 3D-CNN. After convergence, the training loss is about 0.36 and the testing loss is about 0.73.



(c) AD-GAT. After convergence, the training loss is about 0.32 and the testing loss is about 0.72.



(d) MG-Conv. After convergence, the training loss is about 0.44 and the testing loss is about 0.67.

Fig. 10. The effect of training iteration on the prediction results. The testing error is obviously higher than the training error.

7. Conclusions

In the realm of financial investment, predicting the trend of the stock market index is a critical topic. This research offers MG-Conv for index prediction. It first normalizes the index transaction data to a normal distribution, and convolves the normalized data to deep features with a one-dimensional convolution model. Then the static and dynamic adjacency matrices between indices are constructed. The convolved deep features and adjacency matrices are used to accomplish graph convolution. Finally, the graph convolutional result is mapped to the scale of rise and fall using a fully connected network. The proposed MG-Conv can effectively reduce prediction errors.

There are still certain issues that need to be addressed in the index forecasting. Overfitting is still a widespread problem according to the experimental results presented in this paper. Improving the model's generalization capabilities is still the most important issues. The following components can help to lessen the impact of overfitting. The first thought should redefine the index's long-term trend prediction task to replace the short-term prediction problem in this study. The second option is applying tactics such as adversarial training to improve the generalization capabilities of the model. Besides, the graph convolution approach is still an emerging technology in stock market forecasting. How to further effectively fuse the trends of different indices is still the research direction. For example, applying the mechanism of the attribute-mattered aggregator to the multi-graph convolution method would be a further avenue.

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] Lin Y-F, Huang T-M, Chung W-H, Ueng Y-L. Forecasting fluctuations in the financial index using a recurrent neural network based on price features. IEEE Trans Emerg Top Comput Intell 2020;5(5):780–91.
[3] He L, Ishibuchi H, Trivedi A, Wang H, Nan Y, et al. A survey of normalization methods in multiobjective evolutionary algorithms. IEEE Trans Evol Comput 2021;25(6):1028–48.
[4] Kumbure MM, Lohrmann C, Luukka P, Porras J. Machine learning techniques and data for stock market forecasting: a literature review. Expert Syst Appl 2022;116659.
[5] Nguyen V, Naeem M, Wichitaksorn N, Pears R. A smart system for short-term price prediction using time series models. Comput Electr Eng 2019;76:339–52.
[6] 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.
[7] Guresen E, Kayakutlu G, Daim U. Using artificial neural network models in stock market index prediction. Expert Syst Appl 2011;38(8):10389–97.
[8] Yun KK, Yoon SW, Won D. Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process. Expert Syst Appl 2021;186:115716.
[9] Surucu M, Isler Y, Perc M, Kara R. Convolutional neural networks predict the onset of paroxysmal atrial fibrillation: Theory and applications. Chaos 2021;31(11):113119.
[10] 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.
[11] Chung H, Shin K-s. Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. Neural Comput Appl 2020;32(12):7897–914.
[12] Luo J, Zhu G, Xiang H. Artificial intelligent based day-ahead stock market profit forecasting. Comput Electr Eng 2022;99:107837.
[13] 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.
[14] Saha S, Gao J, Gerlach R. A survey of the application of graph-based approaches in stock market analysis and prediction. Int J Data Sci Anal 2022;1–15.
[15] Jiang W, Luo J. Graph neural network for traffic forecasting: A survey. 2021, arXiv preprint arXiv:2101.11174.
[16] Wang S, Hu L, Wang Y, He X, Sheng QZ, Orgun MA, et al. Graph learning based recommender systems: A review. 2021, arXiv preprint arXiv:2105.06339.
[17] 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.
[18] 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.
[19] Zhong T, Peng Q, Wang X, Zhang J. Novel indexes based on network structure to indicate financial market. Physica A 2016;443:583–94.
[20] 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.
[21] 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.
[22] 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.
[23] 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.
[24] 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.
[25] Wu Z, Pan S, Long G, Jiang J, Zhang C. Graph wavenet for deep spatial-temporal graph modeling. 2019, arXiv preprint arXiv:1906.00121.
[26] Ehsan H, Saman H. CNNpred: CNN-based stock market prediction using a diverse set of variables. Expert Syst Appl 2019;129:273–85.
[27] Hao Y, Gao Q. Predicting the trend of stock market index using the hybrid neural network based on multiple time scale feature learning. Appl Sci 2020;10(11):3961.

Changhai Wang was born in Liaocheng, Shandong province, China in 1987. He received the M.S. degrees in Jiangsu University in 2012 and Ph.D. degree in Nankai University in 2016. He works in Zhengzhou University of Light Industry since 2017. His research includes deep learning, convolutional Neural Network, and financial data analysis.

Hui Liang is currently a professor of 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 Communication University of China. His current research interests include big data, deep learning and intelligent data management.

Bo Wang was born in Zhoukou, Henan province, China in 1987. He received the Ph.D. degrees from Xi'an Jiaotong University in 2017. He works in Zhengzhou University of Light Industry since 2017. His research interest includes time series data analysis, machine learning, cloud computing.

Xiaoxu Cui was born in Anyang, Henan Province, China in 1995. Her main research includes financial data forecasting, applied economics, and industrial economics.

Yuwei Xu was born in Huangshan, Anhui Province, China. He is currently an Associate Professor with the School of Cyber Science and Engineering at Southeast University, and a researcher in Purple Mountain Laboratories for Network and Communication Security, Nanjing, China. His research interests include the internet of things, future network architecture, and network security.