Information Processing and Management 60 (2023) 103480

Contents lists available at ScienceDirect

Information Processing and Management

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

Dynamic graph construction via motif detection for stock prediction

Xiang Ma [a], Xuemei Li [a], Wenzhi Feng [a], Lexin Fang [a], Caiming Zhang [a,b,*]

[a] School of Software, Shandong University, Jinan 250101, China
[b] Shandong Provincial Laboratory of Future Intelligence and Financial Engineering, Yantai 264005, China

ARTICLE INFO

Keywords:
Stock prediction
Motif detection
Dynamic graph
Stock distance

ABSTRACT

Stock trend prediction is crucial for recommending high-investment value stocks and can strongly assist investors in making decisions. In recent years, the significance of stock relationships has been gradually recognized for trend prediction, and graph neural networks (GNNs) have been introduced to capture useful features from relationships. However, applying GNNs to stock relationship analysis still faces numerous challenges, including inappropriate distance algorithms, non-dynamic stock graphs, and over-fitting. To address these challenges, we propose a dynamic graph construction module. The module offers the following advantages: (1) A dynamic graph construction module is introduced. (2) A novel stock distance algorithm based on motif detection is proposed to reduce the distance between stocks with similar trends. (3) A dynamic graph-based LSTM is proposed to aggregate the changes in historical graphs. We have conduct numerous experiments on 4503 Chinese A-share stocks, spanning 1218 trading days. Our model demonstrates 8.65% and 1.02% relative improvements in accumulated return and accuracy, respectively. In addition, the trading simulation validates that our algorithm outperforms the state-of-the-art (SOTA) algorithms in terms of profitability.

1. Introduction

The stock market is one of the largest financial markets, affecting the economic development of many countries worldwide. Whether an investor can make profit-making decisions on the market depends on the quality of the information extracted from the data obtained. Recommending stocks with higher investment values has gained popularity as a means to assist investors in making accurate decisions. Stock trend prediction methods are important and effective for stock recommendations (Feng et al., 2022).

As a typical time series, stock data can be predicted using traditional statistical analysis models (Chai, Du, Lai, & Lee, 2015). With the continuous development of artificial intelligence, deep learning models for time series are being proposed and have shown better effectiveness than statistical models (Liu, Guo, Wang and Zhang, 2022). Especially long short-term memory (LSTM) (Chaudhari & Thakkar, 2023), a well-known variant of recurrent neural networks (RNNs), has been employed in stock prediction (Liu, Ma, Li, Li and Zhang, 2022). LSTM selectively remembers historical data with three gate units to mitigate the vanishing gradient problem. However, LSTM and RNNs are limited to considering only the temporal features of the individual stock (Ma, Zhao, Guo, Li, & Zhang, 2022). The interconnectedness between companies means that the trends of stocks are not solely determined by themselves but also influenced by correlated stocks. Several studies (Ye, Zhao, Ye, & Xu, 2021) construct graphs to describe the stock relationships and utilize graph neural networks (GNNs) (Wan, Yuan, Zhan, & Chen, 2022) to analyze and extract features from the graphs. A graph comprises of two parts, nodes and edges. The nodes represent stocks, and the edges denote the relationships between the stocks. By

* Corresponding author at: School of Software, Shandong University, Jinan 250101, China.
E-mail addresses: xiangma@mail.sdu.edu.cn (X. Ma), xmli@sdu.edu.cn (X. Li), fwz@mail.sdu.edu.cn (W. Feng), fanglexin@mail.sdu.edu.cn (L. Fang), czhang@sdu.edu.cn (C. Zhang).

https://doi.org/10.1016/j.ipm.2023.103480
Received 22 March 2023; Received in revised form 10 August 2023; Accepted 14 August 2023
Available online 31 August 2023

transferring and aggregating spatial relationships between nodes, GNNs can update stock representations with relationship features to predict future trends. The commonly used GNNs is graph attention network (GAT) (Wu et al., 2020), which selectively transmit the influence of neighboring stocks instead of simply aggregating the features of all neighbors.

Constructing a graph containing rich relational information is a key process for GNNs to obtain relationship features. Existing methods can generally be categorized into prior methods and learning methods. Prior methods collect explicit relationships from various stock information for graph construction. However, the relational information are sparse and involving only a small number of stocks typically (Hsu, Tsai, & te Li, 2021). Unlike prior methods, learning methods randomly initialize graphs and optimize them by capturing implicit stock relationships in historical data, allowing the graphs to describe the relationships between all stocks. However, without prior data guidance, the learning process may not be able to effectively transfer relational features or generate graphs strongly related to trends during training, resulting in over-fitting (Woo, Liu, Sahoo, Kumar, & Hoi, 2022). There is an urgent need for an effective graph construction method to describe the relationships of all stocks and prevent over-fitting.

Furthermore, stock relationships should continuously change over time due to the influence of economic conditions, industry cycles, and company operating conditions (Tian, Zheng, Zhao, Liu, & Zeng, 2022). Most current studies construct non-dynamic stock graphs that do not respond to changes in stock relationships. Especially when the data span is long, the gap between the non-dynamic graph and real relationships will be greater, and the impact on the effect will be more serious (Jiang, Wei, Feng, Cao, & Gao, 2019). A intuitive and feasible approach for describing dynamic stock relationships is to calculate the distance between the stock price data at each time-step. Price data can directly reflect the current state of a stock and can be generated continuously over time. We can use the distance between the price data of any two stocks as the value of the edge between them for constructing stock graph. Stocks with smaller distances will have smaller edge values. Additionally, constructing graph by calculating distance of price data can obtain a complete graph, and without learning process can solve the over-fitting problem mentioned above. The price data is volatile and complex,making traditional distance methods based on Euclidean distance susceptible to significant calculation deviations (Azad, Deepak, Chakraborty, & Abhishek, 2022; Khedmati & Azin, 2020; Xu, Gao, Zhang, Tan, & Li, 2022). Therefore, the core issue is to define a distance algorithm reasonably according to the characteristics of price data, and construct dynamic graphs to adapt to changes in stock relationship.

In summary, the construction of stock graphs, as a key process in stock trend prediction, still faces three problems: (1) Limitations in calculating the stock distance. (2) Non-dynamic stock graphs fail to adapt to changes in stock relationships. (3) Overfitting during the graph training stage. To address these limitations, we propose a two-stage dynamic graph construction module to construct dynamic stock graphs. Specifically, this module first defines a motif-based stock distance algorithm (named MoDis) by calculating the distance between motifs of different stocks. The motifs are a series of forms that appears repeatedly in historical data and are determined by trends (Che, 2015). MoDis allows stocks with similar trends to have smaller distances, which is important to trend prediction. We can construct the stock graph to describe the stock relationships at each time-step. In the second stage, we propose a dynamic graph-based LSTM (DGLSTM) that analyzes and aggregates the temporal features of historical stock graphs to construct a dynamic stock graph. Unlike the vanilla LSTM, the processing object of DGLSTM is graph rather than time series. To improve training efficiency, the graph construction process is integrated into DGLSTM. While the training stage automatically learns some useful features, it also tends to fit some unimportant features, leading to a decrease in the model‘s effectiveness. Prior data can guide the model to learn key features. So, we gradually introduce prior relationships in chronological order to guide the construction of dynamic graph during training of DGLSTM. This enables the effective fusion of the prior relationships with the implicit relationships learned from the historical graphs. We conducted several experiments on 4503 Chinese A-share stocks. Our model demonstrates 8.65% and 1.02% relative improvements in accumulated return and accuracy, respectively.

The main **contributions** of this paper include:

- A dynamic graph construction module is proposed that can analyze and aggregate stock relationships at different time-steps to construct the dynamic stock graph that responds to changes in stock relationships.
- A novel price data distance algorithm called MoDis is proposed, which utilizes motifs to describe stock relationships and enables stocks with similar trends to have a smaller distance.
- The DGLSTM is proposed to aggregate the changes in historical graphs.

The remainder of this paper is organized as follows. Section 2 introduces the related works including stock relationship analysis based on GNNs, motif detection and stock recommendation. Section 3 describes the dynamic graph construction module, focusing on MoDis and DGLSTM. Some comparisons and demonstrations of the model performance are reported in Section 4. Finally, Section 5 summarizes our work and outlines the directions for future research.

2. Related work

2.1. Stock relationship analysis based on GNNs

Recently, the relationship information between stocks has proved valuable in analyzing stock trends (Chen, Jiang, Zhang, & Chen, 2021). GNN (Wan et al., 2022) have emerged as one of the most common models for capturing these relationships. The primary challenge associated with GNNs in stock trend prediction lies in the construction of the stock graph. TGC (Feng, He et al., 2019) collected and summarized industry and Wikipedia data for constructing the graph of stocks, which inspired many subsequent studies (Feng et al., 2022; Jiang, Wu, Zhao, Zhu, & Zhang, 2023). Nevertheless, the pre-defined graph has a series of problems such as non-dynamic. Fin-GAT (Hsu et al., 2021) used end-to-end graph structure learning to explore dependencies between stocks

without relying on the prior information. Due to the complexity of the stock market, the relationship learned may contain more noise. Tran (Gao et al., 2021) captured the dynamic changes of stock relations through the interaction between historical series and stock description documents. Its limitation is that the collection of stock description documents cannot ensure the completeness of information. Multi-GCCRU (Ye et al., 2021) and hierarchical attention network for stock prediction (HATS) built multiple graphs via various data and assigned them different weights to selectively aggregate information. However, the multiple graphs are still non-dynamic.

2.2. Motif detection of stock

Unlike text, temperature and traffic data, whose raw inputs already contain most features for prediction, stocks are affected by multiple factors such as the macro-economy, financial situation of a company, investor sentiments, etc (Huang, Mao, & Deng, 2021). And financial time series commonly contain high noise (Long, Lu, & Cui, 2019). To extract the motifs hidden in raw stock data, existing studies use traditional statistical and auto regressive methods to model the price data. Traditional statistical methods defined several mathematical expressions of historical prices to describe specific characteristics of presumed patterns, such as moving average (MA) (Settipalli, Gangadharan, & Fiore, 2022), momentum, Bollinger Bands, etc. However, the designed expressions are not comprehensive enough to lose lots of information. Auto regressive methods extract effective patterns through data dimension reduction and compression, such as the principal component analysis (PCA) (Leon, Rodríguez-Rodríguez, Gómez-Gasquet, & Mula, 2020; Singh & Srivastava, 2017), independent component analysis (ICA) (Back & Weigend, 1997), and locally linear embedding (LLE) (Roweis & Saul, 2000), etc. The performance depends on some transformation or kernel selection, and are very sensitive to hyperparameters.

2.3. Stock prediction

Stock prediction algorithm can offer investors valuable insights and guidance when making investment decisions, and a diverse range of methodologies has been employed. Traditional methods utilize statistical analysis to extract features from stock prices, such as the autoregressive integrated moving average (ARIMA) (Wang & Leu, 1996). The advent of RNNs (Feng, Ma, Li, & Zhang, 2023; Hochreiter & Schmidhuber, 1997) have improved the overall level of stock prediction, which can adeptly capture sequential dependencies in stock data. GNNs (Wu et al., 2020) have risen to prominence for their ability to model intricate dependencies among stocks in complex financial networks, revealing hidden correlations. But most of the current GNN-like methods use non-dynamic graphs. Thus, we propose a dynamic graph construction module for responding to the changes in stock relationships.

3. Methodology

This section formulates the stock trend prediction task (Section 3.1), and introduces the prediction framework in detail (Sections 3.3–3.4). In particular, it focuses on the motif-based distance algorithm MoDis and the construction of dynamic graphs via DGLSTM.

3.1. Problem formulation and the prediction framework

Record the price data as $X = [x^1, x^2, \dots, x^T] \in \Re^{D\times T}$. Here T is the time length of the price data and D is the length of intraday price data. The price date of day t is $x^t = [x^t_1, x^t_2, \dots, x^t_D]^\top$, and x^t_n is the nth minute price of day t. The label of prediction task can be defined as $Y = [y^1, y^2, \dots, y^T]$. y^t is the trend from day t to $t+1$, and is defined as the fluctuation of the closing price from day t to $t+1$:

$$y^t = \begin{cases} 1, & x^{t+1}_D > x^t_D \\ -1, & otherwise \end{cases} \tag{1}$$

x^t_D is the closing price (last price) at day t. $y^t = 1$ means upward trend ($x^{t+1}_D > x^t_D$), and $y^t = -1$ means flat or downward trend ($x^{t+1}_D \le x^t_D$). Flat and downward trends do not yield profits; hence, it is necessary to transfer funds from stocks predicted to exhibit such trends to more lucrative stocks. Therefore, both flat and downward trends are marked as -1. The trend prediction task is using historical data X to optimize model F and obtain future trends Y, which can be described as:

$$\hat{y}^t = F(x^{t-\tau+1}, x^{t-\tau+2}, \dots, x^t) \tag{2}$$

Here $\hat{y}^t$ is the prediction result at day t, τ is the days' number of the input and $F(\cdot)$ is the prediction model. The stocks predicted to rise will be recommended to investors for assisting their decision-making.

To solve the problems mentioned before, we built an end-to-end learning framework for stock trend prediction task, as shown in Fig. 1. N means the number of stocks. The framework consists of three module: temporal feature extraction module (Section 3.2), dynamic graph construction module (Section 3.3) and trend prediction module (Section 3.4). Temporal feature extraction module captures the time series features of historical price data via LSTM, and uses the features to build stock representations. Dynamic graph construction module is our main contribution, which is the construction of dynamic stock graph. The module includes two stages: stock graph construction of each time-step via MoDis and graph dynamization via DGLSTM. The details will be described in Section 3.3. Trend prediction module transfers and aggregates the influence between stocks in the dynamic graph by GAT to capture the relational features and update the stock representations. Finally, a fully connected (FC) layer is used to predict trends.

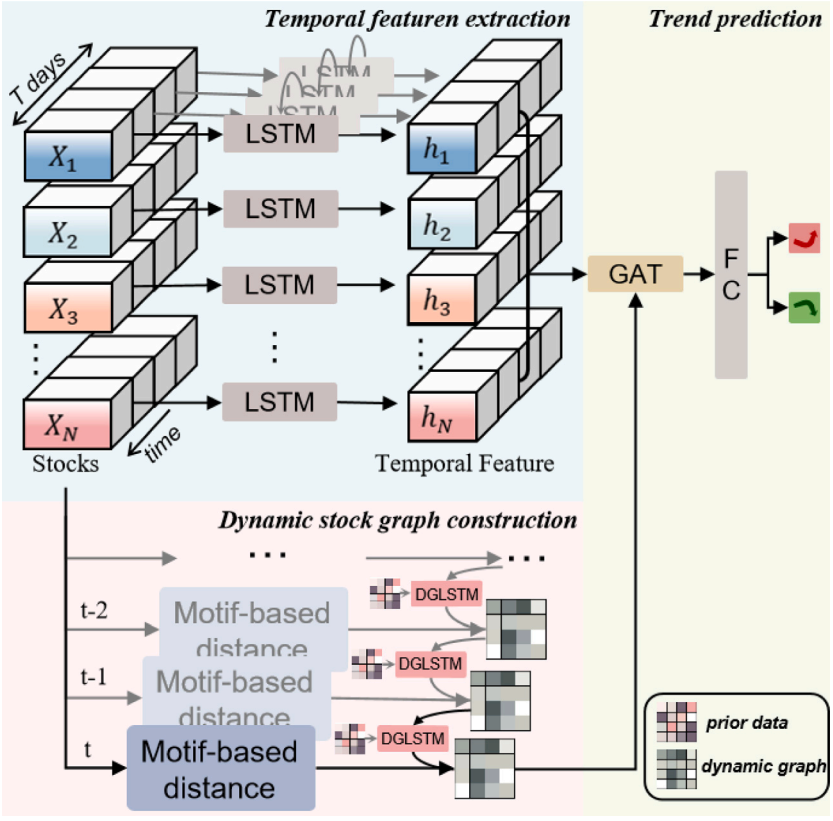

Fig. 1. Our prediction framework includes three modules: temporal feature extraction module, dynamic stock graph construction module and trend prediction module. The core of the framework is the motif-based distance algorithm (MoDis) and the dynamic graph-based LSTM (DGLSTM). Here Graph attention network (GAT) is a well-known variant of GNNs.

3.2. Temporal feature extraction

This section introduces the temporal feature extraction module. To obtain stock representations with temporal features, we adopt LSTM to analyze the price data. LSTM is a variant of RNN widely used in trend prediction. Its gated unit can suppress the vanishing gradient problem of RNN to a certain extent. For stock X, LSTM models the price data as follows:

$$\begin{aligned} i^t &= \sigma(W_i[h^{t-1}, x^t]) \\ f^t &= \sigma(W_f[h^{t-1}, x^t]) \\ \tilde{C}^t &= tanh(W_C[h^{t-1}, x^t]) \\ o^t &= \sigma(W_o[h^{t-1}, x^t]) \\ C^t &= f^t \circ C^{t-1} + i^t \circ \tilde{C}^t \\ h^t &= o^t \circ tanh(C^t) \end{aligned} \tag{3}$$

Here W_i, W_f, W_c, and W_o are parameter matrices to be learned. h^t is the hidden state of time-step t. The last hidden state is the constructed stock representation. C^t is the cell state and $\tilde{C}^t$ is the intermediate value of C^t. $\circ$ means Hadamard product. This is a general temporal features extraction process and will not be described in detail. The modeling process can be simplified into the following forms:

$$h^t = LSTM(x^{t-\tau+1}, x^{t-\tau+2}, \ldots, x^t) \tag{4}$$

τ denotes the days of the input price data.

3.3. Dynamic graph construction

After obtaining a stock representations with temporal features by Eq. (4), we need to construct the dynamic graph for capturing relational features and updating the stock representations. This section describes the process of constructing a dynamic graph,

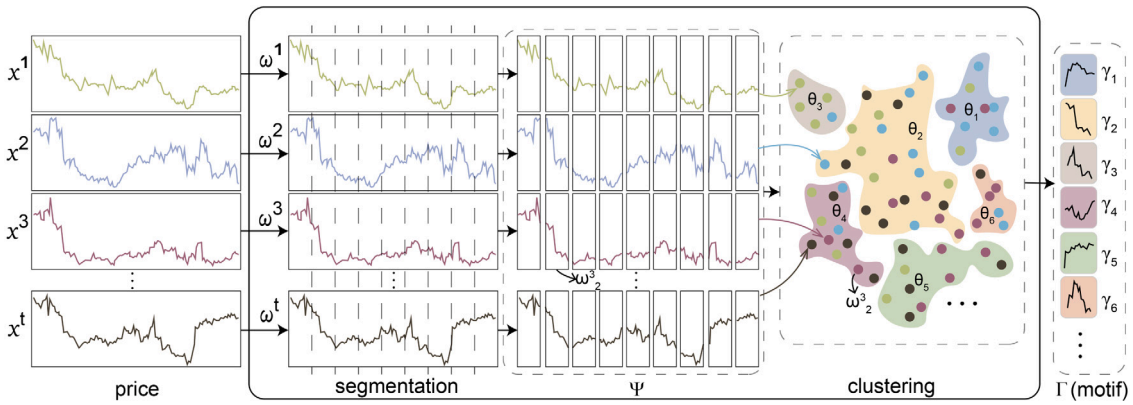

Fig. 2. The process of Motif detection. Firstly, the algorithm segments raw price data into subsequences and integrates them into Ψ. Then, we cluster the subsequences into several categories. Each point in the clustering process is a subsequence ω_i^t. Finally, the centers of categories are collected as motifs.

including stock graph construction of each time-step via MoDis and graph dynamization via DGLSTM. Specifically, we first use MoDis to calculate the distance between all stocks and obtain stock graph at each time-step. Then, DGLSTM is used to construct dynamic graph by aggregating stock graphs. The algorithm is described in detail as follows.

3.3.1. Motif-based distance algorithm (MoDis)

To describe the stock relationships, a distance algorithm of stocks is necessary. In the trend prediction task, we prefer the distance to be highly correlated with the trends. Motifs are a series of repeated short-term forms of time series on historical data (Che, 2015), which are determined by trends. The approximate motifs can be extracted from the price data of two stocks, indicating that they have a specific correlation (Gharghabi, Imani, Bagnall, Darvishzadeh, & Keogh, 2018). So we propose MoDis to calculate the distance of stocks by detecting and calculating the different between motifs.

Motif detection. Commonly, motifs can be extracted based on traditional statistical (Fuchs, Gruber, Nitschke, & Sick, 2009) or auto regressive methods (Garcia-Vega, Zeng, & Keane, 2020). However, most of these methods depends on designed expressions or kernels, and the effect is unstable (Fuchs et al., 2009). For the purpose of trend prediction, we propose a clustering based motif detection method to enhence the correlation between motifs and trends, as shown in Fig. 2.

First, the price data x^t is segmented into M subsequences with sliding windows, and obtains the subsequence set $\omega^t = [\omega_1^t, \omega_2^t, \dots, \omega_M^t] \in \Re^{L\times M}$. Here $\omega_i^t = [x_i^t, x_{i+1}^t, \dots, x_{i+L-1}^t]^\top$ is the ith subsequence of day t. L is the window length, and $M = D - L + 1$. Since time series has a certain degree of self-similarity, we can cluster the subsequences with similar trends into one category. The method integrates all subsequences ω^t into $\Psi = [\omega_1^1, \omega_2^1, \dots, \omega_M^1, \omega_1^2, \dots, \omega_M^T] \in \Re^{L\times(M\times T)}$ and clusters all elements of Ψ by k-means. Clustering can divide subsequences with approximate forms into the same category, and the clustering centers are the motifs of the stock. The motifs detected via clustering are the subsequences repeatedly appear in history, and are closely related to the trends. In other words, each category corresponds to a motif of X. The centers of categories $\Gamma = [\gamma_1, \gamma_2, \dots, \gamma_v] \in \Re^{L\times v}$ are regarded as the extracted motifs. γ_i is the center of category i and also motif i of stock X. v is the number of categories (motifs). We extract the motifs of each stock separately.

1NN distance of motifs. After getting Γ of all stocks, we can calculate the distance between Γ_i and Γ_j to get the distance of stock X_i and X_j. The elements in Γ do not have an order, and Γ_i and Γ_j may contain different numbers of elements. Therefore, the correspondence between the elements of Γ_i and Γ_j in the calculation process is uncertain. Here, we propose a nearest neighbor distance (1NN distance) to calculate the distance between Γ_i and Γ_j, as shown in Fig. 3. For $\gamma_{i,k} \in \Gamma_i$, its distance to Γ_j is the minimum value of its distance to all elements in Γ_j, that is:

$$p_{i,k} = d(\gamma_{i,k}, \gamma_{j,l}),\ \gamma_{j,l} = argmin_{z\in\Gamma_j} \|\gamma_{i,k} - z\|_2^2 \tag{5}$$

Here $\gamma_{i,k}$ is the kth motif of X_i. $\gamma_{j,l}$ is the lth motif of X_j and also the nearest neighbor (1NN) of $\gamma_{i,k}$. $p_{i,k}$ is the 1NN distance from $\gamma_{i,k}$ to Γ_j. $d(\cdot)$ means Euclidean distance. We set $P_{i\leftarrow j} = [p_{i,1}, p_{i,2}, \dots, p_{i,s_i}]$ to store the 1NN distance of all elements of Γ_i. s_i and s_j are the elements number of Γ_i and Γ_j, respectively. As shown in Fig. 4, the traditional Euclidean distance can only calculate the distance between the corresponding positions of the time series, but 1NN distance can obtain the distance of different positions between the two sequences by extracting and comparing motifs.

In addition, even if $\gamma_{j,l}$ is the 1NN of $\gamma_{i,k}$, $\gamma_{i,k}$ may not be the 1NN of $\gamma_{j,l}$. So we use $P_{i\leftarrow j}$ and $P_{j\leftarrow i}$ to represent, respectively. To make the distance measure between Γ_i and Γ_j symmetric, we combine $P_{i\leftarrow j}$ and $P_{j\leftarrow i}$ as $P_{i\leftrightarrow j} = [p_{i,1}, \dots, p_{i,s_i}, p_{j,1}, \dots, p_{j,s_j}]$.

Next, we discuss selecting an appropriate value from $P_{i\leftrightarrow j}$, as the distance of X_i and X_j. According to the definition of $P_{i\leftrightarrow j}$, if we based the distance on the largest value, the measure would become brittle at a single noisy spike or dropout that appeared in either set. At the other extreme, if we based the distance on the smallest value, there would be little discrimination between most Γ. This would be like a distance measure for English sentences that only looked at a single word in common. Since most English sentences

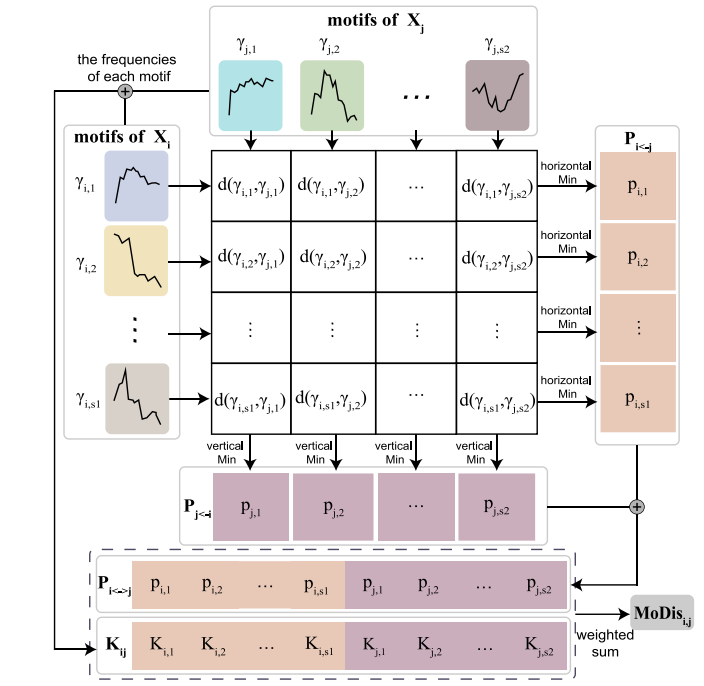

Fig. 3. The process of the nearest neighbor distance (1NN distance) of motifs. Calculating 1NN distance for motifs of all stocks and then carrying out a weighted sum based on frequency to obtain distance.

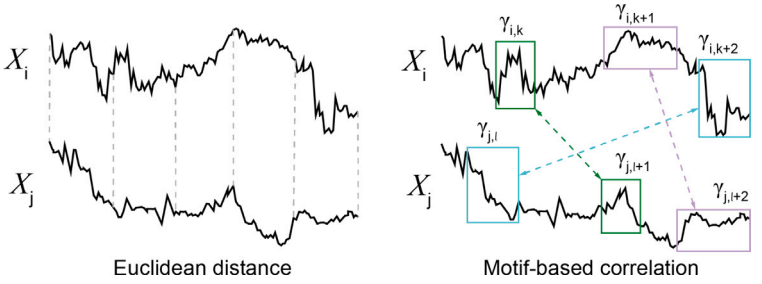

Fig. 4. Measurement strategies comparison of Euclidean distance and MoDis. MoDis extends beyond calculating the numerical differences between corresponding positions. Instead, it determines the differences based on the comparison of trend features extracted from price data.

contain "the" or "a", almost all sentences would be equidistant. To balance the influence of all motifs as much as possible, we choose to carry out the weighted sum for all elements in $P_{i\leftrightarrow j}$. The frequency of motifs appearing in history is different. Generally, the motifs with more occurrences are more important. Therefore, we use the frequency as weights to sum all elements in $P_{a\leftrightarrow b}$. The frequency of motif i is the number of subsequences contained in category i of the clustering result. We record the frequency of motifs in X_i and X_j into set $K_i = [K_{i,1}, K_{i,2}, \dots, K_{i,s_j}]$ and $K_j = [K_{j,1}, K_{j,2}, \dots, K_{j,s_j}]$, respectively. $K_{i,k}$ is the frequency of motif k from stock X_i. Similarly, we combine K_i and K_j as $K_{ij} = [K_{i,1}, \dots, K_{i,s_i}, K_{j,1}, \dots, K_{j,s_j}]$. The weighted sum function is:

$$MoDis_{i,j} = e^{-\frac{P_{i\leftrightarrow j} K_{ij}}{sum(K_{ij})}} \tag{6}$$

Here $sum(\cdot)$ is the sum function. $MoDis_{i,j}$ is the distance between X_i and X_j.

At each time-step, the stock distance can be calculated via Eq. (6), which is the edge value of the stock graph $g^t \in \Re^{s_1 \times s_2}$. $g^t_{i,j} \in g^t$ is the distance from X_i to X_j at time-step t, that is $g^t_{i,j} = MoDis^t_{i,j}$.

MoDis utilizes trend-based motifs instead of the traditional Euclidean distance measurement strategy to calculate the distance between stocks, offering the following advantages: (1) It enables comparison of trend differences among different stocks to determine the distance, rather than solely calculating numerical differences at corresponding positions. (2) It does not require data alignment operations and can handle time series of varying lengths. (3) It exhibits low sensitivity to local noise. (4) It can be processed in parallel, providing high scalability.

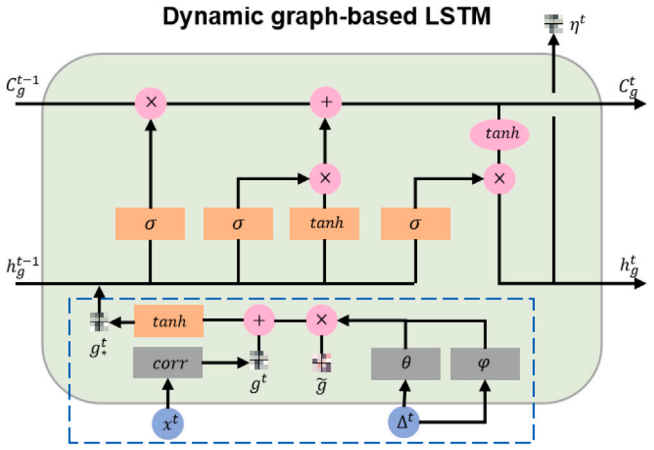

Fig. 5. Architecture of dynamic graph-based LSTM (DGLSTM). The blue dashed box marks our improvements, which is a combination process of prior relationships $\tilde{g}$ and stock graphs g^t.

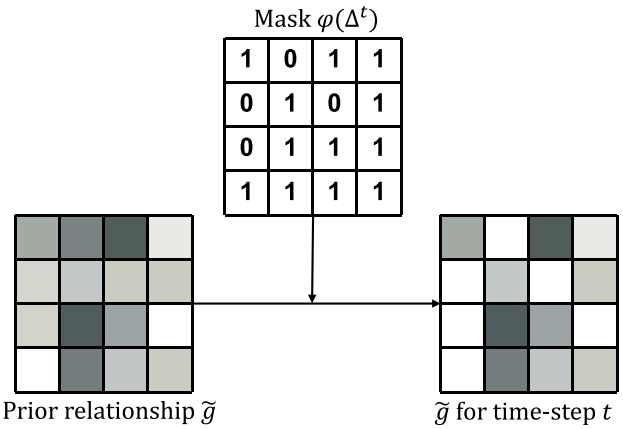

Fig. 6. An example of mask function $\varphi(\Delta^t)$. The future prior relationships are set as 0 to prevent introducing future information during training.

3.3.2. Graph dynamization via dynamic graph-based LSTM (DGLSTM)

Graph g^t is non-dynamic description of stock relationships at time-step t, which does not fully consider the changes in stock relationships. Therefore, we propose a dynamic graph-based LSTM (DGLSTM) for constructing the dynamic graph, as shown in Fig. 5. DGLSTM can integrate the changes of stock graphs in history by using gate units of vanilla LSTM. Based on the vanilla LSTM, we introduce prior data and realize the fusion of prior and implicit relationships learned from historical data, which is in the blue dashed box in Fig. 5. Although models can automatically learn valuable features during training, some unimportant features are also fitted, resulting in poor effects. Prior data can guide the model to learn some key features, making the model can better reflect the regularity of stock data and effectively prevent over-fitting.

Two problems should be paid attention to when introducing prior data: on the one hand, the algorithm needs to prevent the introduction of future information during the training phase to ensure the reliability of the model. So we use a mask function $\varphi(\cdot)$ to cover the prior data later than the current time-step. On the other hand, due to the inherent characteristics of time series, the impact of prior data on the current state decreases with the longer time interval. Therefore, we use a descending function $\theta(\cdot)$ to assign less weight to the prior data with more extended time intervals. The formula can be expressed as:

$$g_*^t = tanh(g^t + W_g\theta(\Delta^t)\varphi(\Delta^t)\tilde{g}) \tag{7}$$

Here g_*^t is the processed input data. W_g is the parameter matrix to be learned. $\tilde{g} \in \Re^{s_1 \times s_2}$ is the prior relationships, and the element $\tilde{g}_{i,j} \in \tilde{g}$ is the prior relationship of the corresponding stock X_i and X_j. $\tilde{g}$ is constructed from three types of data: industry, concept and fund-holdings. Industry and concept are commonly used terms in the stock market. Stocks belonging to the same industry or concept tend to show similar fluctuations, making them natural stock relation data. Fund holdings refer to the holdings of fund managers. The investment behavior of fund managers depends on their attention to the intrinsic value and expected value of certain stocks. In other words, if a fund manager chooses certain stocks, there might be underlying common characteristics among them. Specifically,

Table 1
Main statistics of the dataset.

Data	Statistics
Number of stocks	4503
Number of phases	6
Trading days of each phase	700
Training days of each phase	300
Validation days of each phase	100
Testing days of each phase	300
Testing days of the entire dataset	818
Number of prior relationships	44713

if stocks X_i and X_j belong to n industries or concepts together and are simultaneously held by m fund managers at the same time, the value of $\tilde{g}_{i,j}$ is determined as $\tilde{g}_{i,j} = m + n$.

Δ^t is the time interval matrix. The element $\Delta^t_{i,j} \in \Delta^t$ is the difference between the current time t and the generation time $t_{i,j}$ of $\tilde{g}_{i,j}$, that is:

$$\Delta^t_{i,j} = t - t_{i,j} \tag{8}$$

$\theta(\cdot)$ is a descending function to make the data in $\tilde{g}$ gradually reduce its influence on the current time as the time interval increases. Here $\varphi(\cdot)$ is a mask function to set the future information to 0, as shown in Fig. 6. The definition of $\varphi(\cdot)$ is:

$$\varphi(\Delta^t) = \begin{cases} 1, & \Delta^t_{i,j} > 0 \\ 0, & otherwise \end{cases} \tag{9}$$

If $t_{i,j}$ of $\tilde{g}_{i,j}$ is later than the current time t, $\tilde{g}_{i,j}$ will be masked.

The final output of DGLSTM is the dynamic graph η^t of stocks. Compared with g^t, η^t can reflect the changing process of stock relationships. Furthermore, DGLSTM also introduces $\tilde{g}$ to guide the construction of η^t to avoid over-fitting. For ease of description, the construction process can be expressed as follows:

$$\eta^t = DGLSTM(g^1_*, g^2_*, \ldots, g^t_*) \tag{10}$$

3.4. Prediction and network optimization

After obtaining dynamic graph η^t, we use the graph attention network (GAT) to aggregate the influence between neighboring stocks in η^t to capture the relationship features and update the stock representation. Compared with the traditional GNNs, GAT can selectively transmit the influence of neighbor stocks instead of simply aggregating the features of all neighbors. The process can be expressed as:

$$\varphi^t_i = \sum_j^N \eta^t_{i,j} W_\alpha h^t_j \tag{11}$$

N denotes the number of stocks. φ^t_i is the updated representation of X_i. h^t_j is the representation obtained by Eq. (4). $\eta^t_{i,j}$ means the relationship description in η^t of X_i and X_j. W_α is the parameter matrix to be learned. The above process can be expressed as:

$$\varphi^t_i = GAT(h^t, \eta^t) \tag{12}$$

Finally, we adopt a FC layer as the prediction function to analyze φ^t_i and realize prediction:

$$\hat{y}^t_i = FC(\varphi^t_i) \tag{13}$$

$\hat{y}^t_i$ is the result of our model.

We use cross-entropy to optimize the whole framework:

$$L = \sum_i y^t_i log \hat{y}^t_i + (1 - y^t_i) log(1 - \hat{y}^t_i) \tag{14}$$

L is the loss function to minimize the difference between the predicted result and ground truth. y^t_i and $\hat{y}^t_i$ are the ground truth and predicted result, respectively.

4. Experiments

This section shows the dataset, measurement, parameter setting, comparison experiments with the state-of-the-art (SOTA) algorithms, and ablation experiments. We also provide the visualization and case study, and hyperparameter analysis.

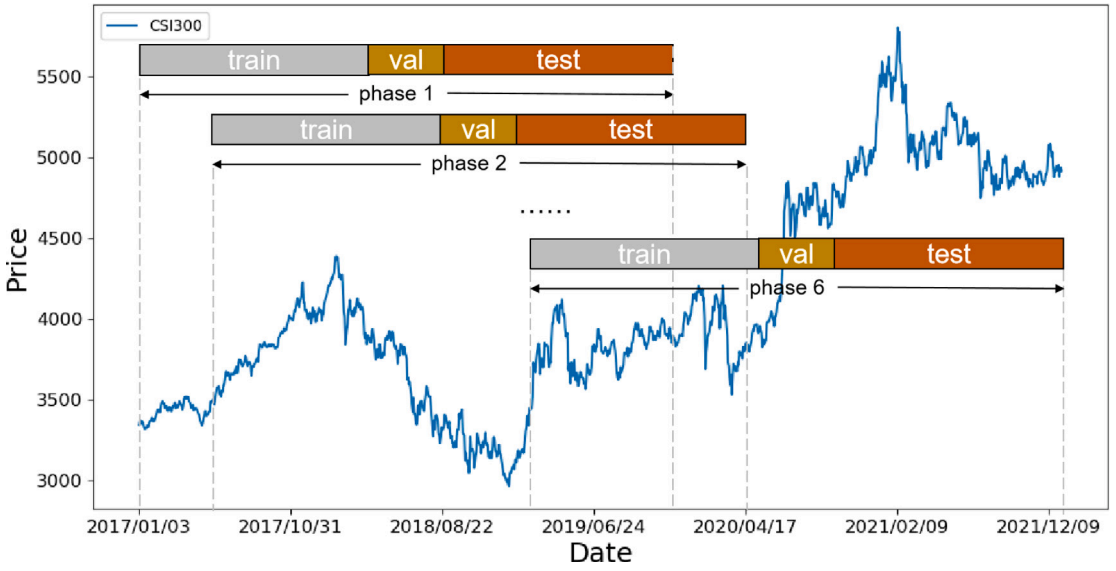

Fig. 7. Dataset arrangement for the experiments. The blue line is the price data. The whole data is divided into 6 phases. Each phase has 700 days including 300 days of training, 100 days of validation and 300 days of testing, and the sliding step is 100 days. In other words, the dataset has about 800 days of testing.

4.1. Dataset

We selected stocks from China A-share as experimental data, and the data was taken from Tushare (https://waditu.com/). The data is from 2017/1/1 to 2022/1/4, including 1218 trading days. We excluded the ST (Special Treatment) stocks to mitigate the effect of abnormal data on the comparison analysis. All comparison methods implemented identical data cleaning procedures, ensuring that the comparison results were not influenced. There were 4503 stocks left. We divided our entire dataset into 6 smaller datasets to determine whether models are effective in different volatile phases, as shown in Fig. 7. Each phase consists of 300 days of training, 100 days of validation, and 300 days of testing. The moving step of phases is 100 days. In other words, the dataset has about 800 days of testing. There are 44713 prior relationships, and the data involves 1772 stocks. Details of the dataset are presented in Table 1.

4.2. Measurement

We evaluate the methods in terms of trend prediction ability and profitability. As trend prediction is a particular type of classification task, we use the widely accepted Accuracy (ACC) and Matthews correlation coefficient (MCC) (Xu & Cohen, 2018) as the measurements for the comparison methods. ACC can express the prediction effect of the model intuitively. MCC is used to evaluate the performance of the two-class classification of the model, which is a correlation coefficient describing the actual and predicted classification. MCC ranges between +1 and −1. 1 means perfect prediction, 0 means no better than the random prediction, and −1 means complete inconsistency between prediction and observation. Each prediction result can be labeled as True Positive (TP), True Negative (TN), False Positive (FP), or False Negative (FN). So the functions of ACC and MCC are as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{16}$$

The stocks predicted to rise will be recommended to investors, so profitability is an effective tool to judge whether the recommendation is helpful and reasonable. Maximum drawdown (MDD), Accumulated return (AR), and Sharpe ratio (SR) are used to measure profitability. MDD is the maximum observed loss from a peak of trading strategies during the back-testing, and a smaller MDD indicates a lower downside risk of trading strategies. AR is an intuitive description of profitability, which is defined as:

$$AR = \sum_{t=t_1}^{t_2-1} \frac{x^{t+1} - x^t}{x^t} \times (-1)^{Action^t} \tag{17}$$

Here t_1 and t_2 are the start and end of the selected period, respectively. $Action^t$ is a binary value. If the algorithm predicts that the stock will rise, $Action^t$ is 1. Otherwise, it is 0. SR is used to measure the performance of an investment compared to its risk. The function of SR is as follows:

$$SR = \frac{E(\delta_a - \delta_f)}{std(\delta_a - \delta_f) + \varepsilon} \tag{18}$$

Table 2
The comparison of AR and ACC in different phases.

		BAH	ARIMA	LSTM	GCN	Adv-ALSTM	Fin-GAT	NVG	CA-SFCN	Price graphs	Ours
AR	Phase 1	0.1814	−0.0853	0.3284	0.3925	0.2979	0.2063	0.2942	**0.5808**	0.4357	0.3712
	Phase 2	0.1864	0.0110	0.1186	0.1988	0.2727	0.3661	0.3779	0.2902	0.4169	**0.4175**
	Phase 3	0.2239	0.1051	0.0547	0.1573	−0.0235	0.3371	**0.3489**	0.2234	0.2213	0.3321
	Phase 4	0.4890	0.4051	0.1007	0.1966	0.2742	0.6096	0.5294	0.2941	0.4168	**0.6778**
	Phase 5	0.3909	0.4479	0.1248	0.2202	0.3095	0.5477	0.5630	0.5165	0.3736	**0.5839**
	Phase 6	0.1114	0.3027	0.1207	0.1856	0.4509	0.3772	0.3169	0.3410	0.3994	**0.4643**
	ALL	0.4866	0.3568	0.4797	0.6869	0.8027	0.7433	0.9340	0.9826	1.0186	**1.1067**
ACC	Phase 1	–	0.5733	0.5100	0.5367	0.5400	0.4933	0.5400	0.5067	0.5633	**0.5745**
	Phase 2	–	0.5567	0.4933	0.5233	0.5500	0.5133	0.5600	0.5400	0.5300	**0.5653**
	Phase 3	–	0.5367	0.4467	0.4833	**0.5433**	0.5167	0.5400	0.5300	0.5067	0.5217
	Phase 4	–	0.4900	0.4567	0.4833	**0.5700**	0.5433	0.5367	0.5500	0.5233	0.5142
	Phase 5	–	0.4933	0.4633	0.5000	0.5733	**0.5867**	0.5367	0.5367	0.5500	0.5333
	Phase 6	–	0.4733	0.4533	0.4933	0.5533	**0.5900**	0.5667	0.5700	0.5333	0.5816
	ALL	–	0.5190	0.4688	0.4994	0.5496	0.5398	0.5398	0.5300	0.5398	**0.5598**

where δ_a denotes the asset return and δ_f is the risk-free rate. We used the treasury bill to calculate the risk-free rates.

4.3. Parameter setting

A grid-search method was used to find the best optimal hyperparameters. The parameter values are include sliding window length $L \in [10, 60]$, the number of categories $\upsilon \in [30, 200]$ and $\tau \in [1, 5]$. The hidden layers number and batch size of LSTM are both 64. The learning rate starts from 0.001 and is reduced by 10% after 50 epochs. Weight decay (Krogh & Hertz, 1992) and early stopping were used to avoid over-fitting. The operation of each phase was processed in parallel to save time.

Most of the experiments were performed on a 14-core server with an Intel Xeon Gold 6132 processor, 64 GB of RAM, 1T HDD, two 12 GB RTX 2080Ti GPUs, running Ubantu 18.04. We also used a PC for some experiments with an i7-10700K (8-core, up to 3.79 GHz) CPU, 512 GB of RAM, a 12 GB RTX 2080Ti GPU, and a 512 GB SSD. All the experiments were repeated five times and the results were averaged to obtain numbers in the tables.

4.4. Performance comparison

To verify the performance superiority of our model, we compare it with the following SOTA methods:

- **ARIMA** (Wang & Leu, 1996) is a classical traditional method for time series analysis.
- **LSTM** (Bao, Yue, & Rao, 2017) uses LSTM for stock trend prediction.
- **GCN** (Chen, Wei, & Huang, 2018) encodes the historical price data via LSTM and feeds the output into GCN to learn the relationship features of stocks.
- **Adv-ALSTM** (Feng, Chen et al., 2019) introduces adversarial training for attention LSTM (ALSTM) to improve the robustness of prediction.
- **Fin-GAT** (Hsu et al., 2021) is a SOTA model for stock ranking prediction based on stock relationship, which directly uses GAT to learn the potential interaction between stocks and industries automatically. We revise its target as trend prediction and retrain the model with our data.
- **NVG** (Huang et al., 2021) is a natural visibility graph extraction algorithm that proposes a moving window strategy to extract the local motif from the whole time series. Here we use their Res18-Motif model as the competitor.
- **CA-SFCN** (Hao & Cao, 2020) is a fully convolutional network incorporating cross attention (CA). CA is also used as dual-stage attention for the variable and temporal dimensions.
- **Price graphs** (Junran, Ke, Xueyuan, Shangzhe, & Jichang, 2022) uses structural information to capture long-term dependencies of the values in a time series and proposes a structural graph extraction framework from price data.

4.4.1. Prediction performance

Table 2 shows the ACC and AR in all phases. Fig. 8 reports the MCC, SR and MDD of different methods. Here BAH (buy-and-hold strategy) means investors are assumed to purchase all stocks at the beginning of back-testing and hold them until they are sold at the end. We can make the following observations:

(1) As shown in Table 2, most active trading strategies generated by different methods beat passive buy-and-hold strategies, except ARIMA and LSTM. Stock historical data is more complex than traditional time series. Insufficient regularity of price data makes it difficult for ARIMA to extract useful features. Similarly, due to the complexity, LSTM is also difficult to model price data. Fin-GAT can obtain a better return than GCN because two-level modeling can capture more useful features than traditional GCN. Adv-ALSTM has a certain effect but is not very stable. Even a negative value appears in phase 3. NVG and CA-SFCN can obtain

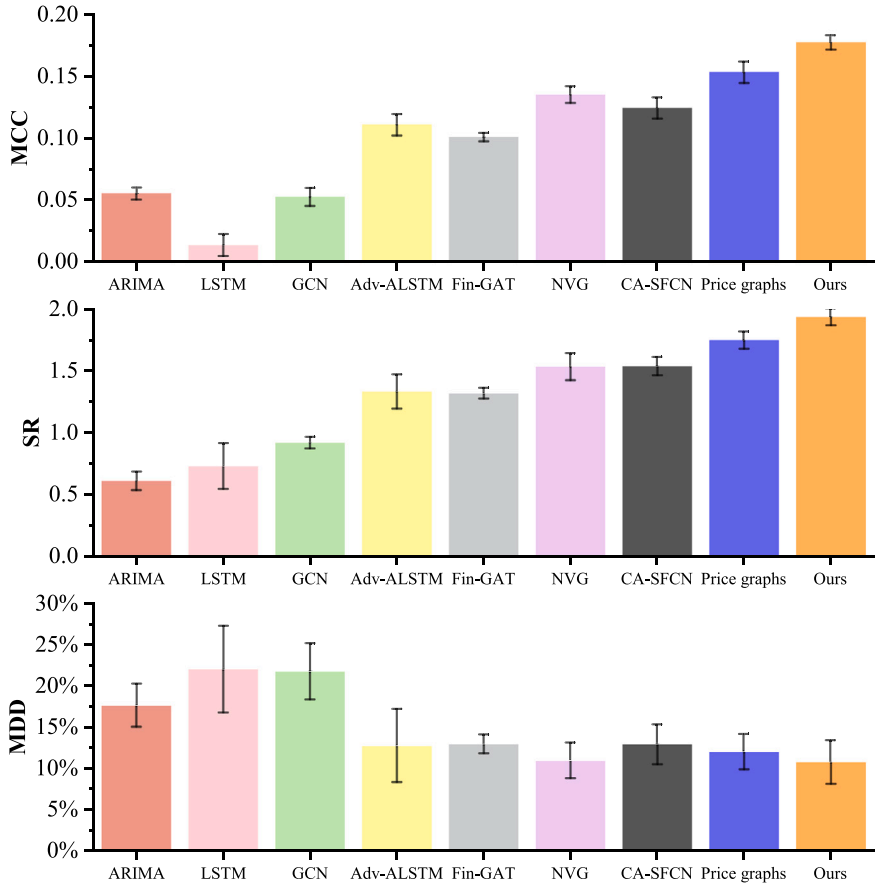

Fig. 8. Performance comparison of different methods during the back-testing. MCC indicates the classification performance, SR measures the performance of an investment compared to its risk, and MDD shows the downside risk of methods. Our model can achieve the best result in all measurements.

objective gains, but they are still fall short compared to our method. Price graphs is only second to our method and outperforms our algorithm in phase 1. Nevertheless, on the whole, our method can make more stable and even can effective profits in almost all phases.

(2) ARIMA achieves better ACC than LSTM and GCN in Table 2, proving that the traditional method can extract patterns from stock data and obtain noticeable results. The results of the Adv-ALSTM show that adversarial training can improve the robustness of the ALSTM model. Our model still makes the best result. Furthermore, we can tell that high accuracy does not mean high return (Feng, He et al., 2019) by comprehensively analyzing the results of Table 2. For example, the ACC of Adv-ALSTM is higher than Price graphs, but Price graphs obtains a higher AR.

(3) It can be seen From Fig. 8, the MCC of complex models, such as Adv-ALSTM, Fin-GAT, NVG, CA-SFCN, Price graphs and our model, is at least higher than 0.1, which means their results are stable. MCC of LSTM and GCN is even inferior to traditional ARIMA, indicating that directly applying these models to price data processing is inappropriate. The results of SR are almost proportional to the results of AR. Adv-ALSTM and Fin-GAT exceed 1, NVG and CA-SFCN can exceed 1.5. Price graphs even reaches 1.75. The SR of our method is about 10.74% higher than that of the second method (Price graphs). The MDD of complex models can be controlled below 15%, which is better than that of ARIMA, LSTM and GCN. Our model also achieves the best MDD.

(4) Our model can achieve optimal results on almost all measurements, proving that our model can sufficiently capture the features of stock trends and has more robust profitability and a more stable effect than the SOTA methods.

4.4.2. Trading simulation

To demonstrate the profitability of our model, we conducted a trading simulation on the test dataset. During the back-testing, the estimated strategy trades at a daily frequency. All models follow the trading strategy: If a rising trend of a stock price is given by our framework, we will take a long position on that stock. While if a falling trend of a stock price is predicted, we will take a short position for that stock. All stocks are equally invested and held for one day. Especially, all short and long positions are opened at the closing price on the forecast day and closed at the closing price on the following day. Accumulated profits are reinvested on the next trading day without transaction costs. Although transaction costs affect the final profit, the relative position based on

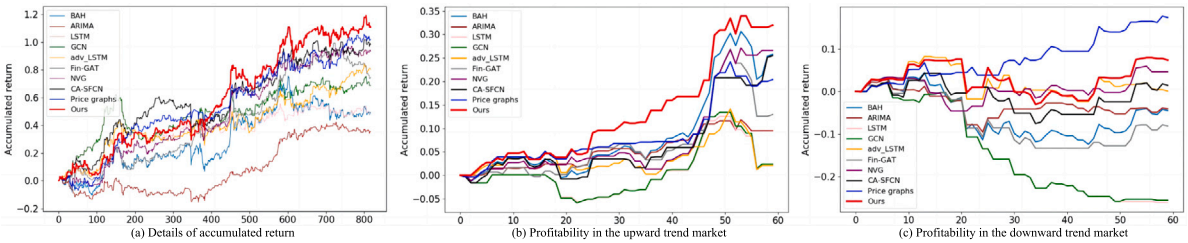

Fig. 9. Trading simulation. BAH means buy-and-hold strategy, where investors are assumed to buy all stocks at the start of the back-testing period and hold them until the end.

Table 3
The comparison of time consumption.

Method	Adv-ALSTM	Fin-GAT	NVG	CA-SFCN	Price graphs	Ours
Training (ses/stock)	23.84	19.44	18.39	19.22	17.98	18.35

Table 4
The comparison of AR and ACC in different phases between our model and its variants.

		w/o-motif	w/o-η^t	w/o-$\tilde{g}$	Ours
AR	Phase 1	0.2951	0.3597	0.3568	**0.3712**
	Phase 2	0.3336	0.3417	0.4125	**0.4175**
	Phase 3	0.2513	0.3061	**0.3320**	0.3321
	Phase 4	0.5556	0.5509	0.6751	**0.6778**
	Phase 5	0.5062	0.4613	0.5808	**0.5839**
	Phase 6	0.3408	0.3731	0.4588	**0.4643**
	ALL	0.7031	0.8218	1.0085	**1.1067**
ACC	Phase 1	0.5455	0.5333	0.5600	**0.5745**
	Phase 2	0.5367	0.5244	0.5539	**0.5653**
	Phase 3	0.4933	0.4841	**0.5200**	**0.5217**
	Phase 4	0.4833	0.4779	0.5048	**0.5142**
	Phase 5	0.5044	0.4900	0.5245	**0.5333**
	Phase 6	0.5108	0.5400	0.5704	**0.5816**
	ALL	0.5049	0.5228	0.5501	**0.5598**

model profit remain unchange due to the consistent trading strategy. For the convenience of comparison, we also provided BAH as a representation of the overall market trend.

Fig. 9(a) illustrates the accumulated profits in detail. Fig. 9(b) and (c) present the performance of all models during periods of overall upward and downward trends in the market. Based on these observations, the following conclusions can be drawn:

(1) Most models are able to outperform the market, but the traditional method ARIMA performs relatively poor performance on the test dataset. This could be attributed to the dataset's complexity and the presence of a large number of stocks. ARIMA is typically more effective in handling data with a strong regularity.

(2) Our model consistently outperforms the others, delivering the highest returns across the entire test dataset. Price graphs rank second, also demonstrating notable gains. The strength of our model primarily lies in its exceptional performance during upward trends in the stock market. While Price graphs excel in minimizing losses during downward trends, our model effectively preserves returns even when the overall market experiences a decline.

4.4.3. Efficiency analysis

This section shows the training time of models, with a particular focus on the efficiency analysis of complex models. Such models tend to be time-consuming in stock prediction tasks. Certain aspects of the graph construction process are typically one-time operations and are therefore excluded from the computational time of training. The overall time consumption of models are not significantly different as shown in Table 3. The adversarial learning mechanism in Adv-ALSTM results in a relatively time-consuming training process. Our model performs exceptionally well in training, ranking second only to Price graphs. Given the requirements of update models in real world of price prediction, we introduce incremental learning to facilitate model updates and save time. We update the model based on the original version every 10 days and achieve about 97% of the retraining effectiveness by training for 50 epochs. The update process for each stock requires only 0.5 s.

4.5. Ablation study

To demonstrate the effectiveness of some components in our model, we also compared some variants of our model:

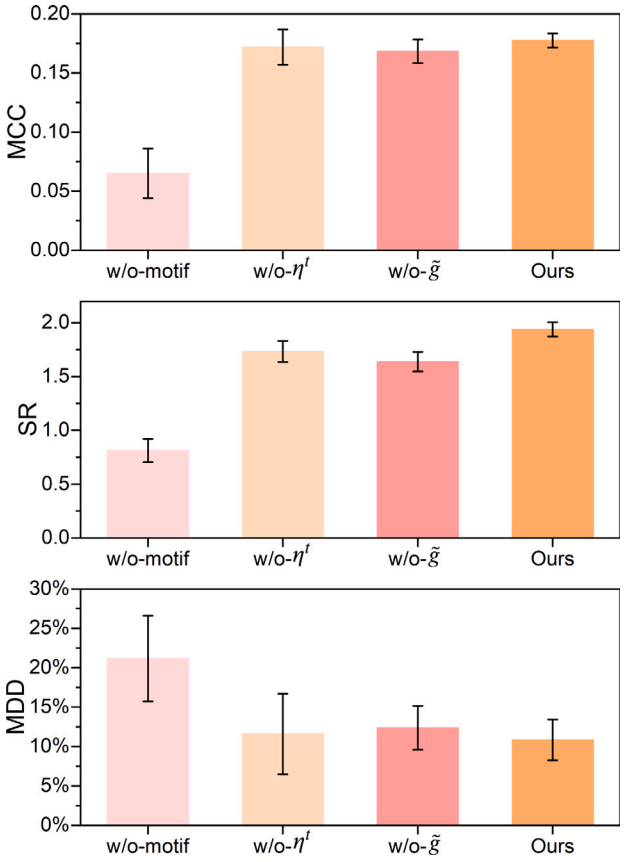

Fig. 10. Performance comparison of our model and its variants during back-testing. It can be seen that the lack of any component can lead to the degradation of model performance.

- **w/o-motif**: a variant without MoDis and uses a random graph for prediction.
- **w/o-η^t**: a variant without DGLSTM, and uses g^t to replace η^t.
- **w/o-$\tilde{g}$**: a variant without $\tilde{g}$ as guidance of graph construction.

According to the comparison results in Table 4 and Fig. 10, we have the following findings:

(1) If any of MoDis, DGLSTM and prior graph $\tilde{g}$ is missing, the variant methods become inferior to our model in all cases. This underlines the necessity of stock graph construction, dynamic graph construction and the guidance of prior data.

(2) The results indicate prior data can improve the effect but are only limited. More importantly, the missing of MoDis for constructing stock graphs significantly impacts on the model effect. The dynamic graph construction is necessary, and can enhance the effect significantly, second only to missing MoDis.

4.6. Visualization and case study

To further demonstrate the role of the dynamic graph and explore how it can improve the model effect, we illustrate a stock graph g^t of randomly selected 50 stocks and a dynamic graph η^t at the same time-step t, as shown in Fig. 11. The η^t belongs to a randomly selected representative stock (002594.SH) and Fig. 12 shows its distance change graphs with these 50 stocks over a while. From the results, we can see that:

(1) Each block in Fig. 11 is the distance of stocks corresponding to the row and column, and the color means different values. The stock graph g^t and dynamic graph η^t are not sparse, and the correlations can be obtained between any stocks, which means it contains richer relationship information than prior relationships. Furthermore, g^t and η^t are different. η^t also contains the information that before time-step t.

(2) From Fig. 12, we can find that the distance change smoothly over time, showing the temporal locality between adjacent dates. The reason is that the motifs detected from adjacent time-steps should not differ much. The higher values have been marked with red circles. The correlations in different periods vary greatly. For example, in the left figure, 000625.SH and 601633.SH are highly correlated with 002594.SH in a period. While we can see the distance between 000625.SH and 002594.SH becomes smaller in the right figure.

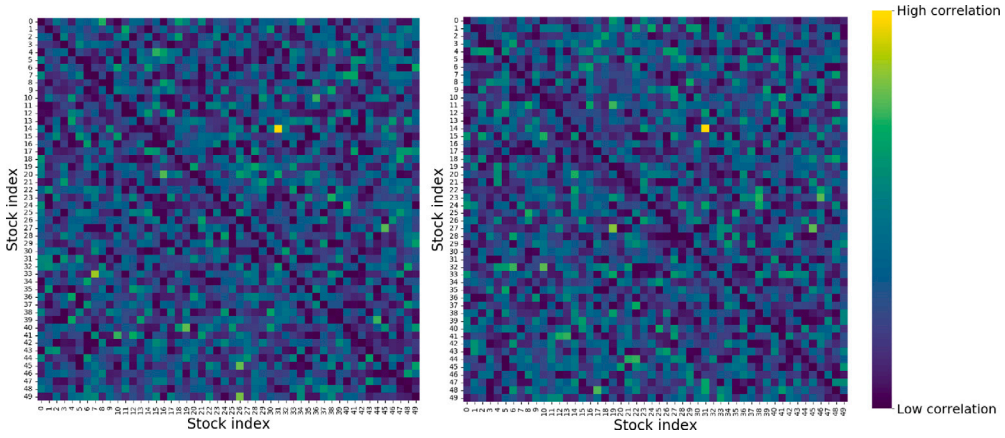

Fig. 11. Visualization of a stock graph g^t and a dynamic graph η^t at the same time-step t of randomly selected 50 stocks. Each black is the distance between the stocks corresponding to rows and columns calculated by Eq. (6). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

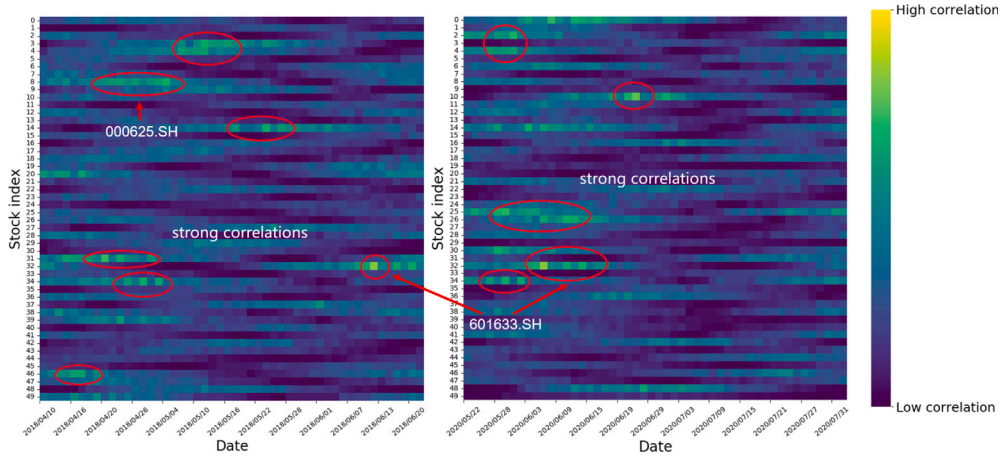

Fig. 12. Visualization of the changes of g^t. The higher values have been marked with red circles. The values of different periods are different and change smoothly over time.

In addition, Fig. 13 illustrates two examples of detected motifs. Different motifs reflect different trends. For example, motif 4 and 6 of 000506.SZ can represent a kind of upward and downward trend, respectively. Motifs of different stocks are also different, which is determined by price data.

4.7. Analysis on MoDis

This section presents several experiments to analyze the details of MoDis. These experiments include examining the differences between MoDis and Euclidean distance, comparing the use of 1NN and KNN, and evaluating the impact of frequency weighting.

4.7.1. Comparison between MoDis and Euclidean distance

We randomly selected three segments of stock data (labeled as time series a, b and c) and calculated their distances to verify the superiority of MoDis over Euclidean distance (ED). Time series a and b are from the same stock. Fig. 14 shows the results and details of ED and MoDis. In the stock graph, it is desirable for the distance between a and b to be shorter due to their common origin. However, the ED result suggests that b and c are closer, whereas MoDis correctly identifies a and b as being closer. Furthermore, it is evident that ED calculates the difference between corresponding positions, whereas MoDis does not rely solely on position correspondence, making it a more reasonable approach.

4.7.2. Comparison of MoDis using 1NN or KNN

1NN intuitively determines whether motif $\gamma_{i,k}$ of X_i appears in X_j or not, while KNN determines if $\gamma_{i,k}$ belongs to the same class as some motifs of X_j. To compare the effect of 1NN and KNN in MoDis, we employ KNN instead of 1NN for distance calculation, and

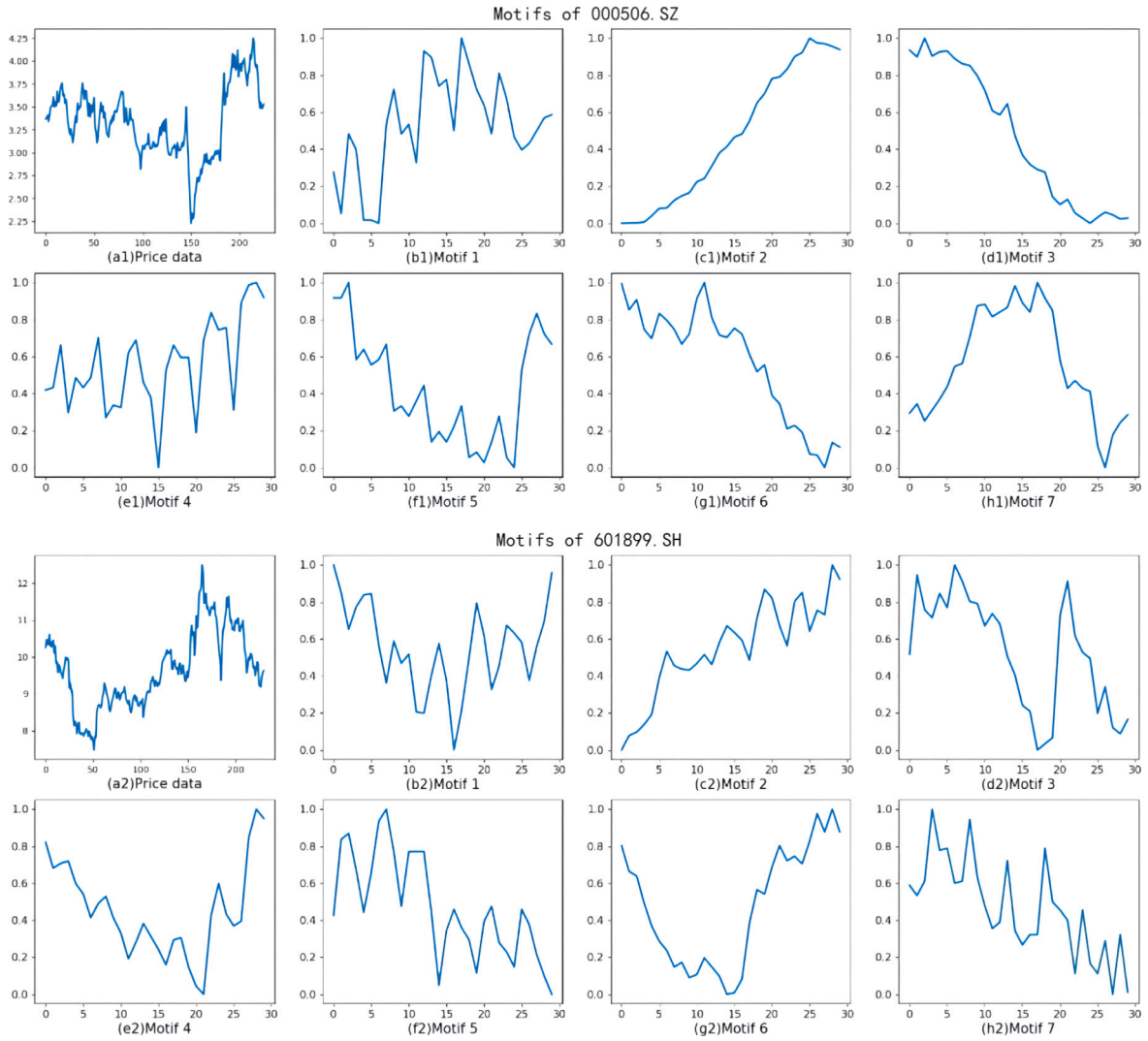

Fig. 13. Examples of detected motifs.

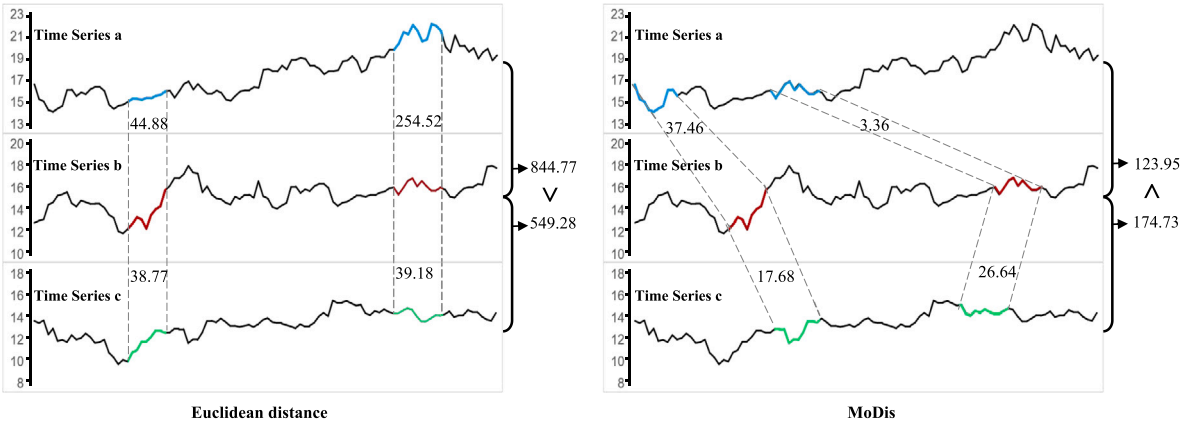

Fig. 14. Comparison between MoDis and Euclidean distance of calculating stock distance.

Table 5
The comparison of MoDis using 1NN or KNN.

	1NN	KNN (K = 3)	KNN (K = 5)	KNN (K = 10)
ACC	0.5598	0.5503	0.5512	0.5509
AR	1.1067	1.0077	1.0354	0.9843

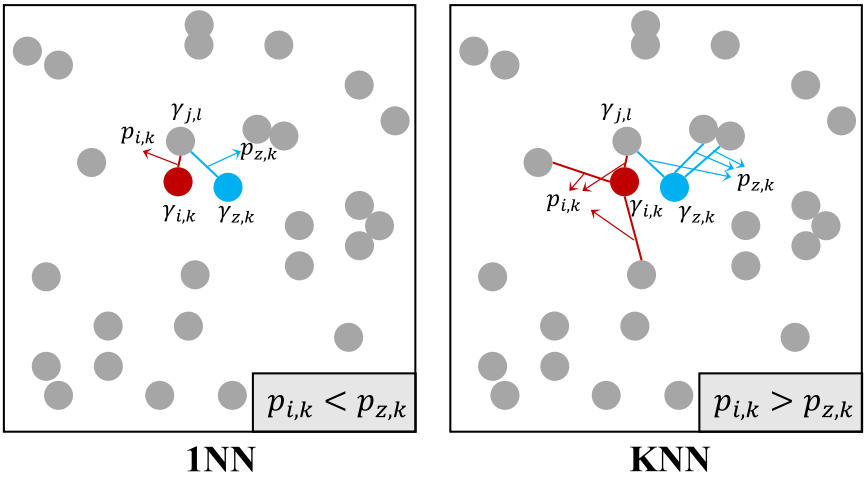

Fig. 15. Comparison between MoDis and Euclidean distance of calculating stock distance.

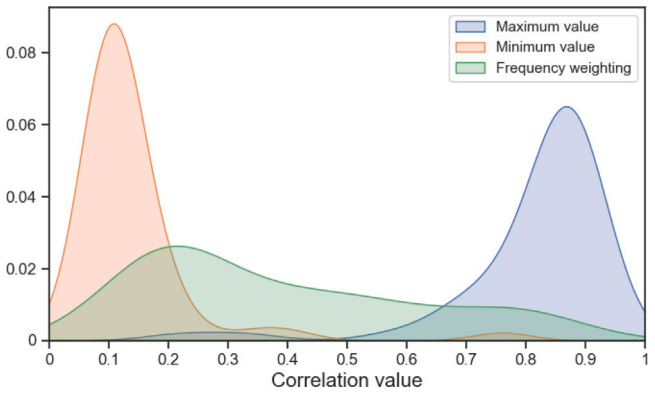

Fig. 16. The distribution of correlations obtained by different distance definitions of $P_{i\leftrightarrow j}$.

the results are presented in Table 5. The comparison reveals that KNN performs worse than 1NN. 1NN is more intuitive and suitable for this task. Furthermore, Fig. 15 illustrates the computational details of both methods. The results will be different because of the change of K. 1NN assigns a smaller distance to the closer motif $\gamma_{i,k}$, aligning with our objective of determining whether $\gamma_{i,k}$ exists in X_j. Thus, 1NN is the more appropriate choice for MoDis.

4.7.3. *Effect of frequency weighting*

Fig. 16 shows the distribution of correlations between stocks obtained by different distance definitions of $P_{i\leftrightarrow j}$. If the maximum value of $P_{i\leftrightarrow j}$ is used as the distance, the distance distribution is concentrated on the right side. If we select the minimum value of $P_{i\leftrightarrow j}$ as the distance, the distribution is obviously concentrated on the left side. The distribution of frequency weighting approach we adopted in MoDis is closer to the normal distribution, which is more reasonable than selecting an extreme value.

4.8. *Discussion*

In this section, we discuss the impact of the number of labels and input dimensions on the results in the stock prediction task.

Table 6
The comparison of binary and ternary classification.

	Ours	$\Theta = 0.002$	$\Theta = 0.01$	$\Theta = 0.1$	$\Theta = 0.2$
ACC	0.5598	0.5912	0.6345	0.6528	**0.7015**
AR	1.1067	**1.1954**	0.9744	0.8075	0.8175

Table 7
The comparison of single dimensional and multi-dimensional input.

	Ours	Ours + A	Ours + B	Ours + C	Ours + D	Ours + A + B + C + D
ACC	0.5598	0.5583	0.5598	**0.5617**	0.5600	0.5471
AR	1.1067	1.0711	1.1039	**1.1367**	1.1133	0.9055

4.8.1. Should stock prediction be approached as binary or ternary classification?

While most current methods treat stock prediction as a binary classification task, recent research (Liu et al., 2021) has shown that using three labels for classification can achieve up to 60% ACC. In order to determine the practicality of using either three or two labels, we define the prediction task as a three-label classification based on the following formula:

$$y^t = \begin{cases} 1, & x_D^{t+1} - x_D^t > \Theta \\ -1, & x_D^t - x_D^{t-1} > \Theta \\ 0, & otherwise \end{cases} \tag{19}$$

Here Θ is an adjustable threshold. The results are presented in Table 6. With the increases of Θ, ACC gradually increases, but AR shows a downward trend overall. This could be attributed to the model's tendency to misclassify small fluctuations in binary classification. While in ternary classification, small fluctuations are grouped into one category, reducing the training complexity and thereby enhancing ACC. However, this also results in the model being more conservative during back-testing. When Θ is small, it becomes possible to avoid certain erroneous operations and achieve higher profits. However, as Θ increases, the number of purchases decreases, leading to lower profits.

4.8.2. Should the input be single-dimensional or multi-dimensional?

To examine the impact of increasing input data dimensions on performance, we have introduced specific to our framework. When calculating the distance between stocks, MoDis calculates the different input dimensions separately and then uses a learnable weighted sum to obtain the final result. The additional data include 5-days moving average, 10-days moving average, turnover and traded quantity. For ease of representation, we denote these four types of data as A, B, C, and D, respectively. The results are shown in Table 7. The performance of our model remains largely unchanged when A or B is added as input. This could be attributed to the fact that the 5-day and 10-day moving averages are derived from price data, and the model might have already captured this information from the stock prices. The model's performance has improved to some extent after adding C or D, suggesting that turnover and traded quantity contain features that are not directly learnable from price data. However, when we add A, B, C, and D simultaneously, the model's performance is even worse than that of the original model. This might be due to the lack of an effective fusion method for different types of input data, resulting in the loss of valuable information. This is also a challenge when extending to multi-dimensional input. Considering that the main focus of this study is on constructing dynamic graphs, we do not delve deeper into this issue. However, this is an intriguing aspect that we intend to explore in our future research.

4.9. Hyperparameter analysis

We aim to understanding how our model performs by varying the values of different hyperparameters, including the sliding window length L and the number of categories υ, as shown in Fig. 17. When varying any of these hyperparameters, we fix other hyperparameters with default settings mentioned in Section 4.3.

Sliding window length L. $L \in [10, 60]$ is used to control the size of subsequences. The results show that the changing of L affects the performance. With the increase of L, the performance decreases first and then increases, and the best result can be acquired when $L = 40$. As L grows further, the effect becomes worse. It may be that too large a window will affect the detection of motifs.

The number of categories υ. υ ranges [30,200], and the best results can be found at $\upsilon = 150$. However, the influence of the value of υ on the result is not stable. We can get the same ACC and AR at $\upsilon = 50$ as $\upsilon = 150$. Therefore, we need select υ according to the specific situation.

5. Conclusion and future work

Stock trend prediction is a key issue in both academia and industry. To solve the problems, including the inappropriate distance algorithm, non-dynamic stock graph and over-fitting, we propose a dynamic graph construction module. The module constructs the dynamic graph using MoDis and DGLSTM. Additionally, we integrate prior knowledge and learned implicit relationships derived

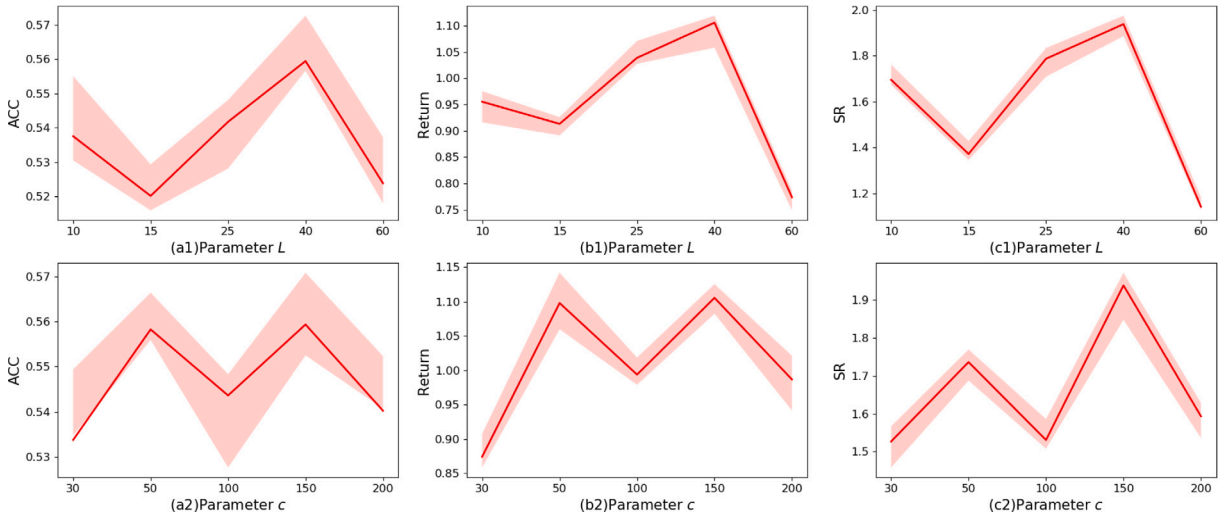

Fig. 17. The results of our model based on different hyperparameters. Each row of the figure corresponds to the parameter L and c, respectively.

from historical price data. Finally, we use GAT to transfer and aggregate the interdependencies among stocks within the dynamic graph, while a FC layer serves as the prediction function to generate results.

We utilize a dataset of 4503 stocks from China A-share to demonstrate the superiority of our algorithm. The main findings are as follows: (1) The constructed dynamic graph can respond to the changes in stock relationships promptly. (2) MoDis can give stocks with similar trends a smaller distance value. Furthermore, MoDis also provides a new and feasible idea for time series data analysis. (3) The fusion of prior and implicit relationships can help the model avoid over-fitting during model training.

Our method has showed a significantly improvement in the stock trend prediction performance, enabling the recommendation of stocks with high investment value. Nevertheless, there is still room for improvement. For instance, our future work will focus on how to apply the algorithm to the actual market transaction effectively, and how to fuse the features of multi-dimensional inputs. Meanwhile, we will explore methods to acquire more comprehensive stock relationship information, thereby enhancing the guidance provided by prior data in the learning process.

CRediT authorship contribution statement

Xiang Ma: Methodology, Software, Investigation, Writing – original draft. **Xuemei Li:** Conceptualization, Writing – review & editing. **Wenzhi Feng:** Software, Validation, Visualization. **Lexin Fang:** Software, Investigation. **Caiming Zhang:** Conceptualization, Methodology, Writing – review & editing.

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

Data will be made available on request.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported by the National Natural Science Foundation of China (NSFC) Joint Fund with Zhejiang Integration of Informatization and Industrialization under Key Project (Grant No. U22A2033) and NSFC (Grant No. 62072281).

References

Azad, H. K., Deepak, A., Chakraborty, C., & Abhishek, K. (2022). Improving query expansion using pseudo-relevant web knowledge for information retrieval. *Pattern Recognition Letters*, *158*, 148–156. http://dx.doi.org/10.1016/j.patrec.2022.04.013.

Back, A. D., & Weigend, A. S. (1997). A first application of independent component analysis to extracting structure from stock returns. *International Journal of Neural Systems*, *8*(04), 473–484.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS One*, *12*(7), Article e0180944.

Chai, J., Du, J., Lai, K. K., & Lee, Y. P. (2015). A hybrid least square support vector machine model with parameters optimization for stock forecasting. *Mathematical Problems in Engineering*, *2015*.

Chaudhari, K., & Thakkar, A. (2023). Data fusion with factored quantization for stock trend prediction using neural networks. *Information Processing & Management*, *60*(3), Article 103293.

Che, W. (2015). The motif detection of short-term tendency in stock time series. In *2015 international conference on applied science and engineering innovation* (pp. 390–395). Atlantis Press.

Chen, W., Jiang, M., Zhang, W.-G., & Chen, Z. (2021). A novel graph convolutional feature based convolutional neural network for stock trend prediction. *Information Sciences*, *556*, 67–94.

Chen, Y., Wei, Z., & Huang, X. (2018). 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* (pp. 1655–1658).

Feng, F., Chen, H., He, X., Ding, J., Sun, M., & Chua, T.-S. (2019). Enhancing stock movement prediction with adversarial training.

Feng, F., He, X., Wang, X., Luo, C., Liu, Y., & Chua, T.-S. (2019). Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, *37*(2), 1–30.

Feng, W., Ma, X., Li, X., & Zhang, C. (2023). A representation learning framework for stock movement prediction. *Applied Soft Computing*, *144*, Article 110409. http://dx.doi.org/10.1016/j.asoc.2023.110409, URL https://www.sciencedirect.com/science/article/pii/S1568494623004271.

Feng, S., Xu, C., Zuo, Y., Chen, G., Lin, F., & XiaHou, J. (2022). Relation-aware dynamic attributed graph attention network for stocks recommendation. *Pattern Recognition*, *121*, Article 108119.

Fuchs, E., Gruber, T., Nitschke, J., & Sick, B. (2009). On-line motif detection in time series with SwiftMotif. *Pattern Recognition*, *42*(11), 3015–3031. http://dx.doi.org/10.1016/j.patcog.2009.05.004.

Gao, J., Ying, X., Xu, C., Wang, J., Zhang, S., & Li, Z. (2021). Graph-based stock recommendation by time-aware relational attention network. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *16*(1), 1–21.

Garcia-Vega, S., Zeng, X.-J., & Keane, J. (2020). Stock returns prediction using kernel adaptive filtering within a stock market interdependence approach. *Expert Systems with Applications*, *160*, Article 113668. http://dx.doi.org/10.1016/j.eswa.2020.113668, URL https://www.sciencedirect.com/science/article/pii/S0957417420304929.

Gharghabi, S., Imani, S., Bagnall, A., Darvishzadeh, A., & Keogh, E. (2018). Matrix profile XII: Mpdist: a novel time series distance measure to allow data mining in more challenging scenarios. In *2018 IEEE international conference on data mining* (pp. 965–970). IEEE.

Hao, Y., & Cao, H. (2020). A new attention mechanism to classify multivariate time series. In C. Bessiere (Ed.), *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI-20* (pp. 1999–2005). International Joint Conferences on Artificial Intelligence Organization, http://dx.doi.org/10.24963/ijcai.2020/277, Main track.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hsu, Y.-L., Tsai, Y.-C., & te Li, C. (2021). FinGAT: Financial graph attention networks for recommending top-K profitable stocks. *IEEE Transactions on Knowledge and Data Engineering*, *35*, 469–481.

Huang, Y., Mao, X., & Deng, Y. (2021). Natural visibility encoding for time series and its application in stock trend prediction. *Knowledge-Based Systems*, *232*, Article 107478.

Jiang, J., Wei, Y., Feng, Y., Cao, J., & Gao, Y. (2019). Dynamic hypergraph neural networks. In *Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19* (pp. 2635–2641). International Joint Conferences on Artificial Intelligence Organization.

Jiang, J., Wu, L., Zhao, H., Zhu, H., & Zhang, W. (2023). Forecasting movements of stock time series based on hidden state guided deep learning approach. *Information Processing & Management*, *60*(3), Article 103328.

Junran, W., Ke, X., Xueyuan, C., Shangzhe, L., & Jichang, Z. (2022). Price graphs: Utilizing the structural information of financial time series for stock prediction. *Information Sciences*, *588*, 405–424. http://dx.doi.org/10.1016/j.ins.2021.12.089, URL https://www.sciencedirect.com/science/article/pii/S0020025521013104.

Khedmati, M., & Azin, P. (2020). An online portfolio selection algorithm using clustering approaches and considering transaction costs. *Expert Systems with Applications*, *159*, Article 113546. http://dx.doi.org/10.1016/j.eswa.2020.113546.

Krogh, A., & Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems* (pp. 950–957).

Leon, R.-D., Rodríguez-Rodríguez, R., Gómez-Gasquet, P., & Mula, J. (2020). Business process improvement and the knowledge flows that cross a private online social network: An insurance supply chain case. *Information Processing & Management*, *57*(4), Article 102237.

Liu, X., Guo, J., Wang, H., & Zhang, F. (2022). Prediction of stock market index based on ISSA-BP neural network. *Expert Systems with Applications*, *204*, Article 117604. http://dx.doi.org/10.1016/j.eswa.2022.117604.

Liu, T., Ma, X., Li, S., Li, X., & Zhang, C. (2022). A stock price prediction method based on meta-learning and variational mode decomposition. *Knowledge-Based Systems*, *252*, Article 109324.

Liu, G., Mao, Y., Sun, Q., Huang, H., Gao, W., Li, X., et al. (2021). Multi-scale two-way deep neural network for stock trend prediction. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence*.

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

Ma, X., Zhao, T., Guo, Q., Li, X., & Zhang, C. (2022). Fuzzy hypergraph network for recommending top-k profitable stocks. *Information Sciences*, *613*, 239–255. http://dx.doi.org/10.1016/j.ins.2022.09.010.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Settipalli, L., Gangadharan, G., & Fiore, U. (2022). Predictive and adaptive drift analysis on decomposed healthcare claims using ART based topological clustering. *Information Processing & Management*, *59*(3), Article 102887.

Singh, R., & Srivastava, S. (2017). Stock prediction using deep learning. *Multimedia Tools and Applications*, *76*(18), 18569–18584.

Tian, H., Zheng, X., Zhao, K., Liu, M. W., & Zeng, D. D. (2022). Inductive representation learning on dynamic stock co-movement graphs for stock predictions. *INFORMS Journal on Computing*.

Wan, Y., Yuan, C., Zhan, M., & Chen, L. (2022). Robust graph learning with graph convolutional network. *Information Processing & Management*, *59*(3), Article 102916.

Wang, J.-H., & Leu, J.-Y. (1996). Stock market trend prediction using ARIMA-based neural networks. In *Proceedings of international conference on neural networks (ICNN'96), Vol. 4* (pp. 2160–2165). IEEE.

Woo, G., Liu, C., Sahoo, D., Kumar, A., & Hoi, S. (2022). CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *International conference on learning representations*. URL https://openreview.net/forum?id=PilZY3omXV2.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24.

Xu, Y., & Cohen, S. B. (2018). Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: long papers)* (pp. 1970–1979).

Xu, Y., Gao, X., Zhang, C., Tan, J., & Li, X. (2022). High quality superpixel generation through regional decomposition. *IEEE Transactions on Circuits and Systems for Video Technology*, 1. http://dx.doi.org/10.1109/TCSVT.2022.3216303.

Ye, J., Zhao, J., Ye, K., & Xu, C. (2021). Multi-graph convolutional network for relationship-driven stock movement prediction. In *2020 25th international conference on pattern recognition* (pp. 6702–6709). IEEE.