Hindawi
Complexity
Volume 2024, Article ID 5791802, 12 pages
https://doi.org/10.1155/2024/5791802

Research Article

A Static-Dynamic Hypergraph Neural Network Framework Based on Residual Learning for Stock Recommendation

Jianlong Hao, Zhibin Liu, Qiwei Sun, Chen Zhang, and Jie Wang

School of Information, Shanxi University of Finance and Economics, Taiyuan 030012, China

Correspondence should be addressed to Jianlong Hao; haojianlong2012@sxufe.edu.cn

Received 9 April 2023; Revised 26 October 2023; Accepted 21 December 2023; Published 8 January 2024

Academic Editor: Dan Selişteanu

Stock ranking prediction is an effective method for achieving a high investment return and plays a crucial role in investment decisions. However, previous studies have overlooked the interconnections among stocks or have solely relied on predefined graphs for stock relationship information. The predefined graphs may not capture all possible relationships and may not be suitable for describing dynamic relationships. To address these issues, we propose a Static-Dynamic hypergraph neural network framework based on Residual Learning (SD-RL). Compared with traditional methods, SD-RL has the following advantages. (1) Stocks are no longer treated as isolated entities; instead, their static and dynamic relationship information is taken into account. (2) Leveraging the data-driven methodology, SD-RL autonomously learns both the static graph and dynamic hypergraph through dedicated graph learning and hypergraph learning modules, respectively. (3) By employing residual learning, various latent relationship information flows are mined, which enhances the stock embedding's capacity to capture trends. Extensive experiments on the real data verify the effectiveness of our proposed methods.

1. Introduction

Stock investment has evolved into a significant avenue for generating personal or institutional profits. As of the first quarter of 2022, the total market capitalization of major global stock markets has surpassed an astounding $105 trillion. Predicting stock market trends accurately is a formidable task due to the nonstationary nature and the high volatility of the market. While the feasibility of precise prediction remains controversial, decades of ongoing research in stock market prediction have yielded impressive results. To a certain extent, it suggests that the stock market is predictable.

Stock data are typically in time series format, which can be analyzed by various methods such as autoregressive models, recurrent neural networks (RNNs) [1, 2], and transformer [3]. These methods have achieved successful applications in trend prediction within the financial domain. However, many previous studies that rely on neural network methods frequently treat the prediction problem as an isolated financial time series, which overlooked the intricate relationships between stocks. Recent research has demonstrated that the spatial dimension underlying time series data in financial markets, which represents the relational information among stocks, significantly impacts the accuracy of predictive models [4–6]. There are two main types of relationships among the stocks: static relationships that remain stable over extended periods and dynamic relationships that continually evolve over short time frames. For instance, the relationship between an upstream company and a downstream company in a supply chain is a static relationship. On the other hand, dynamic relationships may emerge due to unforeseen events, such as emergencies, and disappear as the situation resolves. Graph neural networks (GNNs) are extensively employed models [7, 8] for modeling these relationships. It uses an adjacency matrix to illustrate the correlations between pairs of stocks. The nodes and edges in the graph represent stocks and their relationships, respectively. The influence between each pair of stocks can be captured by node representation learning. However, stock price movements can also be influenced by synergistic factors in real stock markets, such as industry-specific policies or common suppliers among companies. These synergistic relationships naturally group stocks together, which extends beyond individual pairs. The graphs of GNNs may not be sufficient to describe these complex relationships between stocks.

As a natural extension of GNNs, the hypergraph neural networks (HGNNs) introduce an incidence matrix to construct a relational representation of stock groups [9, 10]. The incidence matrix is indexed using hyperedges for columns and stocks (nodes) for rows. Within a hypergraph, a stock (node) can be affiliated with multiple hyperedges, signifying that this stock possesses multiple attributes. Similarly, a hyperedge can encompass multiple stocks (nodes), indicating that the stocks sharing the same hyperedge have the same properties. In contrast to GNNs, HGNNs excel in capturing a broader spectrum of synergy information. However, due to the complexity of the interaction among stocks, there are still two major challenges in this application:

(1) The construction of the adjacency matrix for the graph and the incidence matrix for the hypergraph relies on preexisting information. It can potentially lead to the omission of certain valuable relationships. Consequently, this approach may not be well suited for capturing dynamic relationships among stocks. As shown in Figure 1, the stock 300750.SZ is the leading lithium battery manufacturer. The other two stocks 300343.SZ and 603993.SH are two upstream companies in its supply chain. The price movements of the stock 300750.SZ and the stock 603993.SH exhibited a high degree of similarity during the time period before the marked red box, which attributed to the latter serving as a raw material supplier for the former. After the marked red box, all three companies' stock prices displayed similar fluctuations because stock 300343.SZ had become a significant component of the supply chain for 300750.SZ. If prior information is artificially provided, there is a risk that stock 300343.SZ might not be considered in the stock prediction for 300750.SZ. This highlights the limitations of relying solely on predetermined relationships and the importance of adapting to evolving dynamics in stock markets.

(2) Previous methods have segregated the utilization of GNNs and HGNNs. The former excels at capturing pairwise relationships, while the latter shines in capturing synergistic relationships. However, these two significant advantages provided by each model have not been combined yet.

To tackle the aforementioned challenges, we propose a static-dynamic hypergraph neural network based on residual learning framework for the stock recommendation, abbreviated as SD-RL. The overall SD-RL framework is illustrated in Figure 2. Specifically, we break down financial time series data along two dimensions: time and space. In the time dimension, we begin by feeding historical data for each stock into the GRU network. We employ an attention mechanism to discern the varying significance of different trading days. Concurrently, this network capture sequential dependencies and acquire sequential embeddings of stocks. Within the space dimension, we incorporate residual learning to uncover latent relationship information [11]. We employed a data-driven approach to learn both the inherent static relationships between stocks and the time-evolving dynamic relationships through the graph learning module and the hypergraph learning module, respectively. Ultimately, the prediction module amalgamated these latent relationship information streams derived from multiple modules (static graph and dynamic hypergraph modules) to forecast stock trends and pinpoint stocks with promising potential. In summary, our primary contributions can be summarized as follows:

(1) An end-to-end framework with automatic learning graph structures is proposed for modeling financial time series data and stock recommendations. The proposed framework is more general than existing spatiotemporal graph neural networks because it can handle the financial time series without a predefined graph structure.

(2) The inherent static graph structure is learned in a data-driven manner. An efficient hypergraph generation algorithm is proposed to capture the dynamic relationship of stocks evolving over time. The prediction module fuses multiple relationship information flows and realizes the recommendation task.

(3) Evaluation experiments of the proposed framework have been conducted on two real-market datasets. The results demonstrate the effectiveness and rationality of our approach.

The remainder of this paper is organized as follows. In Section 2, we review the related work and highlight the unresolved problem. Section 3 presents the technical details of the proposed framework. Experimental setups are described in Section 4. Section 5 concludes this study and outlines future research directions.

2. Related Work

This section provides a review of traditional stock forecasting methods and the existing literature on graph neural networks. Hypergraph learning is also briefly discussed.

In the field of academic research on stock forecasting, two main types of methods have been explored: statistical methods and machine learning methods. Statistical models, like the autoregressive moving average model, the autoregressive integrated moving average model, and its variants, have been used to forecast moving average stock prices. However, these early works relied on handcrafted features, which frequently led to predictions lagging behind the actual price movements. In recent years, machine learning algorithms such as logistic regression (LR) and support vector machine (SVM) have shown promising advancements in stock forecasting. With the continuous enhancement of

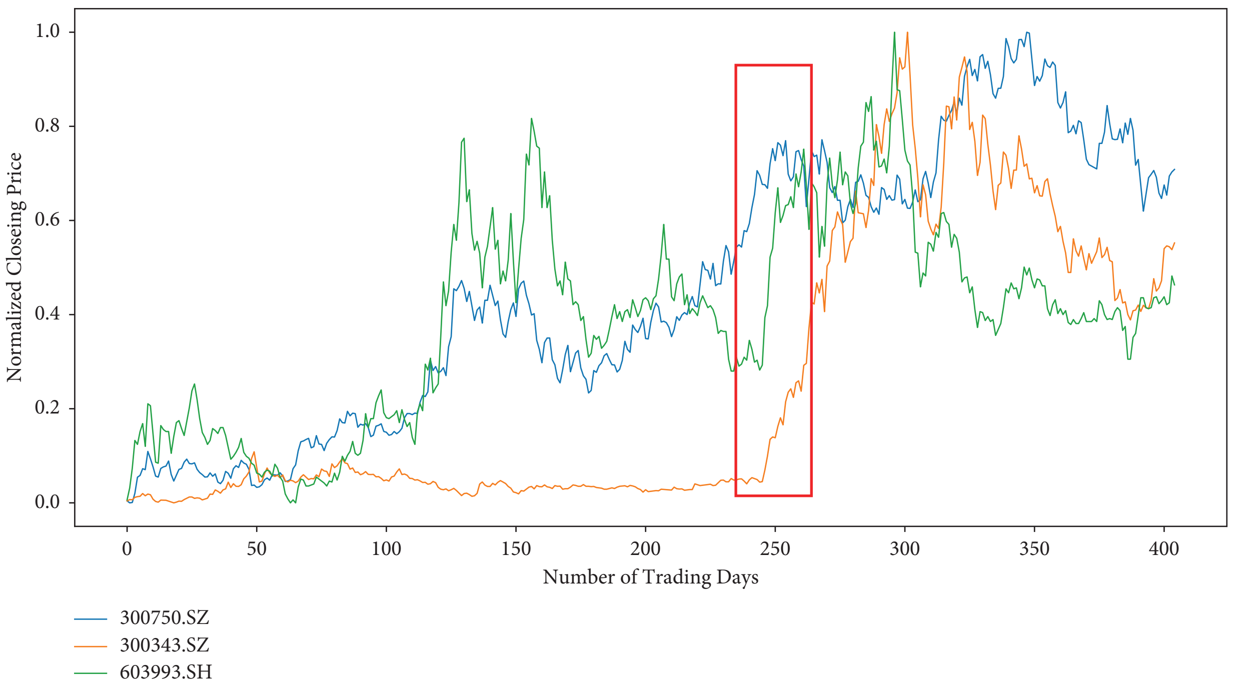

Figure 1: Price volatility patterns of target stocks and related stocks in China's A-share market.

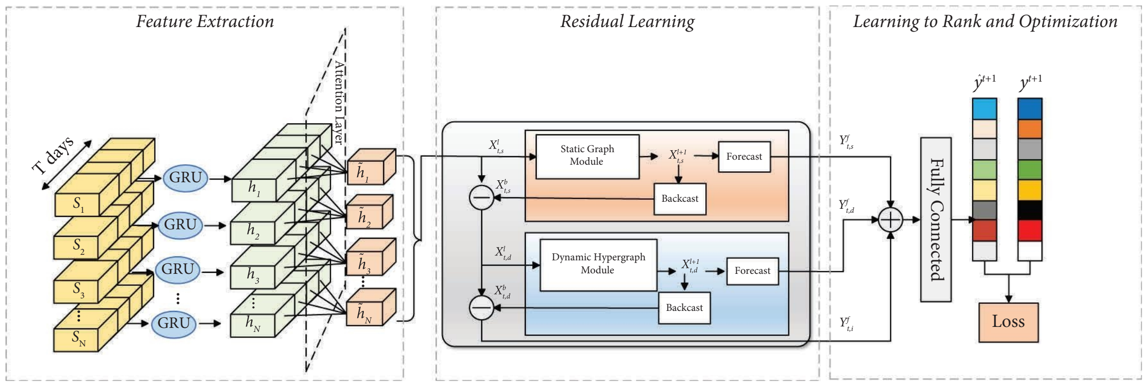

Figure 2: The overall architecture of our framework.

computing power, various deep learning techniques have been applied to stock market forecasting. Nelson et al. [12] utilized the long short-term memory (LSTM) network to predict future stock trends by incorporating historical prices and technical indicators. Shen et al. [13] employed a gated recurrent unit (GRU) network and replaced its traditional output layer with support vector machine (SVM). Feng et al. [14] introduced an adversarial attention LSTM approach, which utilized adversarial training to simulate the stochasticity in the model training process. It enhanced the stock movement prediction. Zhang et al. [15] presented a method based on state frequency memory (SFM) to decompose the hidden state of LSTM memory cells into multiple frequency components for identifying multifrequency trading patterns. Lin et al. [16] introduced a temporal routing adapter (TRA) based on an optimal transfer algorithm, enabling the learning of multiple trading patterns in the stock market data to further enhance stock forecasting performance. Additionally, the transformer architecture proposed by Ding et al. [3] combined multiscale Gaussian priors with a self-attention mechanism to model temporal context information, and it had demonstrated remarkable results in stock trend prediction.

Traditional stock forecasting methods frequently fall short in leveraging the intricate interplay among stocks. Chen et al. [17] innovatively established networks of interconnected companies based on real-market investment events. Then, they enhanced the relationship between stocks through graph convolutional neural networks, resulting in more precise prediction outcomes. Based on this, Feng et al. [4] framed stock prediction as a ranking task. They proficiently addressed the interplay among distinct stocks by aggregating company metadata and encoding time-sensitive stock relationships. Similarly, Xu et al. [18] harnessed graph-based techniques to model pairwise relations among stocks by utilizing sector industry metadata and key business data of companies. Additionally, Cheng et al. [19] meticulously crafted a heterogeneous graph by amalgamating events, news, relationships, and market data within a knowledge graph. This multimodal input fusion significantly bolstered the financial prediction capabilities of the model. However, one limitation of the graph-based methods is their reliance on prior knowledge for constructing static relationships. To address this constraint, various endeavors have been made to uncover hidden relationships in graphs [20, 21].

In recent times, hypergraph learning has witnessed substantial advancements in addressing problems, which involve relationships among data extending beyond pairwise interactions. Applications span diverse domains, including visual object recognition [22], traffic prediction [23], recommender systems [24], and social networks [25]. In the context of stock recommendation tasks, the utilization of hypergraphs involves categorizing stocks into multiple groups and refining their representations. Sawhney et al. [5] introduced a spatiotemporal hypergraph convolutional network (STHGCN) based on the predefined industrial relationships. This innovative method combined gated temporal convolution with hypergraph convolution in the spectral domain. It enabled capturing the evolution of stock prices and relationships in a spatiotemporal-aware manner. To further enhance the propagation of hypergraph information, Cui et al. [26] put forth a hypergraph triple attention network for stock trend prediction based on the foundation of STHGCN. This method augmented stock trend prediction by explicitly modeling group industry affiliations among stocks through a hypergraph attention module. Moreover, Li et al. [27] presented a reinforcement learning approach grounded in hypergraph-based methods for stock portfolio selection. This technique enhanced investment selection by incorporating a hypergraph attention module to represent industry affiliations effectively among stocks.

Based on the research mentioned earlier, it becomes evident that the majority of existing graph-based or hypergraph-based deep learning methods necessitate the incorporation of domain-specific prior knowledge. However, the construction of these models heavily relies on resource-intensive strategies for acquiring relations. Additionally, a limitation of existing hypergraph neural networks is their lack of specialization for temporal learning involving time-evolving features, such as daily stock prices. Hence, in this paper, we adopt a data-driven approach to learn the inherent static connections among stocks and design an efficient hypergraph construction algorithm to capture the evolving dynamic relationships over time. As a result, we achieve the recommendation task by integrating information flows from both static and dynamic time series sources.

3. Framework

In this section, we present a static-dynamic hypergraph neural network framework based on residual learning for end-to-end stock recommendation. As shown in Figure 2, this framework comprises three parts: feature extraction, residual learning, and ranking and optimizations.

3.1. Problem Formulation. Let a stock set $S = \{s_1, s_2, \ldots, s_N\}$ denote N individual stocks. We collect the historical price records of each stock in the past L days $\mathrm{X} \in \mathbb{R}^{N\times L\times F}$. Each stock can be represented as $\mathrm{X}^{\mathbf{i}} = \{[\mathbf{x}_1^{\mathbf{i}}, \mathbf{x}_2^{\mathbf{i}}, \ldots, \mathbf{x}_{\mathrm{L}}^{\mathbf{i}}], s_i \in S\} \in \mathbb{R}^{L\times F}$, where $\mathbf{x}_{\mathbf{t}}^{\mathbf{i}} \in \mathbb{R}^F$ is the set of price features of stock s_i on the t-th trading day, and F is the number of original features (such as opening price, closing price, highest price, and lowest price).

We formulate stock recommendation as a learning-to-rank problem. The true ranking score is defined as the magnitude of a stock's rise and fall $y_i^{t+1} = (c_i^{t+1} - c_i^t)/c_i^t$, where c_i^t is the closing price of the stock s_i on trading day t. Given $\mathrm{X}_{\mathbf{t}} \in \mathbb{R}^{N\times F}$, the purpose of our model is to learn a function $f(\mathrm{X}_{\mathbf{t}}, \mathbf{\Theta})$, which maps $\mathrm{X}_{\mathbf{t}}$ to the ranking scores, and get a score ranking list $\hat{y}^{t+1} = \{\hat{y}_1^{t+1}, \hat{y}_2^{t+1}, \cdots, \hat{y}_N^{t+1}\}$, where $\mathbf{\Theta}$ are the parameters of the model. It recommends top N stocks to investors.

3.2. Temporal Feature Extraction. Historical stock price data have demonstrated their efficacy in forecasting future stock price trends [28]. To capture the time series evolution characteristics of individual stocks, we employ the Attentive GRU model. The Attentive GRU comprises two essential components: the GRU layer and the temporal attention mechanism. In contrast to conventional RNN models, this configuration excels in capturing long-range dependencies while maintaining a concise structure.

3.2.1. GRU Layer. For each stock (take stock s_i as an example), the GRU Layer is used to map $[\mathbf{x}_1^{\mathbf{i}}, \mathbf{x}_2^{\mathbf{i}}, \ldots, \mathbf{x}_{\mathrm{L}}^{\mathbf{i}}] \in \mathbb{R}^{L\times F}$ to $[\mathbf{h}_1^{\mathbf{i}}, \mathbf{h}_2^{\mathbf{i}}, \ldots, \mathbf{h}_{\mathrm{L}}^{\mathbf{i}}] \in \mathbb{R}^{L\times F_h}$, where F_h is the dimension of the *hidden representations.*

3.2.2. Temporal Attention Layer. The attention mechanism is widely employed in various sequence learning problems [3, 14]. Over the last L trading days, the *hidden representations* of different time steps have varying degrees of influence on the *overall hidden representations* of the sequence. As a result, we utilize the attention mechanism to combine the *hidden representations* at different time steps in the following manner:

$$\widetilde{\mathbf{h}}_{\mathrm{t}}^{\mathrm{i}} = \sum_{t=1}^{L} \alpha_t^i \mathbf{h}_{\mathbf{t}}^{\mathrm{i}},$$

$$\alpha_t^i = \frac{\exp^{\mathrm{sim}\left(\mathbf{h}_{\mathrm{t}}^{\mathrm{i}}, \mathbf{h}_{\mathrm{L}}^{\mathrm{i}}\right)}}{\sum_{m=1}^{L} \exp^{\mathrm{sim}\left(\mathbf{h}_{\mathrm{m}}^{\mathrm{i}}, \mathbf{h}_{\mathrm{L}}^{\mathrm{i}}\right)}}, \tag{1}$$

$$\mathrm{sim}\left(\mathbf{h}_{\mathbf{t}}^{\mathrm{i}}, \mathbf{h}_{\mathrm{L}}^{\mathrm{i}}\right) = \left(\mathbf{W}_{\mathbf{k}}\mathbf{h}_{\mathbf{t}}^{\mathrm{i}}\right)^T \left(\mathbf{W}_{\mathbf{q}}\mathbf{h}_{\mathrm{L}}^{\mathrm{i}}\right),$$

where $\mathbf{W}_{\mathbf{k}}$ and $\mathbf{W}_{\mathbf{q}} \in \mathbb{R}^{F_h\times F_h}$ are parameters to be learned. Finally, the *overall hidden representations* of N stocks in the past L days $\widetilde{\mathbf{h}_\mathbf{t}^\mathbf{1}}, \widetilde{\mathbf{h}_\mathbf{t}^\mathbf{2}}, \ldots, \widetilde{\mathbf{h}_\mathbf{t}^\mathbf{N}}$ are obtained. $\widetilde{\mathbf{h}_\mathbf{t}}$ is a matrix of the *overall hidden representations* of N stocks, which represents the current state of each company under the movement of the stock price, $\mathbf{X}_{\mathbf{t,h}} = \widetilde{\mathbf{h}_\mathbf{t}}, \widetilde{\mathbf{h}_\mathbf{t}} \in \mathbb{R}^{N\times F_h}$.

3.3. Static Graph Module

3.3.1. Static Graph Learning. The graph module can be seen as a connecting bridge for associating stocks in both temporal and spatial dimensions. Prior research has empirically demonstrated that static graphs learned through graph learning methods outperform predefined graphs [29]. Motivated by this, we aim to leverage graph learning methods to acquire static stock graphs. This process can be described as follows:

$$\begin{aligned} \mathbf{M_1} &= \tan h(\alpha \mathbf{E_1}\boldsymbol{\theta}_1), \\ \mathbf{M_2} &= \tan h(\alpha \mathbf{E_2}\boldsymbol{\theta}_2), \\ \mathbf{A} &= \varphi\left(\mathbf{M_1}\mathbf{M_2^T}\right) = \text{LeakyReLU}\left(\tan h\left(\alpha\left(\mathbf{M_1}\mathbf{M_2^T}\right)\right)\right), \end{aligned} \tag{2}$$

where $\mathbf{A}$ is the adjacency matrix of the static graph; $\mathbf{E_i} \in \mathbb{R}^{N\times F_e}$ represents the randomly initialized stock embeddings; $\boldsymbol{\theta}_\mathbf{i} \in \mathbb{R}^{F_e\times F_e}$ are learnable parameters; $\tan h$ and LeakyReLU are activation functions; and α is a hyperparameter for controlling the saturation rate of the activation function.

Because companies with different market capitalization have different impacts on the market, static stock relations should be asymmetric. We retain the positive relations among the learned static relations to get the adjacency matrix $\mathbf{A}^+$.

$$\begin{aligned} \mathbf{C} &= \mathbf{A} + \mathbf{A}^\mathbf{T} + \mathbf{I}, \\ A^+_{\text{ij}} &= \begin{cases} A_{\text{ij}}, & C_{\text{ij}} \geqslant 0, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

3.3.2. Static Graph Convolution. To ensure that the stock nodes preserve their original information during the information propagation process, we retain a portion of the original hidden state of the stock time series, which is illustrated as follows:

$$\mathbf{X_{t,s}^{l+1}} = \left(\lambda \mathbf{X_{t,s}^{l}} + (1-\lambda)\mathbf{D}^{-1}\overline{\mathbf{A}}^+ \mathbf{X_{t,s}^{l}}\right)\mathbf{W_s}, \tag{4}$$

where $\mathbf{X_{t,s}^l} \in \mathbb{R}^{N\times F_h}$ represents the input of the lth static graph convolution layer at time step t; $\mathbf{X_{t,s}^0} = \mathbf{X_{t,h}}$; $\overline{\mathbf{A}}^+ = \mathbf{A}^+ + \mathbf{I}$; $D_{\text{ii}} = \sum_{j=0}^{N} \overline{A}^+_{\text{ij}}$; $\mathbf{D}$ is degree diagonal matrix; and $\mathbf{W_s} \in \mathbb{R}^{F_h\times F_h}$ is a learnable parameter matrix.

We feed the output of static graph convolution $\mathbf{X_{t,s}^{l+1}}$ into two fully connected layers with LeakyReLU activation functions $\sigma(\cdot)$ to generate backcast $\mathbf{X_{t,s}^b}$ and forecast $\mathbf{Y_{t,s}^f}$:

$$\begin{aligned} \mathbf{X_{t,s}^b} &= \sigma\left(\mathbf{X_{t,s}^{l+1}}\mathbf{W_s^b}\right), \\ \mathbf{Y_{t,s}^f} &= \sigma\left(\mathbf{X_{t,s}^{l+1}}\mathbf{W_s^f}\right), \end{aligned} \tag{5}$$

where $\mathbf{W_s^b}$ and $\mathbf{W_s^f}$ are parameters to be learned.

3.4. Dynamic Hypergraph Module

3.4.1. Hypergraph. To model the stock market hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, we regard each stock as a node. Each hyperedge $e \in \mathcal{E}$ represents a subset of related stocks $\{s_1, s_2, \ldots, s_k\} \in S$. Each hyperedge is assigned to a positive weight in the hyperedge set. All weights are stored in the diagonal matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$, and the initial weight of matrix is one, which means that each hyperedge is treated equally. It has been proved in [30] that a hypergraph degenerates into an ordinary graph if and only if each hyperedge is associated with two vertices.

3.4.2. Incidence Matrix. The incidence matrix in hypergraph theory reveals the relationship between hypergraph vertices and hyperedges. For an undirected hypergraph incidence matrix with no isolated points $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$, it is defined as:

$$H_{\text{ij}} = \begin{cases} 1, & v_i \in e_j, \\ 0, & v_i \notin e_j. \end{cases} \tag{6}$$

For a vertex in a hypergraph, its degree is defined as the sum of all hyperedge weights associated with the vertex:

$$D_{ii} = \sum_{i=1}^{|\mathcal{E}|} W_{\text{ee}} H_{\text{ie}}. \tag{7}$$

Similarly, the degree of a hyperedge is defined as:

$$B_{\text{ee}} = \sum_{i=1}^{|\mathcal{V}|} H_{\text{ie}}, \tag{8}$$

where $\mathbf{D} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $\mathbf{B} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ are both diagonal matrices.

3.4.3. Dynamic Hypergraph Construction. Dynamic hypergraphs are devised to unveil additional information concealed within static graphs. Notably, there are abundant signals in the temporal price movements of related stocks. By drawing inspiration from [18], we employ a residual architecture to extract the information flow of each stock after filtering out the static relationships.

As shown in Figure 3 (left frame), we initialize the hypergraph structure with the input feature embedding $\mathbf{X_{t,h}} - \mathbf{X_{t,s}^b}$. Dynamic hypergraphs are created through the following steps. We start with N hyperedges, where each hyperedge "e" is initialized with the feature embeddings of its corresponding nodes "v." Subsequently, each node identifies the hyperedge closest to itself, excluding the hyperedge initialized with its own feature embedding. Following that, each node is incorporated into the closest hyperedge, potentially leading to the removal of hyperedges which did not contain any nodes. For those hyperedges that persist after this process, the nodes used in their initialization are also added to them. Lastly, it is worth noting that among these N nodes, some nodes join a single hyperedge while others join two hyperedges (the ones whose initialization hyperedges have not been deleted). For nodes joining two hyperedges, we employ the k-nearest neighbor algorithm to find the k closest nodes to form a new hyperedge. The hyperedge set is dynamically adjusted when the feature embeddings evolve with the network going deeper. In our proposed method, the incidence matrix $\mathrm{H}_{\text{dynamic}}$ is obtained through the dynamic hypergraph construction algorithm. The entire process of dynamic hypergraph construction is shown in Algorithm 1.

3.4.4. Hypergraph Convolution. Hypergraph convolution is rooted in spatial domain graph theory, which conceptualizes hypergraph learning as a process of information exchange among interconnected nodes with neighbor relationships [10]. The input of the l^{th} hypergraph convolution layer is the feature embedding $\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l}}$, $\mathbf{X}_{\mathbf{t,d}}^{\mathbf{0}} = \mathbf{X}_{\mathbf{t,h}} - \mathbf{X}_{\mathbf{t,s}}^{\mathbf{b}}$. As shown in Figure 3 (right frame), the hypergraph spatial domain convolution updates the feature $\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l}}$ to a new feature $\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l+1}}$. This closed-loop message-passing cycle mechanism involves two-stage directed message flow propagation, which can be described in matrix form as follows.

The first stage: $\mathbf{Y}_{\mathbf{t,d}}^{\mathbf{l}} = \mathbf{W}\mathbf{B}^{-1}\mathbf{H}^{\top}\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l}}$ represents the aggregation of each node feature $\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l}} \in \mathbb{R}^{N\times F_h}$ into hyperedge feature $\mathbf{Y}_{\mathbf{t,d}}^{\mathbf{l}} \in \mathbb{R}^{N\times F_h}$.

The second stage: $\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l+1}} = \sigma(\mathbf{D}^{-1}\mathbf{H}\mathbf{Y}_{\mathbf{t,d}}^{\mathbf{l}}\mathbf{\Theta})$ indicates that each vertex updates its features by aggregating hyperedge features. Also, $\sigma(\cdot)$ indicates the activation function. Therefore, the entire update rule of hypergraph convolution is obtained as follows:

$$\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l+1}} = \mathrm{LeakyReLU}\left(\mathbf{D}^{-1}\mathbf{H}\mathbf{W}\mathbf{B}^{-1}\mathbf{H}^{\mathbf{T}}\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l}}\mathbf{\Theta}\right), \tag{9}$$

where $\mathbf{\Theta}$ is a learnable parameter matrix; LeakyReLU is a rectified linear unit activation function; and $\mathbf{D}$, $\mathbf{B}$, $\mathbf{H}$, and $\mathbf{W}$ have been defined in Sections 3.4.1 and 3.4.2. We set $\mathbf{W} = \mathbf{I}$, $\mathbf{H} = \mathbf{H}_{\text{dynamic}}$.

Similar to the static graph module, the dynamic hypergraph module also has two output branches, i.e., backcast $\mathbf{X}_{\mathbf{t,d}}^{\mathbf{b}}$ and forecast $\mathbf{Y}_{\mathbf{t,d}}^{\mathbf{f}}$:

$$\begin{aligned}\mathbf{X}_{\mathbf{t,d}}^{\mathbf{b}} &= \sigma\left(\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l+1}}\mathbf{W}_{\mathbf{d}}^{\mathbf{b}}\right),\\ \mathbf{Y}_{\mathbf{t,d}}^{\mathbf{f}} &= \sigma\left(\mathbf{X}_{\mathbf{t,d}}^{\mathbf{l+1}}\mathbf{W}_{\mathbf{d}}^{\mathbf{f}}\right),\end{aligned} \tag{10}$$

where $\mathbf{W}_{\mathbf{d}}^{\mathbf{b}}$ and $\mathbf{W}_{\mathbf{d}}^{\mathbf{f}}$ are learnable parameter matrices.

To mitigate the impact of both static and dynamic relationships, we introduce an independent module for capturing individual stock time series information. $\mathbf{W}_{\mathbf{i}}^{\mathbf{f}}$ is a learnable parameter matrix.

$$\begin{aligned}\mathbf{X}_{\mathbf{t,i}} &= \mathbf{X}_{\mathbf{t,h}} - \mathbf{X}_{\mathbf{t,s}}^{\mathbf{b}} - \mathbf{X}_{\mathbf{t,d}}^{\mathbf{b}},\\ \mathbf{Y}_{\mathbf{t,i}}^{\mathbf{f}} &= \sigma\left(\mathbf{X}_{\mathbf{t,i}}\mathbf{W}_{\mathbf{i}}^{\mathbf{f}}\right).\end{aligned} \tag{11}$$

3.5. Learning to Rank and Network Optimization. We sum up three different stock representations $\mathbf{Y}_{\mathbf{t,s}}^{\mathbf{f}}$, $\mathbf{Y}_{\mathbf{t,d}}^{\mathbf{f}}$, and $\mathbf{Y}_{\mathbf{t,i}}^{\mathbf{f}}$ as $\mathbf{Y}_{\mathbf{t}} = \mathbf{Y}_{\mathbf{t,s}}^{\mathbf{f}} + \mathbf{Y}_{\mathbf{t,d}}^{\mathbf{f}} + \mathbf{Y}_{\mathbf{t,i}}^{\mathbf{f}}$ and feed $\mathbf{Y}_{\mathbf{t}}$ into a fully connected (FC) layer to predict ranking $\hat{\mathbf{y}}^{\mathbf{t+1}}$:

$$\begin{aligned}\hat{\mathbf{y}}^{\mathbf{t+1}} &= \mathrm{FC}(\mathbf{Y}_{\mathbf{t}})\\ &= \mathrm{FC}\left(\left[\mathbf{Y}_{\mathbf{t,s}}^{\mathbf{f}} + \mathbf{Y}_{\mathbf{t,d}}^{\mathbf{f}} + \mathbf{Y}_{\mathbf{t,i}}^{\mathbf{f}}\right]\right).\end{aligned} \tag{12}$$

A combination of pointwise regression loss and pairwise rank-aware loss is used to optimize SD-RL:

$$\mathscr{L} = \left\|\hat{\mathbf{y}}^{\mathbf{t+1}} - \mathbf{y}^{\mathbf{t+1}}\right\|^2 + \beta\sum_{i=0}^{N}\sum_{j=0}^{N}\max\left(0, -\left(\hat{y}_i^{t+1} - \hat{y}_j^{t+1}\right)\left(y_i^{t+1} - y_j^{t+1}\right)\right), \tag{13}$$

where $\mathbf{y}^{\mathbf{t+1}}$ is the true value and β is a hyperparameter to balance the two loss terms. The former of the loss function minimizes the difference between the predicted ranking and the true ranking. The latter encourages the predicted ranking of a stock pair to have the same relative order as the true ranking.

4. Experiments

In this section, the details of the dataset, training settings, and experimental evaluation metrics are provided. Subsequently, a series of experiments are conducted and the results verify the effectiveness of the proposed SD-RL method.

4.1. Dataset and Training Settings

4.1.1. CSI 300 and CSI 100 [18]. The CSI 300 consists of the most representative 300 stocks in Shanghai and Shenzhen A-shares. The CSI 100 is made up of the 100 largest stocks in the CSI 300 stock set. We utilized the most recent 60-day raw data of these stocks, including opening prices, highest prices, lowest prices, closing prices, trading volumes, and volume-weighted average prices. The stock data from the CSI 300 and CSI 100 were collected from 01/01/2007 to 12/31/2020. We then divided this dataset into training, validation, and test sets in chronological order. Our model was trained, and its parameters were fine-tuned until it achieved the best performance on the validation set. We conducted each experiment five times and computed the average model performance on the test set to ensure robust results.

4.1.2. NASDAQ and NYSE [4]. Considering the inherent volatility of financial markets, we conducted additional experiments to assess the robustness of our model across different market conditions. For this purpose, we utilized two publicly available stock datasets created in [4], where the authors compiled price records for 1026 NASDAQ and 1737 NYSE stocks spanning from 01/02/2013 to 12/08/2017. For both datasets, only the information of stock industry-belonging relationships is provided.

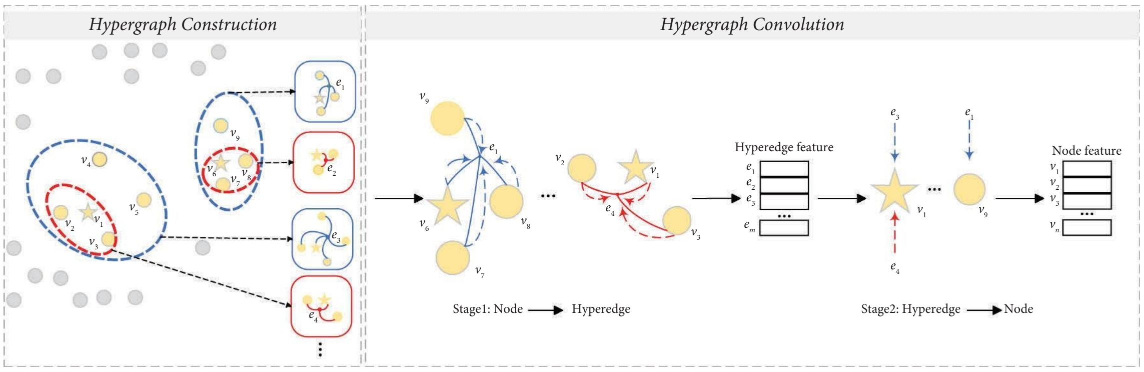

FIGURE 3: The illustration of hypergraph construction and hypergraph convolution. In the process of hypergraph construction, we follow the following steps. Initially, each node search for its nearest neighbor, which is marked as a star node in the figure. For the sake of clarity in our illustration, we depict two-star nodes in the figure, representing the closest points among the surrounding nodes. These star nodes are enclosed within blue dotted lines, effectively forming hyperedges. Next, we treat the star node as the central node and create a hyperedge by applying the k-nearest neighbor algorithm. The connection is denoted by a red dotted line (in the figure, it is indicated as 2-nn).

```
     Input: Input embedding $\mathbf{X}$; Nearest neighbor parameter $k$
     Output: Hyperedge set $\mathbf{E}$; The set of vertices possessed by a hyperedge $e$ $\mathbf{pos(e)}$
     Function: K-nearest neighborhood selection $knn$; Distance function $dis$; Smallest distance index selection $topK$
(1)  for $u$ in $range$ ($len$ ($\mathbf{X}$)) do
(2)      $\mathbf{D} = dis(\mathbf{X}(u), \mathbf{X})$
(3)      $\mathbf{D} = sort(\mathbf{D})$
(4)      $ind = topK(\mathbf{D}, 2) - topK(\mathbf{D}, 1)$
(5)      $\mathbf{E}.insert(ind)$
(6)      $\mathbf{pos(ind)}.insert(u)$
(7)  end for
(8)  for $e$ in $\mathbf{E}$ do
(9)      $v_e = knn(\mathbf{X}[e], \mathbf{X}, k)$
(10)     $\mathbf{pos(e)}.insert(v_e)$
(11) end for
```

ALGORITHM 1: Dynamic hypergraph construction.

4.1.3. Training Settings. For our proposed framework, we employed the Adam optimizer to fine-tune parameters and set the initial learning rate at 0.0002. Here are the specific parameter ranges used in our experiments: the hidden state size of GRU within (32, 64, 128), the embedding size of the static graph module within (32, 64, 128), the ranges of saturation rate α from 0.5 to 3, the ranges of parameter λ from 0 to 0.6, the ranges of parameter k in the k nearest neighbor algorithm from 2 to 16, and the ranges of β from 0.1 to 2. The models were implemented using the PyTorch framework, and we conducted a grid search to determine the optimal hyperparameters.

4.2. Evaluation. We assessed the effectiveness of our approach in terms of the ranking performance. Building upon prior research [18], we evaluated the prediction results using two commonly used metrics: Information Coefficient (IC) and Rank IC, which are defined as follows:

$$\begin{aligned}\mathrm{IC}\left(\mathbf{y}^{\mathbf{t}}, \hat{\mathbf{y}}^{\mathbf{t}}\right) &= \mathrm{corr}\left(\mathbf{y}^{\mathbf{t}}, \hat{\mathbf{y}}^{\mathbf{t}}\right),\\ \mathrm{RankIC}\left(\mathbf{y}^{\mathbf{t}}, \hat{\mathbf{y}}^{\mathbf{t}}\right) &= \mathrm{corr}\left(\mathrm{rank}_{\mathbf{y}^{\mathbf{t}}}, \mathrm{rank}_{\hat{\mathbf{y}}^{\mathbf{t}}}\right),\end{aligned} \tag{14}$$

where $\mathrm{corr}(\cdot)$ is the Pearson correlation coefficient and $\mathrm{rank}_{\mathbf{y}^{\mathbf{t}}}$ and $\mathrm{rank}_{\hat{\mathbf{y}}^{\mathbf{t}}}$ are the labels and predicted rankings from high to low, respectively. In addition, we also use another metric Precision@N to evaluate the accuracy of the top N predictions of the model. Assuming N equals 10, 5 labels out of the top 10 predictions are positive. Precision@10 equals 50%. In order to compare with existing research, we also set N to 3, 5, 10, and 30 to evaluate the models.

4.3. Experimental Results and Analysis. To validate the superior ranking performance of SD-RL, we conducted a comparative analysis against several existing baseline models. These baseline models include the following.

4.3.1. SFM [15]. It is a variant of the LSTM network that decomposes the hidden state of the LSTM storage unit into multiple frequency components, effectively memorizing time series information of different frequencies.

4.3.2. ALSTM [14]. It is a variant of the LSTM network with better generalization ability, which uses increased adversarial training to simulate the randomness in the model training process.

4.3.3. ALSTM + TRA [16]. It is an extension of ALSTM that leverages a temporal routing adapter (TRA) to learn multiple trading patterns in stock market data.

4.3.4. Transformer [3]. It is a stock trend prediction model based on the transformer architecture, which integrates a multiscale Gaussian prior with a self-attention mechanism to model temporal context information.

4.3.5. GATs [8]. It is a variant of graph convolutional networks that aggregates time series feature embeddings extracted by a GRU network using an attention mechanism.

4.3.6. HIST [18]. It is a graph-based neural network that extracts concept-oriented shared information to forecast stock trends.

Table 1 presents a summary of the ranking performance achieved by various methods across different values of N. Notably, ALSTM + TRA outperforme other models in terms of both IC and Rank IC metrics among those models that do not integrate relational information. Hence, when assessing these metrics with nonrelational models, we solely compare our model's results with the ALSTM + TRA model. Furthermore, within the domain of graph-based models, HIST showcases superior performance compared to GATs and other baseline models that do not integrate relational information. It is worth emphasizing that, while HIST relies on a predefined graph structure, our approach SD-RL excells by proficiently capturing both static and dynamic relationships through a data-driven methodology. This results in significantly improved performance across multiple metrics. To be more precise, on the CSI 100 datasets, SD-RL outperforms the second-place model by an average margin of nearly 3.3% and 2.50% in terms of IC and Rank IC, respectively.

Furthermore, we continue our experiments on the NASDAQ and NYSE stock datasets. Table 2 shows the ranking performance of different methods. It is noteworthy that the overall performance of all models in the A-share market surpass that in the US stock market. This difference in performance could be attributed to the relatively shorter data period covered by the US stock dataset and the larger number of stocks it comprised. Our model achieves promising prediction results, consistently standing at the top two positions in most evaluation metrics. It can be found that the performance of graph-based models have significantly improved by leveraging the relationship information. In addition, we also find that HIST achieves significantly better results on the NYSE stock datasets than NASDAQ, which could be attributed to the fact that HIST defines the static relationships at the beginning (leveraging the prior information of industry relationships). Industry relationships reflect more of long-term correlations between stocks, and NASDAQ is more susceptible to short-term factors. However, the variation of our model on these two datasets is relatively small. It could be attributed to the fact that our model learns the static-dynamic (long and short term) relationships between stocks over time in a data-driven manner.

4.4. Model Component Ablation Study. To investigate the contributions of different components in SD-RL, three variants of SD-RL were designed, namely, "GRU + Attn," "GRU + Attn + Sta," and "GRU + Attn + Sta + Dy."

(i) GRU + Attn: Only the temporal feature extraction module in SD-RL is retained. It is used to verify the impact of adding a temporal attention layer to the GRU neural network.
(ii) GRU + Attn + Sta: Only the temporal feature extraction module and static graph module in SD-RL are retained. It is used to verify the impact of the static graph module in improving performance.
(iii) GRU + Attn + Sta + Dy: The temporal feature extraction module, static graph module, and dynamic hypergraph module in SD-RL are retained. It is used to verify the impact of integrating the two information flows from the static graph and dynamic hypergraph.
(iv) GRU + Attn + Sta + Dy + Ind: the complete model (SD-RL) is used to verify the impact of combining the three types of information flows from static graph, dynamic hypergraph, and individual stock time series information.

Table 3 compares the performance between SD-RL and the variant methods. It is evident that SD-RL without any relational information (GRU + Attn) yields the least favorable results, underscoring the value of considering interrelations among stocks. Additionally, we observe that the model which integrates both static and dynamic relationship information (GRU + Attn + Sta + Dy) outperforms the one that solely focuses on static relationships (GRU + Attn + Sta). This affirms the significance of both static and dynamic relationships, highlighting that neither of them should be disregarded. Notably, SD-RL (GRU + Attn + Sta + Dy + Ind) attains the most favorable outcomes, providing further validation that amalgamating the output information from distinct modules enhances the capacity of feature embedding to capture trends.

4.5. Visualization of Embeddings. To further validate the efficacy of our proposed model, we employt-distributed stochastic neighborhood embedding (t-SNE) [31] to

Table 1: Ranking performance of different methods on the China's A-share dataset when considering different numbers of *N*.

Model	CSI 100						CSI 300					
	IC (↑)	Rank IC (↑)	Precision@N (↑)				IC (↑)	Rank IC (↑)	Precision@N (↑)			
			Top3	Top5	Top10	Top30			Top3	Top5	Top10	Top30
MLP	0.071	0.067	56.53	56.17	55.49	53.55	0.082	0.079	57.21	57.10	56.75	55.56
SFM	0.081	0.074	57.79	56.96	55.92	53.88	0.102	0.096	59.84	58.28	57.89	56.82
GRU	0.103	0.097	59.97	58.99	58.37	55.09	0.113	0.108	59.95	59.28	58.59	57.43
LSTM	0.097	0.091	60.12	59.49	59.04	54.77	0.104	0.098	59.51	59.27	58.40	56.98
ALSTM	0.102	0.097	60.79	59.76	58.13	55.00	0.115	0.109	59.51	59.33	58.92	57.47
Transformer	0.089	0.090	59.62	59.20	57.94	54.80	0.106	0.104	60.76	60.06	59.48	57.71
ALSTM + TRA	0.107	0.102	60.27	59.09	57.66	55.16	0.119	0.112	60.45	59.52	59.16	58.24
GATs	0.096	0.090	59.17	58.71	57.48	54.59	0.111	0.105	60.49	59.96	59.02	57.41
HIST	0.120	0.115	61.87	60.82	59.38	56.04	0.131	**0.126**	61.60	61.08	60.51	58.79
SD-RL	**0.124**	**0.118**	**62.86**	**62.07**	**60.05**	**56.28**	**0.131**	0.125	**62.39**	**61.70**	**60.84**	**58.83**

The best result in terms of each metric is indicated in bold.

Table 2: Ranking performance of different methods on the NASDAQ and NYSE datasets when considering different numbers of *N*.

Model	NASDAQ						NYSE					
	IC (↑)	Rank IC (↑)	Precision@N (↑)				IC (↑)	Rank IC (↑)	Precision@N (↑)			
			Top3	Top5	Top10	Top30			Top3	Top5	Top10	Top30
MLP	0.023	0.015	49.45	48.43	46.32	44.16	0.021	0.013	49.04	47.72	47.93	45.28
SFM	0.027	0.022	51.19	50.42	50.21	49.92	0.024	0.024	51.19	51.05	50.22	49.57
GRU	0.031	0.029	52.74	51.86	50.43	50.13	0.031	0.028	52.17	51.47	51.44	50.93
LSTM	0.030	0.027	51.65	51.22	50.73	50.18	0.030	0.027	51.75	51.52	50.89	50.48
ALSTM	0.036	0.034	53.79	52.03	51.98	51.39	0.036	0.035	53.03	52.33	50.71	50.30
Transformer	0.029	0.026	51.58	51.19	50.30	50.08	0.030	0.027	52.02	51.17	50.63	50.19
ALSTM + TRA	0.040	0.040	54.02	53.49	53.41	52.86	0.039	0.038	53.95	53.52	52.96	52.24
GATs	0.038	0.037	53.87	53.21	52.48	52.09	0.038	0.038	53.87	53.62	52.89	51.72
HIST	0.042	0.041	54.71	54.15	53.43	53.02	**0.047**	0.045	55.14	55.08	54.81	**53.97**
SD-RL	**0.044**	**0.044**	**55.28**	**55.34**	**54.85**	**54.43**	**0.047**	**0.046**	**55.68**	**55.41**	**54.84**	53.83

The best result in terms of each metric is indicated in bold.

Table 3: The results of ablation study.

Model	CSI 100						CSI 300					
	IC (↑)	Rank IC (↑)	Precision@N (↑)				IC (↑)	Rank IC (↑)	Precision@N (↑)			
			Top3	Top5	Top10	Top30			Top3	Top5	Top10	Top30
GRU + Attn	0.106	0.100	60.02	59.31	58.36	55.25	0.114	0.108	60.51	59.72	58.67	57.52
GRU + Attn + Sta	0.115	0.108	60.46	60.30	58.75	55.49	0.116	0.111	60.88	59.79	58.90	57.62
GRU + Attn + Sta + Dy	0.120	0.113	60.82	60.99	59.01	55.78	0.122	0.117	61.67	60.89	60.16	58.13
SD-RL	**0.124**	**0.118**	**62.86**	**62.07**	**60.05**	**56.28**	**0.131**	**0.125**	**62.39**	**61.70**	**60.84**	**58.83**

The best result in terms of each metric is indicated in bold.

project two types of stock embeddings into a 2D space: one originating from the GRU and the other from our method. Figure 4 shows these two types of embeddings from 100 stocks in the CSI 100 dataset. Stocks belonging to the banking and brewing industries are highlighted with bright colors. Stock embeddings in the same industry should be as similar as possible. To achieve this, we circle the labeled stocks (depicted as dots) belonging to the same industry, with the circle's size serving as an indicator of the clustering degree of the stock embeddings. Smaller circles indicate a higher level of aggregation. This observation showes that compared with GRU embeddings, the stock embeddings within the same industry in SD-RL are significantly more clustered (circles are smaller), and the embeddings of stocks in different industries are more dispersed. The above results show that our method is more effective in capturing and preserving the correlation among stocks in the same industry, and the learned stock embeddings are more discriminative. In addition, we also notice that BYD (a new energy vehicle company) is an isolated point in GRU embedding, while there is a relatively close spatial projection distance between the embedding of BYD stock and the embedding of Huaneng Power (an energy company) stock in SD-RL. This also confirmes the ability of our proposed method to capture hidden relationships between different companies.

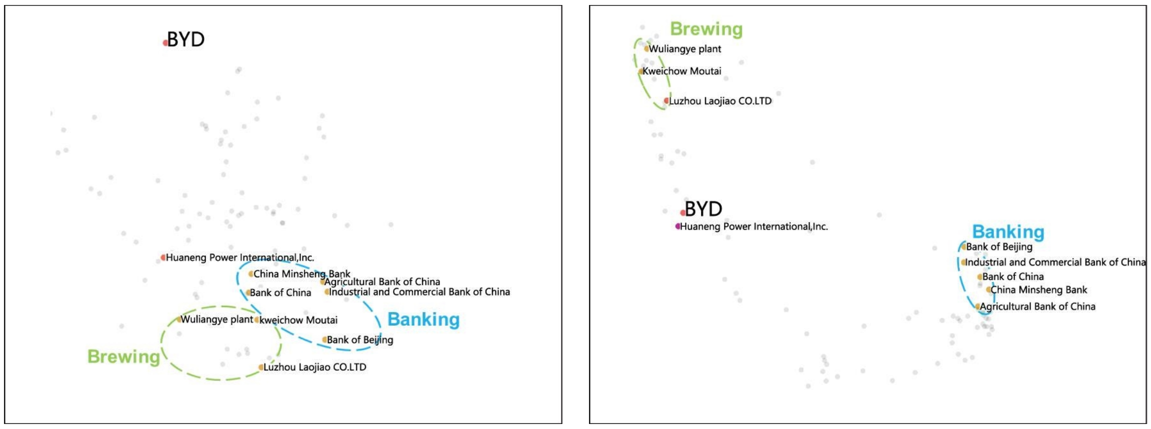

Figure 4: Comparison of GRU embeddings and SD-RL embeddings. (a) GRU embeddings. (b) SD-RL embeddings.

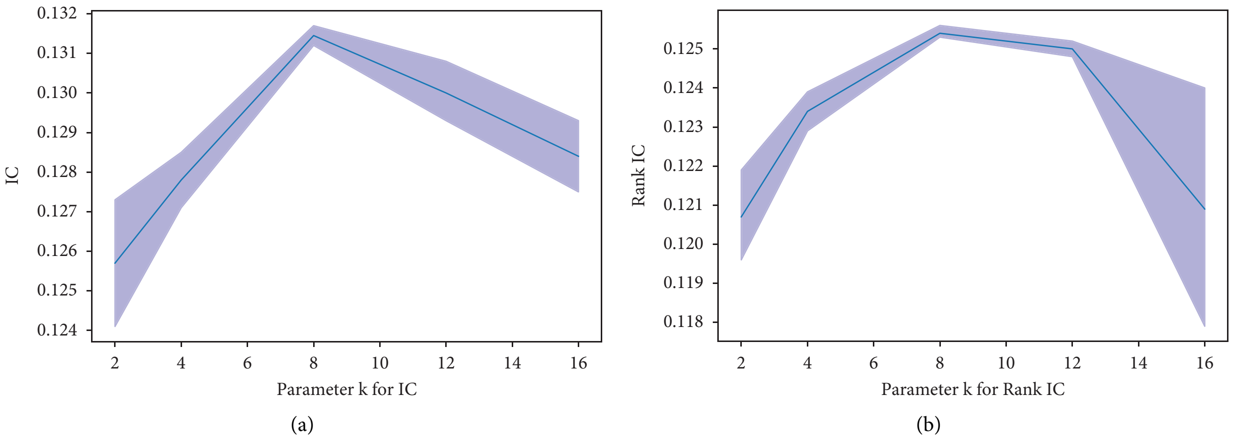

Figure 5: The effect of the parameter k of the knn algorithm on SD-RL.

4.6. Hyperparameter Analysis. To investigate the performance of SD-RL with different hyperparameter values, we maintain the default settings mentioned in Section 4.1 while varying a single hyperparameter at a time. Figures 5(a) and 5(b) illustrate the impact of "k" (the *knn* algorithm in the dynamic hypergraph construction module) on performance. The results reveal a pattern of improvement followed by degradation. The optimal results are achieved when "k" = 8. If "k" is excessively large or small, it diminishes the capacity to represent stock embedding trends, resulting in less discriminative learned stock embeddings.

5. Conclusions

It is a challenging but highly valuable task to recommend stocks by predicting the daily ranking of stock price changes. In this paper, we propose a static-dynamic hypergraph neural network framework based on residual learning to predict the ranking of stocks. The SD-RL framework offers significant advantages in modeling high-order data correlations and uncovering latent relationship information. The effectiveness of our proposed model is validated on two real-world market datasets. First, we compare the prediction performance of our model against various baseline models to assess its feasibility. Second, we construct and analyze three different variants of the SD-RL model to understand the influence of each component. Third, the feature embeddings of SD-RL and the GRU network are visually represented in a two-dimensional space. Additionally, we investigate the impact of various hyperparameter values on model performance. Experimental results indicate that our proposed approach is more practical and well suited for the real-world applications compared to the existing methods. It equips investors with the crucial information to make profitable investment decisions. Furthermore, our model can extend its applicability in the analysis of other graph data fields, such as traffic flow prediction.

In the future research, we plan to explore the integration of multisource information, including online financial news and social media data based on our model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the Young Research Foundation of Shanxi University of Finance and Economics (Grant no. QN-202014 and QN-202016), the National Natural Science Foundation of China (Grant no. 62003198), the Natural Science Research Project of Shanxi Basic Research Program (Grant no. 202203021221218), and the Graduate Student Research Innovation Program of Shanxi Province (Grant no. 2023KY523).

References

[1] C. Li, D. Song, and D. Tao, "Multi-task recurrent neural networks and higher-order Markov random fields for stock price movement prediction: multi-task RNN and higer-order MRFs for stock price classification," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1141–1151, New York, NY, USA, July 2019.

[2] T. Liu, X. Ma, S. Li, X. Li, and C. Zhang, "A stock price prediction method based on meta-learning and variational mode decomposition," *Knowledge-Based Systems*, vol. 252, Article ID 109324, 2022.

[3] Q. Ding, S. Wu, H. Sun, J. Guo, and J. Guo, "Hierarchical multi-scale Gaussian transformer for stock movement prediction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 4640–4646, Hong Kong, China, January 2020.

[4] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T. S. Chua, "Temporal relational ranking for stock prediction," *Association for Computing Machinery Transactions on Information Systems*, vol. 37, no. 2, pp. 1–30, 2019.

[5] R. Sawhney, S. Agarwal, A. Wadhwa, and R. R. Shah, "Spatiotemporal hypergraph convolution network for stock movement forecasting," in *Proceedings of the 2020 IEEE International Conference on Data Mining*, pp. 482–491, Sorrento, Italy, November 2020.

[6] S. Feng, C. Xu, Y. Zuo, G. Chen, F. Lin, and J. XiaHou, "Relation-aware dynamic attributed graph attention network for stocks recommendation," *Pattern Recognition*, vol. 121, Article ID 108119, 2022.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv.org/abs/1609.02907.

[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, https://arxiv.org/abs/1710.10903.

[9] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 3558–3565, 2019.

[10] Y. Gao, Y. Feng, S. Ji, and R. Ji, "Hgnn $\hat{}$+ $: general hypergraph neural networks," *Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, pp. 3181–3218, 2022.

[11] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-Beats: Neural basis expansion analysis for interpretable time series forecasting," 2019, https://arxiv.org/abs/1905.10437.

[12] D. M. Nelson, A. C. Pereira, and R. A. De Oliveira, "Stock market's price movement prediction with LSTM neural networks," in *Proceedings of the 2017 International joint conference on neural networks*, pp. 1419–1426, Anchorage, AK, USA, May 2017.

[13] G. Shen, Q. Tan, H. Zhang, P. Zeng, and J. Xu, "Deep learning with gated recurrent unit networks for financial sequence predictions," *Procedia Computer Science*, vol. 131, pp. 895–903, 2018.

[14] F. Feng, H. Chen, X. He, X. Ding, M. Sun, and T. S. Chua, "Enhancing stock movement prediction with adversarial training," 2018, https://arxiv.org/abs/1810.09936.

[15] L. Zhang, C. Aggarwal, and G. J. Qi, "Stock price prediction via discovering multi-frequency trading patterns," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 2141–2149, New York, NY, USA, August 2017.

[16] H. Lin, D. Zhou, W. Liu, and J. Bian, "Learning multiple stock trading patterns with temporal routing adaptor and optimal transport," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1017–1026, New York, NY, USA, August 2021.

[17] Y. Chen, Z. Wei, and X. Huang, "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, New York, NY, USA, October 2018.

[18] W. Xu, W. Liu, L. Wang et al., "HIST: a graph-based framework for stock trend forecasting via mining concept-oriented shared information," 2021, https://arxiv.org/abs/2110.13716.

[19] D. Cheng, F. Yang, S. Xiang, and J. Liu, "Financial time series forecasting with multi-modality graph neural network," *Pattern Recognition*, vol. 121, Article ID 108218, 2022.

[20] J. Jiang, Y. Wei, Y. Feng, J. Cao, and Y. Gao, "Dynamic hypergraph neural networks," in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 2635–2641, Xiamen, China, August 2019.

[21] N. Yin, F. Feng, Z. Luo et al., "Dynamic hypergraph convolutional network," in *Proceedings of the 2022 IEEE 38th International Conference on Data Engineering*, pp. 1621–1634, Kuala Lumpur, Malaysia, May 2022.

[22] J. Wei, Y. Wang, M. Guo, P. Lv, X. Yang, and M. Xu, "Dynamic hypergraph convolutional networks for skeleton-based action recognition," 2021, https://arxiv.org/abs/2112.10570.

[23] J. Wang, Y. Zhang, Y. Wei, Y. Hu, X. Piao, and B. Yin, "Metro passenger flow prediction via dynamic hypergraph convolution networks," *Institute of Electrical and Electronics Engineers Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7891–7903, 2021.

[24] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang, "Hypergraph contrastive collaborative filtering," in *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*, pp. 70–79, New York, NY, USA, July 2022.

[25] A. Amini, N. Firouzkouhi, A. Gholami, A. R. Gupta, C. Cheng, and B. Davvaz, "Soft hypergraph for modeling global interactions via social media networks," *Expert Systems with Applications*, vol. 203, Article ID 117466, 2022.

[26] C. Cui, X. Li, C. Zhang, W. Guan, and M. Wang, "Temporal-relational hypergraph tri-attention networks for stock trend prediction," *Pattern Recognition*, vol. 143, Article ID 109759, 2023.

[27] X. Li, C. Cui, D. Cao, J. Du, and C. Zhang, "Hypergraph-based reinforcement learning for stock portfolio selection," in *Proceedings of the ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4028–4032, Singapore, May 2022.

[28] J. M. Jeanblanc, M. Yor, and M. Chesney, *Mathematical Methods for Financial Markets*, Springer Science & Business Media, Heidelberg, Germany, 2009.

[29] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 753–763, New York, NY, USA, August 2020.

[30] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognition*, vol. 110, Article ID 107637, 2021.

[31] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, 2008.