Learning Dynamic Dependencies With Graph Evolution Recurrent Unit for Stock Predictions

Hu Tian, Xingwei Zhang, Xiaolong Zheng, *Member, IEEE*, and Daniel Dajun Zeng, *Fellow, IEEE*

***Abstract*—Investment decisions and risk management require understanding the time-varying dependencies between stocks. Graph-based learning systems have emerged as a promising approach for predicting stock prices by leveraging interfirm relationships. However, existing methods rely on a static stock graph predefined from finance domain knowledge and large-scale data engineering, which overlooks the dynamic dependencies between stocks. In this article, we present a novel framework called graph evolution recurrent unit (GERU), which uses a dynamic graph neural network to automatically learn the evolving dependencies from historical stock features, leading to better predictions. Our approach consists of three parts: first, we develop an adaptive dynamic graph learning (ADGL) module to learn latent dynamic dependencies from stock time series. Second, we propose a clustered ADGL (clu-ADGL) to handle large-scale time series by reducing time and memory complexity. Third, we combine the ADGL/clu-ADGL with a graph-gated recurrent unit to model the temporal evolutions of stock networks. Extensive experiments on real-world datasets show that our proposed methods outperform existing methods in predicting stock movements, capturing meaningful dynamic dependencies and temporal evolution patterns from the financial market, and achieving outstanding profitability in portfolio construction.**

***Index Terms*—Gated recurrent unit, graph representation learning, learning dynamic dependencies, stock prediction system.**

I. Introduction

STOCK predictions aim to predict the future trend and price of the listed company and play a crucial role in the financial investment domain. In the past few years, machine learning algorithms for financial market predictions have rapidly developed [1]. Due to the strong abilities to capture complex patterns and learn the nonlinear relationships from data, the deep learning methods have better performances than traditional machine learning systems on the stock movement predictions [2], [3]. Unfortunately, from the perspective of information usage, most of these solutions follow a single-stock prediction schema, i.e., they treat stocks typically as independent of each other and ignore the relations between them.

In reality, stock markets are increasingly referring to complex interconnected systems as the recognition of vast interactivity between the listed companies and investors [4]. Currently, there are several common characteristics concerning the complexity of the interactions between the stocks. First, stock interactions are challenging to recognize precisely due to the existence of hidden interdependencies [6] (e.g., insider trading, price manipulation, and hidden shareholders beneath the surface of financial regulations). Second, the interfirm dependencies are dynamic evolutionary and shaped by multiple elements, such as business strategy adjustments [7] or the changes in the macro-economic environment [8]. Third, experimental financial researches indicate that the financial network is sparse, and only a few numbers of stocks own large degrees and influence powers [4]. Knowing the properties of financial complex systems is helpful to develop effective network-based prediction models or micro-trading tactics.

Graph (network) as an essential data structure describes the relationships between different entities. Stock predictions can be viewed naturally from a graph perspective. Graph neural networks (GNNs) have shown excellent performance in processing various real-world graphs due to their permutation-invariance, local connectivity, and compositionality [9], [10]. Recent research has proposed predicting the stock price movement by using interfirm relationships based on human knowledge and intuition [11], [12], [13], [14] and obtained some promising performances. These predictive systems follow a similar roadmap where one predefines a financial network (graph) by utilizing available interfirm relationships such as business partnerships to convert stock prediction into a node classification task, where the prediction for one stock can learn from other stocks in a graph. Subsequently, GNNs are trained on the predefined financial network to predict the future stock movements.

Although the existing network-based methods have shown better performance than other deep learning models, we argue that such solutions have three main limitations. First, constructing a predefined financial network relies on solid finance domain knowledge and large-scale data engineering, which is time-consuming, laborious, and costly. Second, the predefined network is incapable of addressing the complicated stock dependencies because the human intuition and knowledge with

Manuscript received 22 February 2023; accepted 30 May 2023. Date of publication 10 July 2023; date of current version 17 October 2023. This work was supported in part by the Ministry of Science and Technology of China under Grant 2020AAA0108401, and in part by the Natural Science Foundation of China under Grant 72225011 and Grant 72293575. This article was recommended by Associate Editor W.-K. V. Chan. *(Corresponding author: Xiaolong Zheng.)*

The authors are with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100190, China (e-mail: tianhu2018@ia.ac.cn; zhangxingwei2019@ia.ac.cn; xiaolong.zheng@ia.ac.cn; dajun.zeng@ia.ac.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2023.3284840.

Digital Object Identifier 10.1109/TSMC.2023.3284840

various prior assumptions may neglect the hidden interactions critical for effective decision-making. Third, these graph-based methods only construct stable interfirm relationships and learn node representations over the static stock network, which lacks the dynamic characteristic of the financial market. As such, while the nature of the market varies over time [5], wrong interdependencies between stocks can incur much noise for the learning models, thus potentially restricting the static model's generalization. Hence, modeling the dynamic temporal dependencies between stocks is indispensable for understanding the market evolution and predicting the future stock status.

To overcome these limitations, we propose a novel neural network framework, graph evolution recurrent unit (GERU), which tries to directly learn the dynamic temporal dependencies between stocks from the financial time series for better stock predictions. Specifically, we feed the current features of each stock and historical stock embeddings to an adaptive dynamic graph learning (ADGL) module to explicitly capture the latent dependencies and learn the relation-wise sequential representations. ADGL also exploits a sparsification mechanism to filter unimportant links and obtain fine interpretability. Furthermore, we propose a clustered ADGL (clu-ADGL) module, which improves ADGL to handle large-scale time series by only considering the dependencies between a stock and several other stocks affiliating the same cluster. Next, we exploit a gated graph recurrent unit (GGRU) [25], [42] to string ADGL/clu-ADGL and learn the node representations concerning topology and node attribute evolutions in dynamic graphs. Finally, we feed the latest stock representations to a fully connected (FC) layer to predict the stock movements.

The main contributions of this work can be summarized as follows.

1) We investigated directly learning the latent dynamic dependencies between stocks from the financial time series and developed an ADGL module to generate sparse but meaningful time-varying stock graphs.
2) We proposed a clu-ADGL which can present comparable performance with the ADGL and has high efficiency in terms of memory and computation when used in large-scale time-series scenarios.
3) We provided a generic dynamic graph learning and predictive system GERU by incorporating ADGL/clu-ADGL into the gated graph recurrent networks, which can model the interactions of time series at each time step and capture both the topology and node attribute changes in dynamic graphs.
4) We justified that GERU has better predictive performance than the baseline methods and can provide interpretability by discovering meaningful underlying dependencies between stocks with the evolution of the financial market.

The remainder of this article is organized as follows. In Section II, a brief review of related studies is presented. In Section III, our proposed GERU is described in detail. Section IV presents the experimental results on five real-world stock market datasets. Section V concludes this article and outlines the future work.

II. Literature Review

This article is conceptually and methodologically related to previous studies in graph-based financial predictions, learning graph models, and dynamic GNNs. In this section, we will review related work in these areas and highlight our contributions.

A. Graph-Based Financial Prediction

The traditional solutions aim to recognize valuable signals from the constructed financial networks to enhance the performance of predictive systems [15], [16], [17]. For example, Creamer et al. [15] found that the average eigenvector centrality of the corporate news networks at different time points has an impact on return and volatility of the leading blue-chip index for the Eurozone. By mining daily role-based trading networks, Sun et al. [16] fed the investors' trading behavior characteristics to the neural network to predict the stock price. Cao et al. [17] found out that the stock price volatility pattern networks' topology characteristics are significant features for the machine learning predictive models. Such methods can utilize the dynamic factors of stock market behavior from a systematic perspective but cannot consider the dependencies between stocks and the traditional machine learning methods are incapable of handling the graph-structured data.

Recently, using GNN methods to predict the stock market received increased attention. Based on shareholding relationships, Chen et al. [11] created a financial network consisting of the listed companies and their top 10 stockholders and trained a standard GCN model on graph-structured data to make predictions. Considering the diverse and complicated corporation relationships, some researchers proposed using multiple types of relations to build a financial network. Feng et al. [12] regarded the sector-industry and knowledge-based company relations as the edges of building a static stock graph. They used a long short-term memory (LSTM) and GCN to learn node embeddings from temporal features. HAST [13] introduced a hierarchical graph attention system to selectively aggregate information on different types of corporate relations. Viewing the different relations as different financial networks and exploiting multigraph convolutional networks to encode multiple types of stock graphs, Ye et al.'s [14] method can learn more complex relations for stock predictions. HAD-GNN [19] built the time-varying stock graphs leveraging stock price correlations and proposed a hybrid-attention dynamic GNN method to predict the stock market.

The superiority of graph-based prediction methods indicates that considering time series' dependencies is a valuable signal for more accurate forecasting. However, these solutions rely on predefined and static networks and cannot model relationship evolution's dynamic patterns. Prebuilding the network strongly relies on domain knowledge and data engineering and may neglect the hidden interactions critical for effective decision-making. Therefore, we argue that directly learning dynamic dependencies from time series of stocks can be a practical approach to address these challenges.

B. Learning Graph Models

When handling the time series with unknown interactions, the primary step is identifying the underlying graph topology. Learning graphs from data is an essential theme for graph machine learning and graph signal processing. The traditional approaches to learning graphs mainly contain the graphical Lasso methods [20] and the Granger causality-based methods [21]. Cardoso and Palomar [20] imposed Laplacian structural constraints into the Gaussian Markov random field model, which can find conditional correlations between stocks. Independent of the downstream tasks, these methods cannot learn the meaningful network structures for the task-specific and are not accessible to incorporate into GNNs to take advantage of the adaptations of deep neural networks (DNNs).

Recently, the DNN-based methods have been proposed to learn dependencies of time series [22], [23], [24]. These methods follow a similar principle: learning a global node embedding matrix by incorporating a learning graph module into the task-specific neural networks. Such a matrix can measure the nodes' similarities via paired inner-product. SDGNN [25] extended such an idea by concatenating the time series features with the initial node embeddings to infer time-varying node relations. HAST [13] and FinGAT [26] used a graph attention network (GAT) [18] to derivate latent and dynamic dependencies of time series by transforming the learned attention matrixes. However, these methods cannot model the stock graph's evolutionary characteristics. In addition, the above learning graph methods have quadratic space complexity regarding the number of time series because the learned relation matrix needs to be explicitly stored in the graphic processing unit (GPU) memory. These methods cannot scale to a large-scale financial market with thousands of securities.

III. Problem Formulation

Considering the stochasticity and volatility of the financial market, predicting the price movement is more attainable than forecasting the exact price. Thus, we target predicting the stock price movement in this article, a classification task in machine learning. Meanwhile, we assume that the set of investigated stocks is fixed and in the same financial market, implying that the stocks usually have strong interactions [5]. Hence, we can model the stocks with discrete-time dynamic graphs.

Given a stock set S that contains n stocks, the features of any stock $s \in S = \{1, 2, \ldots, n\}$ at time t is represented as $\widetilde{\mathbf{x}}_t^s = [\widetilde{x}_{1,t}^s, \widetilde{x}_{2,t}^s, \ldots, \widetilde{x}_{d,t}^s] \in \mathbb{R}^d$, where d is the number of features. These features could include those related to stock price (e.g., open and close prices), trading activities (e.g., trading volume), as well as other technical indicators (e.g., moving average). For all stocks in S, their features at time t can be denoted as $\widetilde{\mathbf{X}}_t = [\widetilde{\mathbf{x}}_t^1, \widetilde{\mathbf{x}}_t^2, \ldots, \widetilde{\mathbf{x}}_t^n]^T \in \mathbb{R}^{n\times d}$. We define the next τ-step price movement as

$$y^s(t+\tau) = \begin{cases} 0, & p_{t+\tau}^s \le p_t^s \\ 1, & p_{t+\tau}^s > p_t^s \end{cases} \tag{1}$$

where p_t^s denotes the close price of stock s at time t. Our target is to find a function f_θ to predict the next τ-step price movement based on the past T steps (lag size) historical data

$$\mathbf{y}(t+\tau) = f_\theta\big(\widetilde{\mathbf{X}}_t, \widetilde{\mathbf{X}}_{t-1}, \ldots, \widetilde{\mathbf{X}}_{t-T+1}\big) \tag{2}$$

where $\mathbf{y}(t+\tau) = [1\text{hot}(y^1(t+\tau)), \ldots, 1\text{hot}(y^n(t+\tau))]^T \in \mathbb{R}^{n\times 2}$.1hot$(\cdot)$ encodes the binary value to a one-hot vector. In this article, we investigate the stock dependencies on the long-term time scale. Each time step represents one month.

IV. Proposed Method

In this section, we first present the overall architecture of GERU system, which is shown in Fig. 1. The proposed GERU cantinas two main components: the ADGL module and the GGRU. The vanilla ADGL explicitly learns stock dependencies from the raw stock data for each time point by introducing a multihead sparse self-attention network. To reduce the space and time complexity of vanilla ADGL, we devise a clu-ADGL. In the GGRU, the linear transformation layers are replaced with GNNs to encode the stock representations. Finally, we obtain a generic method called GERU by combing ADGL and GGRU to model the dynamic evolutions of the learned stock dependencies and stock time series.

A. Adaptive Dynamic Graph Learning

To capture the underlying dependencies between stock time series, the ADGL directly learns dynamic similarities of the stock feature representations deriving from the current stock feature embeddings and the historical stock embeddings (See Fig. 2). Because our method can adapt to different market situations and constantly capture the dependencies of stocks, we call it ADGL.

1) Multihead Self-Attention: Self-attention is an attention mechanism that directs the model's focus and makes it learn the correlation between any two elements in the same input sequence [27]. In broad terms, self-attention is in charge of managing and quantifying the interdependence within the input elements. Based on this idea, we regard stocks in a financial market as a node sequence S. Each node (stock) $s \in S$ has its own raw features $\widetilde{\mathbf{x}}_t^s$ at time t. Before inputting into the ADGL, $\widetilde{\mathbf{x}}_t^s$ is transformed into a latent space whose dimension equals the model dimension d_m by a linear transformation $\mathbf{x}_t = \mathbf{W}_X^T\widetilde{\mathbf{x}}_t^s$, where $\mathbf{W}_X \in \mathbb{R}^{d\times d_m}$.

The self-attention function can be described as mapping the node embeddings into a set of query vectors and key vectors. By using the linear transformation, self-attention encodes the concatenation of $\mathbf{H}_{t-1}$ and $\mathbf{X}_t$ as a set of keys $\mathbf{K}_t \in \mathbb{R}^{n\times d_m}$ and quires $\mathbf{Q}_t \in \mathbb{R}^{n\times d_m}$

$$\begin{aligned} \mathbf{K}_t &= \big[\mathbf{H}_{t-1}; \mathbf{X}_t\big]\mathbf{W}_K \\ \mathbf{Q}_t &= \big[\mathbf{H}_{t-1}; \mathbf{X}_t\big]\mathbf{W}_Q \end{aligned} \tag{3}$$

where $\mathbf{X}_t = \widetilde{\mathbf{X}}_t\mathbf{W}_X$, $\mathbf{H}_{t-1} = [\mathbf{h}_{t-1}^1, \mathbf{h}_{t-1}^2, \ldots, \mathbf{h}_{t-1}^n]^T \in \mathbb{R}^{n\times d_m}$ is the historical hidden states at time step $t-1$, d_m is the dimension of $\mathbf{h}_t^s$, $[\cdot;\cdot]$ denotes the concatenation operator, $\mathbf{W}_K$ and $\mathbf{W}_Q \in \mathbb{R}^{2d_m\times d_m}$ are learnable parameters. Intuitively, the stock graphs at consecutive time steps should not vary too much, i.e., the current stock graph evolves from the predecessor. It implies that the ADGL should know the graph information from the last time step while learning the stock graph at the current

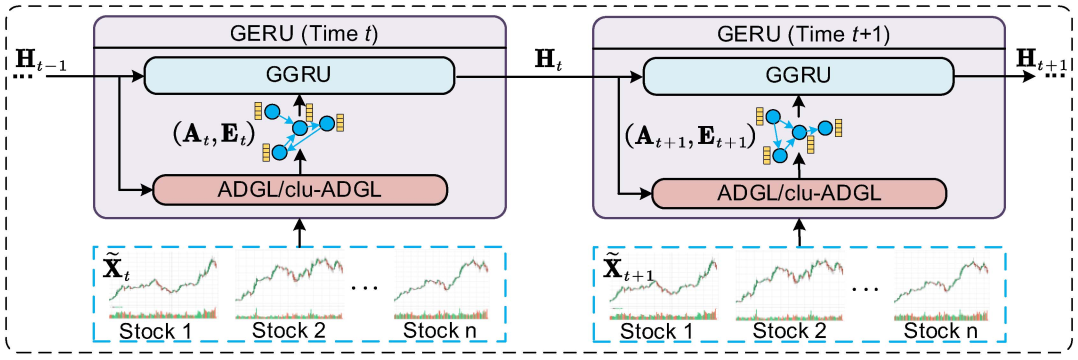


Fig. 1. Schematic illustration of the proposed predictive system. For simplicity, we only present the GERU at time t and $t+1$. The input of the GERU at time t is the feature of all stocks. The ADGL and clu-ADGL explicitly learn a stock graph $\mathbf{A}_t$ according to the current input data and its hidden state from the last time step. GGRU encodes the representations $\mathbf{E}_t$ of nodes concerning topology and node attribute changes in dynamic graphs. The detailed architectures of the ADGL/clu-ADGL and GGRU can be found in Figs. 2 and 5, respectively.

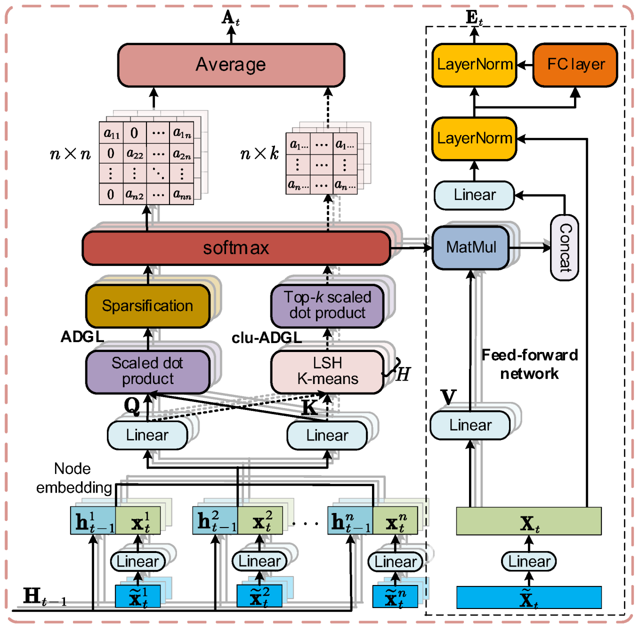


Fig. 2. ADGL is a modified multihead self-attention network consisting of H independent self-attention layers running in parallel and a sparsification function transforming the learned dense relation matrixes into sparse matrixes. The clu-ADGL exploits LSH based K-means clustering and replaces scaled dot product and sparsification with top-k scaled dot product (see Fig. 3). The right part represents the FFN.

time step. Hidden state $\mathbf{H}_{t-1}$ represents the dynamic node representations containing the locally topological information and node features of stock graphs before time t and concatenating $\mathbf{H}_{t-1}$ with $\mathbf{X}_t$ as the input allows the ADGL to generate more consistent stock graphs from prior knowledge.

The dynamic relation matrix $\mathbf{R}_t$ between all stocks at time t is computed by a compatibility function (i.e., the scaled dot-product attention [27] in (4) and Fig. 2) of the query with the corresponding key

$$\mathbf{R}_t = \frac{\mathbf{Q}_t\mathbf{K}_t^T}{\sqrt{d_m}}. \tag{4}$$

As suggested by [27], we also found it beneficial to perform self-attention layer H times with independent learned parameters $\mathbf{W}_K$ and $\mathbf{W}_Q$ (see Fig. 2). For the hth head, the dynamic relation matrix $\mathbf{R}_t^h$ can be formulated as

$$\mathbf{R}_t^h = \frac{\mathbf{Q}_t^h\mathbf{K}_t^{hT}}{\sqrt{d_m}} \tag{5}$$

where

$$\begin{aligned}\mathbf{K}_t^h &= [\mathbf{H}_{t-1}; \mathbf{X}_t]\mathbf{W}_K^h \\ \mathbf{Q}_t^h &= [\mathbf{H}_{t-1}; \mathbf{X}_t]\mathbf{W}_Q^h.\end{aligned} \tag{6}$$

A multihead attention mechanism means that the ADGL can jointly attend to information from different representation subspaces [27] at each time step. Intuitively, a multihead attention mechanism can learn richer stock dependencies, thus not neglecting those latent interactions critical for effective decision-making in the financial markets.

2) Sparsification: The dynamic relation matrix $\mathbf{R}_t$ can be normalized by the softmax function to generate a weighted adjacency matrix, which defines a dense digraph. However, the dense weights may lead to more diffused attention, i.e., predicting the focal stock could need to aggregate information from many neighbor stocks, even if some are not much correlated to the focal stock. A sparse financial network with a fewer number of edges also can significantly reduce the computation cost of the GNN component compared to a dense network. Therefore, we propose a sparsification strategy that keeps the k dependencies for the focal stock that are most conducive to making accurate predictions.

As shown in Fig. 3, the sparsification strategy's basic idea is straightforward. We explicitly select the k-largest element of each row in $\mathbf{R}_t^h$ and record their positions in the index matrix $\mathbf{IDX}_t^h$

$$\begin{aligned}&\mathbf{idx} = \text{argtopk}\left(\mathbf{R}_t^h[i,:]\right)\\ &\mathbf{IDX}_t^h[i,\mathbf{idx}] = 1\\ &\mathbf{IDX}_t^h[i,-\mathbf{idx}] = 0\end{aligned} \tag{7}$$

where k is a hyperparameter, argtopk$(\cdot)$ returns the indices of the top-k largest values of a vector, and $-\mathbf{idx}$ refers to the indices not in $\mathbf{idx}$. Then, we mask those unchosen elements in $\mathbf{R}_t^h$ with a sufficiently small negative value N (e.g., $-10e16$). The formulation definition can be illustrated as follows:

$$\overline{\mathbf{R}}_t^h = \mathbf{R}_t^h \otimes \mathbf{IDX}_t^h + N\left(\mathbf{1}\mathbf{1}^T - \mathbf{IDX}_t^h\right) \tag{8}$$

where the operator $\otimes$ is the Hadamard product, and $\mathbf{1}$ is an all-one vector. Since those unchosen stock relations are set as

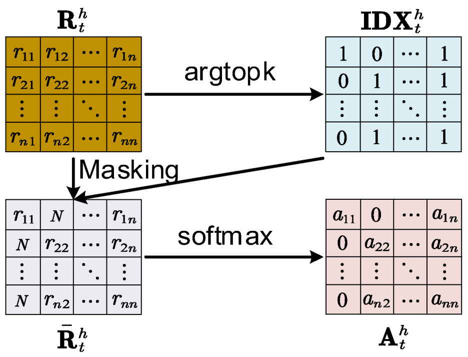
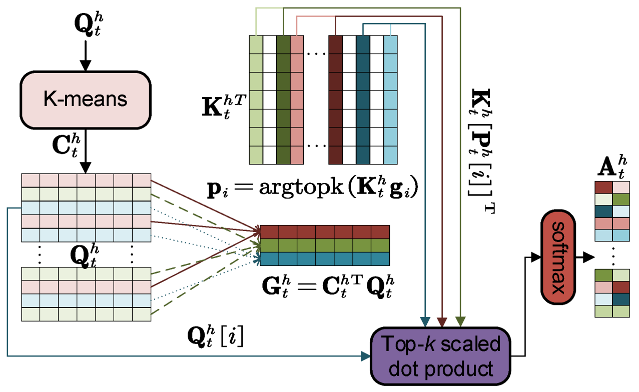


Fig. 3. Principle figure of sparsification. With the mask based on top-k selection and softmax function, only the most contributive neighbor stocks are assigned with weights.



$\mathbf{Q}_t^h$
K-means
$\mathbf{C}_t^h$
$\mathbf{K}_t^{hT}$
$\mathbf{p}_i = \text{argtopk}(\mathbf{K}_t^h \mathbf{g}_i)$
$\mathbf{K}_t^h[\mathbf{p}_t^h[i]]^T$
$\mathbf{G}_t^h = \mathbf{C}_t^{hT}\mathbf{Q}_t^h$
$\mathbf{Q}_t^h[i]$
Top-k scaled dot product
softmax
$\mathbf{A}_t^h$

Fig. 4. Computation process of dynamic sparse adjacency matrix $\mathbf{A}_t^h$, where $c = 3$ and $k = 2$. The left one-third figure shows the partition of $\mathbf{Q}_t^h$. The median one-third figure shows the calculation of cluster centroids and the selection of top-k keys of each cluster. The right one-third figure displays the learned sparse adjacency matrix $\mathbf{A}_t^h$.

sufficiently small negative value N, their weights are almost equal to 0 after softmax normalization. Finally, the masked relation matrix $\overline{\mathbf{R}}_t^h$ is normalized by applying a softmax function

$$\mathbf{A}_t^h = \text{softmax}\left(\overline{\mathbf{R}}_t^h\right) \tag{9}$$

where $\mathbf{A}_t^h$ refers to the adjacency matrix of a sparse stock graph.

In essence, the matrix $\mathbf{A}_t^h$ defines a weighted directed graph whose edge weight represents the proximity of source and destination in the latent space. Higher proximity indicates that source and destination stocks have a more similar pattern and stronger dependency. The element $\mathbf{A}_t^h[i,j]$ denotes the edge weight from source node j to target node i.

Finally, the unified adjacency matrix $\mathbf{A}_t$ can be obtained by averaging the sum of all $\mathbf{A}_t^h$

$$\mathbf{A}_t = \frac{1}{H}\sum_{h=1}^{H}\mathbf{A}_t^h. \tag{10}$$

The strategy of averaging all adjacency matrixes after the sparsification can retain the important dependencies in the corresponding subspaces and avoid using more complex multi-GNNs in the later recurrent unit, thus reducing the computation and memory cost. In practice, our experimental results indicate that for a specific stock, its top-k important neighbors have lots of overlaps across different heads. Thus, the summation cannot induce dense stock graphs in general. Because $\mathbf{R}_t^h$ has to be stored explicitly, the space complexity of ADGL is $\mathcal{O}(2nd_m + n^2)$. The time complexity of ADGL is $\mathcal{O}(2nd_m^2 + n^2 d_m)$.

3) Feed-Forward Network: It has been verified that the feed-forward network (FFN) of the transformer [27] can learn expressive representations by operating as key-value memories [28]. Following this line, we conduct the same feed-forward operations as the encoder module of the transformer (see the upper right of Fig. 2) to generate expressive stock embeddings

$$\begin{aligned}\mathbf{V}_t^h &= \mathbf{X}_t\mathbf{W}_V^h\\ \mathbf{O}_t &= \left[\mathbf{A}_t^h\mathbf{V}_t^1; \cdots ; \mathbf{A}_t^h\mathbf{V}_t^H\right]\mathbf{W}_o\\ \mathbf{O}_t' &= \text{LayerNorm}(\mathbf{X}_t + \mathbf{O}_t)\\ \mathbf{E}_t &= \text{LayerNorm}\big(\big(\text{ReLU}\big(\mathbf{O}_t'\mathbf{W}_1 + \mathbf{b}_1\big)\mathbf{W}_2\\ &\qquad + \mathbf{b}_2\big) + \mathbf{O}_t'\big)\end{aligned} \tag{11}$$

where $\mathbf{W}_V^h \in \mathbb{R}^{d_m\times d_m}, \mathbf{W}_o \in \mathbb{R}^{Hd_m\times d_m}, \mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d_m\times d_m}$, $\mathbf{b}_1$, $\mathbf{b}_2 \in \mathbb{R}^{d_m}$ are the model parameters, $ReLU(\cdot)$ is the activation function, and $LayerNorm(\cdot)$ is the layer normalization used for stabilizing the hidden state dynamics of neurons and reducing the training time [29]. As the feature embeddings of stocks, $\mathbf{E}_t \in \mathbb{R}^{n\times d_m}$ will be fed into the GGRU with the learned graph $\mathbf{A}_t$.

B. Clustered Adaptive Dynamic Graph Learning

Although the ADGL can be derived from a self-attention network, (5) leads to an expensive $\mathcal{O}(n^2)$ memory requirement, which makes the ADGL unable to handle large-scale scenarios, especially for a financial market with thousands of securities. To handle a large number of series, we make a tradeoff between the memory size and the precision of the learned graph. We propose a clu-ADGL, which only quantifies the interdependencies between a stock s's query and k stocks' keys that have top-k highest similarities with the centroid vector of s's cluster instead of all stocks' keys like that of vanilla ADGL. In fact, our method is similar to the Routing transformer [30], which K-means queries and keys simultaneously and only computes the similarity scores of queries and keys grouped into the same cluster.

The objective of clu-ADGL is to construct the sparse adjacency matrix $\mathbf{A}_t \in \mathbb{R}^{n\times k}$ directly. The overall process can be found in Fig. 4. First, we use K-means clustering to partition n stock queries $\mathbf{Q}_t^h$ into c clusters and obtain a cluster index matrix $\mathbf{C}_t^h \in [0,1]^{n\times c}$. To achieve an efficient Lloyd's algorithm [31] of K-means clustering, we transform $\mathbf{Q}_t^h$ into the Hamming space to speed the distance computation by exploiting random projection-based locality-sensitive hashing (LSH) [32]. Then, the centroids of all clusters can be calculated by

$$\mathbf{G}_t^h = \mathbf{C}_t^{hT}\mathbf{Q}_t^h \tag{12}$$

where $\mathbf{G}_t^h \in \mathbb{R}^{c\times d_m}$ is the hidden vectors of c centroids. For each centroid vector $\mathbf{g}_i = \mathbf{G}_t^h[i]^T \in \mathbb{R}^{d_m}$ $(i \in \{1,\ldots,c\})$, its top-k most similar keys can be selected by

$$\mathbf{p}_i = \text{argtopk}\left(\mathbf{K}_t^h\mathbf{g}_i\right) \tag{13}$$

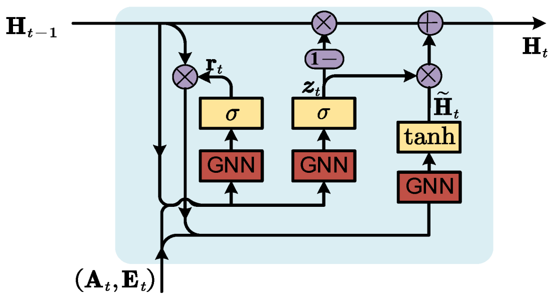


Fig. 5. Architecture of GGRU. $\mathbf{A}_t$ and $\mathbf{E}_t$ are from Fig. 2.

where $\mathbf{p}_i \in \{1, \ldots, n\}^k$ contains the indices of the top-k largest values of $\mathbf{K}_t^h \mathbf{g}_i$. Then, we can obtain all top-k indices $\mathbf{P}_t^h = [\mathbf{p}_{c_1}, \ldots, \mathbf{p}_{c_n}] \in \mathbb{R}^{n \times k}$, where $c_i = \text{argtop1}(\mathbf{C}_t^h[i])$. Finally, the dynamic sparse adjacency matrix $\mathbf{A}_t^h$ can be formulated as

$$\mathbf{A}_t^h[i,j] = \frac{\exp\left(\frac{\mathbf{Q}_t^h[i]\mathbf{K}_t^h[\mathbf{P}_t^h[i,j]]^T}{\sqrt{d_m}}\right)}{\sum_{l=1}^{k} \exp\left(\frac{\mathbf{Q}_t^h[i]\mathbf{K}_t^h[\mathbf{P}_t^h[i,l]]^T}{\sqrt{d_m}}\right)} \tag{14}$$

where $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, k\}$. We call the bracketed expression of numerator in (14) top-k scaled dot product. K-means is applied only in each iteration's forward phase and those clusters are fixed during the backward phase. Accordingly, $\mathbf{A}_t$ in (10) can be rewrote as an average of H sparse matrixes. $\mathbf{O}_t$ in (11) can be rewrote as

$$\mathbf{O}_t = \left[\widetilde{\mathbf{V}}_t^1; \cdots; \widetilde{\mathbf{V}}_t^h; \cdots; \widetilde{\mathbf{V}}_t^H\right]\mathbf{W}_o \tag{15}$$

where $\widetilde{\mathbf{V}}_t^h[i] = \sum_{j=1}^{k} \mathbf{A}_t^h[i,j]\mathbf{V}_t^h[\mathbf{P}_t^h[i,j]]$, $i \in \{1, \ldots, n\}$.

The time complexity of random projection-based LSH is $\mathcal{O}(nbd_m)$ [32], where b is the number of bits used for hashing. The time complexity of the original Lloyd's K-means is $\mathcal{O}(ncd_m r)$ [31], where r is the number of iterations. In this article, b and r are set as 32 and 10, respectively. Transformed into Hamming space, the complexity of computing distance changes from $\mathcal{O}(ncd_m)$ into $\mathcal{O}(nc)$. Thus, the time complexity of K-means clustering is $\mathcal{O}(ncr + cbr)$. The overall time complexity of LSH and K-means clustering is $\mathcal{O}(nbd_m + ncr + cbr)$. The time complexity of (6) and (12)–(14) is $\mathcal{O}(2nd_m^2 + ncd_m + nkd_m)$. Finally, the time complexity of $\mathcal{O}(nbd_m + ncr + cbr + ncd_m + nkd_m + 2nd_m^2)$. Because $\mathbf{A}_t^h$ is a sparse matrix, the space complexity of the clu-ADGL is $\mathcal{O}(2nd_m + nc + cd_m + nk)$. When n is large, the clu-ADGL changes the ADGL's time and space complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

C. Gated Graph Recurrent Unit

In addition to the interfirm relationships, stock prediction involves capturing evolutions about stock dependencies and features. Recurrent networks have shown superior performance in modeling temporal sequence data. Motivated by variational graph recurrent networks [33], we integrate GNNs into GRU as a GGRU to model the spatial and temporal patterns of the dynamic stock graphs. The architecture of GGRU can be found in Fig. 5.

TABLE I
DETAILED INFORMATION OF FIVE STOCK DATASETS

Dataset	CSI 100	CSI 300	S&P 100	Russ 1K	Russ 3K
#stocks	78	221	98	811	1886
Train. period	2015/01-2018/12		2015/01-2018/12		
Valid. period	2019/01-2019/06		2019/01-2019/06		
Test. period	2019/07-2020/12		2019/07-2020/12		

GGRU hidden layer is formulated mathematically as follows:

$$\begin{aligned}
\mathbf{r}_t &= \sigma\left(\text{GNN}_1\left(\mathbf{A}_t, \left[\mathbf{H}_{t-1}; \mathbf{E}_t\right]\right)\right) \\
\mathbf{z}_t &= \sigma\left(\text{GNN}_2\left(\mathbf{A}_t, \left[\mathbf{H}_{t-1}; \mathbf{E}_t\right]\right)\right) \\
\widetilde{\mathbf{H}}_t &= \tanh\left(\text{GNN}_3\left(\mathbf{A}_t, \left[\mathbf{r}_t \bigotimes \mathbf{H}_{t-1}; \mathbf{E}_t\right]\right)\right) \\
\mathbf{H}_t &= (\mathbf{1} - \mathbf{z}_t) \bigotimes \mathbf{H}_{t-1} + \mathbf{z}_t \bigotimes \widetilde{\mathbf{H}}_t
\end{aligned} \tag{16}$$

where $\text{GNN}_i(\cdot)$ $(i = 1, 2, 3)$ is independent one-layer GNN, $\mathbf{r}_t$ and $\mathbf{z}_t \in \mathbb{R}^{n \times d_m}$ are reset gate and update gate, respectively, and $\sigma(\cdot)$ is the sigmoid function. The graph structure $\mathbf{A}_t$ is learned from $\mathbf{H}_{t-1}$ and $\mathbf{X}_t$ via ADGL or clu-ADGL. The node representation $\mathbf{H}_t$ is generated from the past node representation $\mathbf{H}_{t-1}$, the current stock embedding $\mathbf{X}_t$, and the graph structure $\mathbf{A}_t$. In this way, GERU can jointly model the stock dependencies and hidden states of GGRU to capture both topological evolution and time-varying node attributes in dynamic stock graphs.

Finally, to predict the next τ-step price movements, we stack an FC layer on GGRU to calculate the movement probabilities

$$\hat{\mathbf{y}}(t + \tau) = \sigma\left(\mathbf{H}_t \mathbf{W}_y + \mathbf{b}_y\right) \tag{17}$$

where $\mathbf{W}_y \in \mathbb{R}^{d_m \times 2}$ and $\mathbf{b}_y \in \mathbb{R}^2$ are the trainable parameters. As a classification task, we choose the standard cross-entropy loss function to train the model. The parameters of ADGL/clu-ADGL and GGRU can be jointly updated via stochastic gradient descent.

V. EXPERIMENTAL RESULTS

In this section, we compare the performance of our proposed GERU with some state-of-the-art methods on stock movement prediction task. The practical value of our model in investment decision-making is also evaluated on real-world datasets via simulation trading.

A. Datasets

From the perspective of financial market evolution, stocks tend to have observable patterns on middle and long-term time scales [5]. Thus, we choose monthly prediction tasks to evaluate the effectiveness of GERU in the real world. We collected five market index-based datasets from the Chinese and American security markets. Using Tushare Pro API (https://tushare.pro/), we built the CSI 100 dataset and CSI 300 dataset in the Chinese market, consisting of the 100 and 300 largest and most liquid A-share stocks during 2020. From Yahoo Finance! (https://finance.yahoo.com/), we constructed the S&P 100, Russell 1 K, and Russell 3-K datasets which contain the 101, 1000, and 3000 largest and most established companies in the U.S. equity markets during 2020. Table I provides detailed information on these datasets.

We removed the stocks that were traded for fewer than 90% of all trading months during the corresponding data collection periods and retained 78, 221, 98, 811, and 1886 stocks on five datasets. We split each dataset into three parts: 1) 66.7% (48 months) for training (Train.); 2) 8.3% (6 months) for validation (Valid.); and 3) 25% (18 months) for testing (Test.). To comprehensively evaluate our proposed method's performance, we set three prediction tasks with $\tau = 1$, 3, and 6, respectively. The next τ-step price movement (label) of each stock was given by (1). All raw datasets are transformed via z-score normalization according to the training datasets. Our smallest dataset, CSI 100, has 5616 data points, and the largest dataset, Rus 3 K, has 135 792 data points. The dataset scale can display the discriminative powers of the evaluated methods.

B. Experimental Settings

We compared our proposed GERU with some representative and state-of-the-art baselines. These baselines can be divided into independent prediction, predefined graph-based, and learning graph-based methods. Considering that the baselines MTGNN [22] and AGCRN [23] both used GCN as their graph encoders, we also use GCN as the graph representation learning module in other graph-based methods except HATS [13] and FinGAT [26], which use the GAT.

1) Independent Prediction Methods: Independent prediction methods include LSTM [34], ALSTM [35], and Adv-LSTM [36], which only utilize each stock's own information for making predictions.

2) Predefined Graph-Based Methods: We predefined three types of interfirm relationships based on financial knowledge: 1) a completely connected stock graph (CSG); 2) a distance-based stock graph (DSG) is by computing the pairwise Euclidean distances between stock prices and selecting the threshold to make every node has at least one edge; and 3) a sector stock graph (SSG) [26] according to the global industry classification standard (GICS). A two-layer GCN model with ReLU activation function was used on CSG and DSG. We call these two baselines CSG-GCN and DSG-GCN. For the SSG, we chose two state-of-the-art methods FinGAT [26] (also called SSG-GAT) and HATS [13] which implicitly infer dynamic graphs. Although the FinGAT and HATS are based on the predefined static stock graphs, they both exploit the GAT that can derive time-varying graph attention matrixes, which can be seen as dynamic and hidden relations of stocks. Therefore, they are also two learning dynamic graph-based models. HAD-GNN [19] is a dynamic GNN method for time-varying stock graphs constructed based on historical stock price co-movement.

3) Learning Graph-Based Methods: MTGNN explicitly learns the static uni-directed relations among multivariate time series through a graph learning module and utilizes the graph convolution and temporal convolution modules to learn useful node representation for predictions [22]. AGCRN uses a similar approach to infer the static interdependencies between multiple variables implicitly and adopts a recurrent graph convolutional network to capture the temporal correlations in multivariate time series [23]. Furthermore, SDGNN [25] extends MTGNN and AGCRN as a learning dynamic graph-based model by concatenating the stock features with the initial stock embeddings to infer time-varying stock relations.

4) Hyperparameters: Following the setting of [37], we deployed the Adam optimizer with beta coefficients 0.9 and 0.999 and set the hidden dimension d_m as 64 for all models. Meanwhile, we set their corresponding hyperparameters, including the learning rate, regularization weight of L_2, lag size T, number of head H, and sparsification indicator k were selected via grid search within the sets of [0.0005, 0.001, 0.005, 0.01], [0, 5e-4, 5e-3], [1, 3, 6, 9, 12], [1, 3, 5, 7, 9], and [1, 3, 5, 10]. For clu-GERU, we search c within [3, 5, 10, 15]. For independent prediction methods, the batch size is set to 128. For graph-based methods, the batch size is set to 18 for S&P 100, CSI 100, and CSI 300, 3 for Rus 1 K, and 1 for Rus 3 K to avoid exceeding GPU memory. Especially for clu-GERU, the batch size is set to 18 for Rus 1 K, and 9 for Rus 3 K to speed up the training. We use PyTorch Geometric [38] to implement the mini-batch processing of GNNs. All methods are implemented with PyTorch and run on NVIDIA Tesla V100 GPUs. We exploit an early stopping strategy with the patience of 15. The code is available at https://github.com/Hugo-CAS/GERU/.

5) Evaluation Metrics: To measure the predictive performance, we calculate two evaluation metrics, including accuracy and precision, by the following formations:

$$\text{accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$
$$\text{precision} = \frac{tp}{tp + fp} \tag{18}$$

where tp, tn, fp, and fn are the number of true positive, true negative, false positive, and false negative cases, respectively. Although other measures are also useful to evaluate the performance, we care more about the precision of the predictions. Intuitively, if our precision is higher, we have a higher probability of picking out some stocks whose prices will go up. All the experiments in the following section were repeated ten times.

C. Predictive Performance

We conducted experiments to compare 1-step, 3-step, and 6-step price movement prediction performance on five real-world datasets. The evaluation results are summarized in Tables II–IV. Comparing the independent prediction and learning graph-based methods, almost all learning graph-based methods outperform the independent prediction methods throughout five datasets, which confirms that learning graph-based methods is a promising technology for stock movement predictions. From Tables II– IV, we can find that our proposed GERU and clu-GERU consistently outperform all baselines throughout five real-world datasets and two different financial markets. For instance, compared with the best baselines on 1-step predictions, the GERU's accuracy improvements are 5.0%, 2.2%, 4.8%, 4.6%, and 4.9% on the five datasets, respectively.

TABLE II
1-Step Predictive Performance Comparison Among Various Methods

Datasets	CSI 100		CSI 300		S&P 100		Rus 1K		Rus 3K	
Metrices	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Pre
LSTM	53.0 ± 1.1	51.1 ± 1.6	51.9 ± 1.4	50.5 ± 1.3	54.2 ± 1.5	51.0 ± 2.7	52.9 ± 1.4	53.1 ± 1.4	52.1 ± 1.2	51.0 ± 1.5
ALSTM	53.6 ± 1.4	50.8 ± 2.6	52.7 ± 0.9	51.5 ± 1.3	55.2 ±1.4	47.3 ± 1.4	53.0 ± 1.2	53.0 ± 0.5	52.6 ± 1.0	52.4 ± 1.4
Adv-LSTM	52.0 ± 1.8	50.2 ± 2.3	51.1 ± 1.7	50.8 ± 1.5	54.8 ± 1.6	48.1 ± 3.1	51.9 ± 1.0	53.9 ± 1.2	53.0 ± 0.6	54.0 ± 0.6
CSG-GCN	51.7 ± 1.7	49.7 ± 2.0	51.5± 1.1	48.3 ± 1.9	53.1 ± 1.9	53.7 ± 1.4	51.9 ± 1.1	52.3 ± 0.2	52.6 ± 1.7	52.1 ± 1.4
DSG-GCN	52.3 ± 1.8	51.4 ± 2.6	51.9 ± 1.5	50.5 ± 1.2	53.7 ± 1.8	52.1 ± 2.6	52.7 ± 1.3	52.3 ± 0.8	50.5 ± 0.6	50.3 ± 0.3
FinGAT	53.5 ± 1.0	51.5 ± 2.4	53.0 ± 1.3	51.7 ± 1.6	55.1 ± 1.2	48.3 ± 2.5	52.0 ± 1.3	53.8 ± 0.5	53.5 ± 1.5	53.0 ± 0.2
HATS	53.7 ± 1.4	50.9 ± 2.9	53.2 ± 1.4	52.0 ± 1.5	55.7 ± 0.0	49.0 ± 1.7	53.7 ± 1.3	53.0 ± 0.5	54.4 ± 0.6	54.9 ± 0.5
HAD-GNN	53.8 ± 1.5	52.0 ± 2.0	53.6 ± 0.7	52.3 ± 1.2	55.1 ± 0.1	52.0 ± 2.0	53.9 ± 1.3	54.2 ± 0.7	55.0 ± 0.5	55.1 ± 0.6
MTGNN	53.7 ± 1.6	51.8 ± 2.4	53.3 ± 0.8	51.8 ± 1.3	55.2 ± 0.1	52.2 ± 2.4	53.7 ± 1.4	54.1 ± 0.8	54.7 ± 0.5	54.7 ± 0.5
AGCRN	53.9 ± 1.3	52.3 ± 2.4	53.6 ± 0.5	52.0 ± 1.8	56.3 ± 0.1	52.1 ± 2.3	54.0 ± 1.4	54.0 ± 0.5	55.1 ± 0.4	55.2 ± 0.4
SDGNN	54.0 ± 1.4	52.5 ± 2.5	53.8 ± 0.6	52.2 ± 1.6	56.0 ± 0.2	52.2 ± 2.1	54.2 ± 1.5	54.3 ± 0.6	55.3 ± 0.6	55.4 ± 0.5
GERU	**56.7 ± 1.1****	**55.9 ± 2.6***	55.0 ± 0.5***	**57.5 ± 1.8*****	**59.0 ± 0.1*****	**60.1 ± 2.1*****	56.7 ± 0.2**	57.5 ± 0.3***	58.0 ± 1.4***	58.7 ± 0.8***
clu-GERU	56.5 ± 1.2*	54.2 ± 2.0	**56.1 ± 0.8*****	57.2 ± 1.7***	57.9 ± 1.5	56.1 ± 2.6	**57.4 ± 0.3*****	**58.8 ± 0.5*****	**59.1 ± 0.7*****	**59.8 ± 0.8*****

We conducted a paired samples t-test between the **GERU/clu-GERU** and the strongest baseline. *: p-value < 0.1; **: p-value < 0.05, ***: p-value < 0.001.

TABLE III
3-Step Predictive Performance Comparison Among Various Methods

Datasets	CSI 100		CSI 300		S&P 100		Rus 1K		Rus 3K	
Metrices	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Pre
LSTM	54.0 ± 1.7	55.5 ± 2.2	52.4 ± 1.3	54.3 ± 1.3	57.6 ± 1.5	59.5 ± 1.9	56.2 ± 1.2	56.9 ± 2.0	54.0 ± 1.4	52.3 ± 1.2
ALSTM	54.3 ± 1.4	56.0 ± 1.7	53.3 ± 1.1	54.1 ± 1.6	57.8 ± 0.0	57.9 ± 0.0	56.1 ± 1.7	56.0 ± 1.8	53.4 ± 1.6	53.6 ± 0.7
Adv-LSTM	53.2 ± 1.4	55.3 ± 1.6	52.8 ± 1.3	53.9 ± 0.8	57.3 ± 0.0	57.6 ± 1.7	56.5 ± 0.2	56.5 ± 0.4	53.8 ± 1.5	53.9 ± 0.3
CSG-GCN	55.2 ± 1.7	56.0 ± 1.4	53.1 ± 1.5	53.3 ± 1.6	57.9 ± 1.2	57.5 ± 0.0	51.2 ± 1.6	52.4 ± 1.1	54.0 ± 1.0	54.5 ± 1.0
DSG-GCN	55.7 ± 1.8	56.3 ± 1.1	53.5 ± 1.2	53.9 ± 1.2	58.3 ± 1.7	59.2 ± 1.8	51.7 ± 1.4	52.4 ± 1.4	53.7 ± 0.5	52.9 ± 0.7
FinGAT	54.3 ± 1.7	57.5 ± 1.1	53.1 ± 1.0	53.9 ± 0.5	58.3 ± 2.2	58.2 ± 3.0	51.2 ± 1.2	54.1 ± 1.6	53.8 ± 0.5	54.4 ± 0.4
HATS	54.7 ± 2.1	56.9 ± 1.9	53.7 ± 1.5	54.1 ± 1.2	58.6 ± 2.1	61.0 ± 2.2	54.2 ± 1.3	56.3 ± 0.9	54.3 ± 0.0	54.5 ± 0.0
HAD-GNN	57.0 ± 1.5	57.8 ± 1.2	54.1 + 1.8	54.6 ± 1.0	59.3 ± 0.3	59.6 ± 1.3	54.1 ± 0.9	57.0 ± 0.6	54.8 ± 0.4	54.9 ± 0.6
MTGNN	56.9 ± 1.6	57.9 ± 1.3	53.9 ± 1.6	54.5 ± 1.4	58.9 ± 0.1	59.3 ± 1.4	54.4 ± 1.2	57.2 ± 0.8	55.0 ± 0.5	55.0 ± 0.3
AGCRN	56.0 ± 1.0	56.4 ± 1.7	53.9 ± 1.5	54.5 ± 1.3	59.6 ± 1.4	61.2 ± 1.0	55.2 ± 0.8	57.4 ± 1.2	55.9 ± 0.8	56.1 ± 0.9
SDGNN	56.6 ± 1.2	56.8 ± 1.8	54.0 ± 1.3	54.7 ± 0.9	59.4 ± 1.2	60.9 ± 1.3	55.7 ± 1.0	57.5 ± 1.0	56.0 ± 0.6	56.3 ± 1.0
GERU	59.2 ± 1.3*	**60.4 ± 1.5****	56.7 ± 1.3**	**57.2 ± 1.2****	**65.2 ± 0.0*****	**69.2 ± 0.0*****	60.4 ± 0.8***	60.3 ± 1.1**	58.6 ± 0.4***	57.6 ±1.1*
clu-GERU	**61.0 ± 1.2****	60.0 ± 1.3**	**56.9 ± 1.8****	55.5 ± 1.0	63.1 ± 0.0***	63.2 ± 0.1**	**61.4 ± 1.4*****	**61.3 ± 1.6****	**59.3 ± 0.3*****	**58.9 ± 0.7*****

We conducted a paired samples t-test between the **GERU/clu-GERU** and the strongest baseline. *: p-value < 0.1; **: p-value < 0.05, ***: p-value < 0.001.

TABLE IV
6-Step Predictive Performance Comparison Among Various Methods

Datasets	CSI 100		CSI 300		S&P 100		Rus 1K		Rus 3K	
Metrices	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Pre
LSTM	54.9 ± 1.9	55.4 ± 1.8	53.6 ± 0.8	53.7 ± 1.2	58.8 ± 0.1	59.2 ± 1.2	57.6 ± 1.4	57.1 ± 1.3	53.4 ± 1.8	53.6 ± 1.5
ALSTM	55.5 ± 1.8	55.2 ± 1.9	53.8 ± 1.8	53.2 ± 0.8	59.3 ± 0.1	59.6 ± 0.1	57.6 ± 0.2	57.2 ± 0.4	53.6 ± 0.0	56.0 ± 0.0
Adv-LSTM	55.6 ± 1.8	56.0 ± 2.2	53.2 ± 1.6	50.7 ± 1.2	59.2 ± 0.1	59.5 ± 0.1	57.5 ± 1.3	57.1 ± 1.1	53.9 ± 0.4	54.0 ± 0.4
CSG-GCN	56.3 ± 1.4	56.1 ± 0.2	53.6 ± 0.0	53.6 ± 0.0	59.0 ± 2.8	58.9 ± 0.1	52.7 ± 1.9	54.6 ± 1.0	53.2 ± 1.1	53.6 ± 1.6
DSG-GCN	55.9 ± 1.6	56.7 ± 1.8	53.5 ± 0.3	54.0 ± 1.3	59.6 ± 0.1	59.9 ± 0.0	54.0 ± 1.5	55.1 ± 1.5	52.6 ± 1.2	53.9 ± 1.2
FinGAT	55.7 ± 1.6	56.4 ± 1.8	53.8 ± 1.4	53.6 ± 1.2	59.3 ± 0.0	60.0 ± 0.0	52.8 ± 1.7	54.4 ± 1.9	54.0 ± 0.5	54.1 ± 0.5
HATS	55.8 ± 1.4	56.2 ± 1.6	53.7 ± 1.8	53.9 ± 1.4	59.9 ± 0.0	60.0 ± 0.0	55.1 ± 1.2	57.0 ± 1.3	54.1 ± 0.0	56.2 ± 0.0
HAD-GNN	56.3 ± 1.1	57.0 ± 1.3	53.7 ± 0.8	54.2 ± 1.2	50.9 ± 0.0	62.0 ± 0.0	58.1 ± 0.3	58.7 ± 0.4	55.3 ± 0.6	57.2 ± 1.0
MTGNN	56.7 ± 1.3	57.5 ± 1.5	53.3 ± 0.7	53.8 ± 1.4	60.0 ± 0.0	62.2 ± 0.0	57.9 ± 0.1	58.4 ± 0.2	54.7 ± 0.5	56.7 ± 0.9
AGCRN	56.2 ± 1.2	57.1 ± 1.4	53.6 ± 1.2	54.1 ± 1.1	60.3 ± 1.7	61.7 ± 2.2	58.3 ± 0.6	59.4 ± 0.8	56.8 ± 0.3	57.1 ± 0.6
SDGNN	56.1 ± 1.3	57.3 ± 1.0	53.4 ± 0.9	53.7 ± 1.3	60.5 ± 1.5	62.1 ± 1.9	58.7 ± 0.8	59.6 ± 1.1	56.3 ± 0.2	56.8 ± 0.4
GERU	**60.3 ± 1.0*****	**60.4 ± 0.5*****	**57.1 ± 1.4****	**57.0 ± 1.3****	65.1 ± 0.0***	**66.3 ± 0.1*****	**63.6 ± 0.3*****	**63.8 ± 0.5*****	**60.0 ± 1.1*****	58.8 ± 1.4**
clu-GERU	59.9 ± 1.0**	60.0 ± 0.3**	56.7 ± 1.2**	56.4 ± 1.2*	**65.7 ± 0.0*****	66.1 ± 0.0***	63.4 ± 0.6***	56.4 ± 1.2***	**60.0 ± 0.8*****	**59.6 ± 1.7****

We conducted a paired samples t-test between the **GERU/clu-GERU** and the strongest baseline. *: p-value < 0.1; **: p-value < 0.05, ***: p-value < 0.001.

GERU also improved precisions by 6.5%, 9.9%, 11.9%, 5.9%, and 6.0% on the five datasets. Similarly, clu-GERU increased the accuracies by 4.6%, 4.3%, 2.8%, 5.9%, and 6.9%, and increased the precisions by 3.2%, 9.4%, 4.5%, 8.3%, and 7.9% on the five datasets, respectively. Such significant performance improvements indicate the superiority of GERU and clu-GERU for financial market predictions. In addition, we observed that clu-GERU tends to outperform GERU on large datasets, including Rus 1 K and Rus 3 K. A possible reason is that GERU with vanilla ADGL needs more GPU memory, so we can only use a small batch size to train the model.

Furthermore, the learning graph-based methods achieved better results than predefined graph-based methods in terms of accuracy and precision. It can be owed to the flexibility of learning latent stock relations. The dynamic graph-based models, including HAD-GNN, FinGAT, HATS, and SDGNN, slightly outperform the static models, which is consistent with the stock market characteristics. However, our proposed GERU and clu-GERU exhibit significant improvements over dynamic graph-based methods. We observe relative accuracy improvements of approximately 5% for CSI 100, Rus 1 K, and Rus 3 K datasets. These promising results, when compared to HAD-GNN, demonstrate the effectiveness of adaptively learning dynamic interactions between stocks using the ADGL/clu-ADGL module. Furthermore, the superiority of GERU over the other three dynamic methods validates the benefits of modeling topology and node attribute evolutions, as opposed to independently learning the dynamic graphs.

D. Model Analyses

This section conducted more detailed model analyses, including ablation study, robustness analysis, scalability analysis, and dynamic temporal stock graph analysis to closely look at the proposed GERU/clu-GERU.

TABLE V
PREDICTION ACCURACY COMPARISON BETWEEN GERU AND DIFFERENT COMBINATIONAL METHODS ON 1-STEP STOCK MOVEMENT PREDICTIONS

Method	CSI 100	CSI 300	S&P 100	Rus 1K	Rus 3K
CSG-GCN	$51.7 \pm 1.7^{***}$	$51.7 \pm 1.1^{***}$	$53.1 \pm 1.9^{***}$	$51.9 \pm 1.1^{***}$	$52.6 \pm 1.7^{***}$
DSG-GCN	$52.3 \pm 1.8^{**}$	$51.9 \pm 1.5^{***}$	$53.7 \pm 1.8^{***}$	$52.7 \pm 1.3^{***}$	$50.5 \pm 0.6^{***}$
SSG-GAT (FinGAT)	$53.5 \pm 1.0^{**}$	$53.0 \pm 1.3^{**}$	$55.1 \pm 1.2^{***}$	$52.0 \pm 1.3^{***}$	$53.5 \pm 1.5^{***}$
CSG-GGRU	$53.5 \pm 1.2^{**}$	$53.1 \pm 0.8^{***}$	$54.3 \pm 1.1^{***}$	$52.8 \pm 0.4^{***}$	$53.9 \pm 1.4^{***}$
DSG-GGRU	$53.9 \pm 1.4^{*}$	$53.9 \pm 0.9^{**}$	$55.8 \pm 0.6^{***}$	$53.7 \pm 0.2^{***}$	$52.7 \pm 0.9^{***}$
SSG-GGRU	$54.0 \pm 1.1^{**}$	$53.2 \pm 0.8^{***}$	$55.4 \pm 0.2^{***}$	$53.1 \pm 0.8^{***}$	$54.6 \pm 0.8^{***}$
DyDSG-GGRU	$54.4 \pm 0.9^{**}$	$54.0 \pm 1.1^{*}$	$56.5 \pm 0.9^{***}$	$54.4 \pm 0.3^{***}$	$56.8 \pm 0.4^{***}$
ADGL-GCN	$54.1 \pm 1.3^{*}$	$53.8 \pm 1.2^{**}$	$55.6 \pm 1.6^{**}$	$53.6 \pm 0.9^{***}$	$54.1 \pm 1.1^{***}$
GERU	**56.7 ± 1.1**	**55.0 ± 0.5**	**59.0 ± 0.1**	**56.7 ± 0.2**	**58.0 ± 1.4**

*: p-value < 0.1; **: p-value < 0.05, ***: p-value < 0.001.

1) Ablation Study: To evaluate the utility of each key component over GERU, we first developed two types of comparison methods of GERU by combining the different methods of building stock graphs and GNNs. We reported the average accuracy with a standard deviation over ten runs in Table V.

a) GGRU with predefined graph: To investigate the influence of ADGL, we replaced the learned graph of ADGL at per time point with static CSG, DSG, SSG, and dynamic DSG (DyDSG), respectively. DyDSG is built using a 12-month sliding window to calculate the distance of pairwise stocks. In Table V, GERU with ADGL has higher predictive performance than the methods with predefined static stock graphs or heuristically dynamic stock graphs, which indicates that ADGL is an efficient learning graph network to discover the underlying dependencies between stocks.

b) ADGL with GCN: ADGL-GCN learns the stock graph from raw time series $\widetilde{\mathbf{X}}_t$ at each time step and does not use the information $\mathbf{H}_{t-1}$ from the previous stock graph. And ADGL-GCN exploits GCN to encode the node embedding $\mathbf{E}_t$ instead of GGRU. In Table V, the GERU significantly surpasses the ADGL-GCN as GGRU enables the GERU to capture temporal and spatial patterns from the evolving stock graphs. Overall, our ADGL and GGRU modules can be deployed either separately or jointly, and they consistently boost the predictive performance.

2) Robustness Analysis: To investigate the influence of the sparsification indicator k, clusters c, learning rate, regularization weight, attention head H, and lag size T, we evaluate the model performance by varying a hyperparameter's value while fixing the other hyperparameters with optimal values. The effects of k and c are shown in Fig. 6 and 7. Fig. 6 shows that increasing k cannot improve the predictive performance, yet the smaller k cannot weaken the performance. For example, the GERU and clu-GERU achieve the best performance when $k = 1$ on the S&P 100 and CSI 300 datasets, respectively. It demonstrates that sparse dynamic stock graphs are a promising perspective to provide valuable predictive information. For cluster c, Fig. 7 illustrates that the optimal value of c is different on different datasets. The optimal value of c for clu-GERU is 5 on S&P 100 and 10 on CSI 100 and CSI 300. The validation performance is inconsistent with the testing counterpart on CSI 100. It is because the early stopping leads to premature terminations. The results for $c = 5$ and $c = 10$ will surpass $c = 1$ if we disable the early stopping setting.

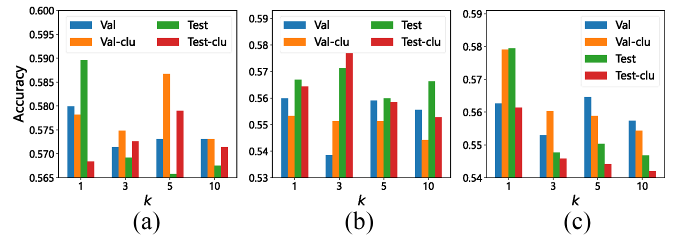


Fig. 6. Sensitivities of the GERU and the clu-GERU with respect to sparsification indicator k on the validation and testing of three datasets. (a) S&P 100. (b) CSI 100. (c) CSI 300.

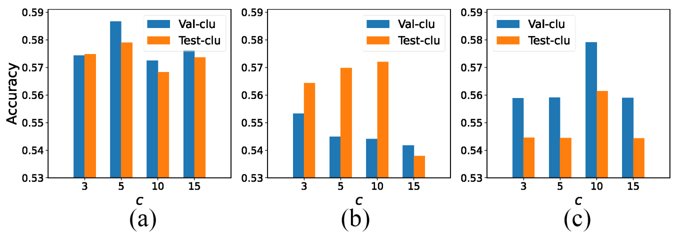


Fig. 7. Sensitivities of the clu-GERU with respect to the number of clusters c on the validation and testing of three datasets. (a) S&P 100. (b) CSI 100. (c) CSI 300.

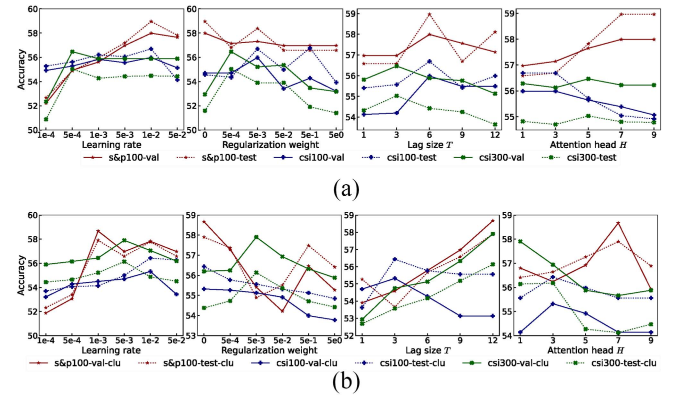


Fig. 8. Sensitivities of the GERU and the clu-GERU with respect to different hyperparameters on the validation and testing of three datasets. (a) Performance of the GERU. (b) Performance of the clu-GERU.

Fig. 8 shows the performance of GERU and clu-GERU on the validation and testing of the three datasets when adjusting the values of the learning rate, regularization weight, H, and T. We can find that in most cases, the curves of performance change smoothly with varying hyperparameters on the validation and testing datasets, which means that GERU and clu-GERU are not very sensitive to hyperparameters. The accuracy curves on the testing have similar trends to those on the validation. Such characteristic is conducive to choosing optimal values of hyperparameters. The only exception is that clu-GERU has significant bad results on S&P 100 datasets when regularization weight equals 1e-3 and 5e-3. We reviewed the training processes and found that early stopping leads to premature terminations. The performance will significantly increase if we disable the early stopping.

We have known from Fig. 6 that the optimal sparsification indicator k is usually less than 5. Thus, we fixed $k = 5$ and varied the value of H to observe the density of the learned dynamic stock graphs. The average density is listed

TABLE VI
AVERAGE DENSITY OF THE LEARNED STOCK GRAPHS CONCERNING TO H ON THREE DATASETS WHEN k IS FIXED AS 5

Method	H	CSI 100	CSI 300	S&P 100
GERU	1	0.0469	0.0020	0.0408
	3	0.0834	0.0330	0.0759
	9	0.1079	0.0442	0.0938
clu-GERU	1	0.0449	0.0019	0.0382
	3	0.0712	0.0316	0.0612
	9	0.0943	0.0420	0.0839

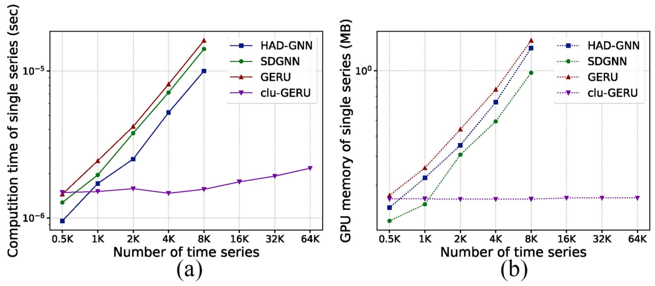


Fig. 9. (a) Time cost and (b) GPU memory comparisons of the dynamic models with the different number of time series.

in Table VI. We can find that the average density is still relatively small with the increase of H, which demonstrates that summing the multihead adjacency matrixes in (10) can maintain modest sparsity. The reason is that the learned multihead adjacency matrixes have many overlap edges.

3) Scalability Analysis: Our analysis has proved that the clu-ADGL reduces the ADGL's quadratic complexity to linear complexity. To further investigate the efficiency of the clu-ADGL in practice, we conducted a scalability analysis with synthetic time series data whose shape is identical to $\widetilde{\mathbf{x}}_t^s$. Given the same computing environment (11 GB GPU memory), we ran the HAD-GNN, SDGNN, GERU, and clu-GERU with the number of time series increasing from 0.5 to 64 K. We divided the overall time cost and GPU memory with the batch size, the number of time series, and T to calculate the time cost and GPU memory of single time series. As shown in Fig. 9, for the first three methods, the computational time and memory of single series scale linearly concerning the number of time series, and they are out of the memory when the number of time series is larger than 8 K. In contrast, with the number of time series increasing from 0.5 to 64 K, the clu-GERU shows an approximately constant computation time and memory for single series, which indicates that the clu-GERU has better scalability than other dynamic graph-based methods in practice.

4) Learned Stock Graphs: We first visualized the learned dynamic graphs in Fig. 10 to show the evolution of the stock market. Macroscopically, these stock graphs have similar structures and dominated stock nodes but moderate local link changes for GERU and clu-GERU at successive time steps. And the clu-GERU's graphs own transparent communities because clu-ADGL explicitly discovers the clusters with LHS K-means. To display the overall evolution of the learned dynamic graphs, we calculated the outcoming-edge weight sum of each stock by performing a summing along with $\mathbf{A}_t$'s columns. A larger weight sum means that the

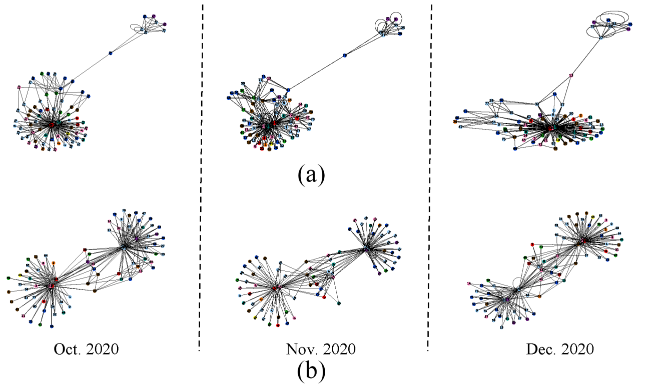


Fig. 10. Learned stock graphs ($k = 3, c = 5$) in the CSI 100 dataset from Oct. 2020 to Dec. 2020 by the (a) GERU (with ADGL) and the (b) clu-GERU (with clu-ADGL), respectively. The node color denotes the stock sector affiliation.

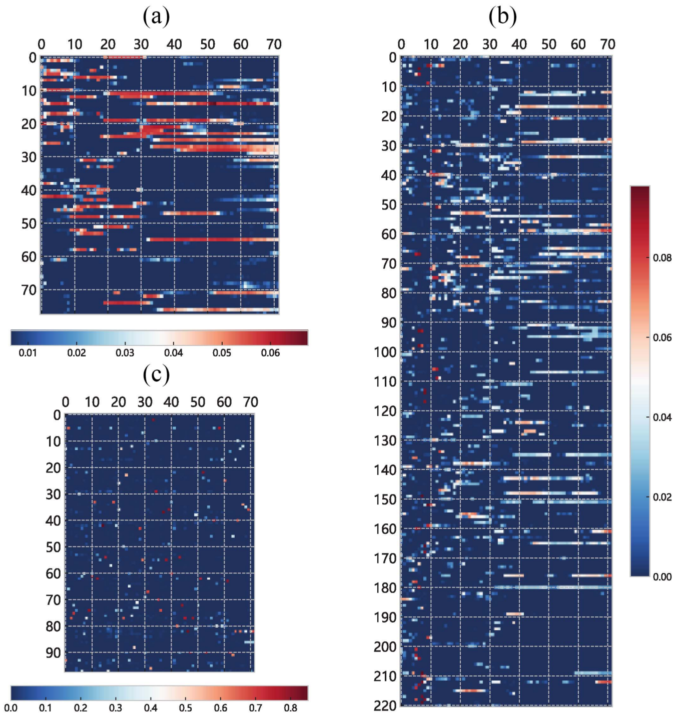


Fig. 11. Influence power of each stock evolves along the time axis. The horizontal axis represents time, and the longitudinal axis denotes the individual stock index. (a) CSI 100. (b) CSI 300. (c) S&P 100.

stock can provide more information for its neighbors' price movement predictions. The weights along the time axis are shown in Fig. 11, where each element represents the influence of one stock on its neighbor stocks at a specific time point.

From Fig. 11, we can observe an interesting finding that the evolution patterns of influence power between the Chinese stock market and the American stock market are very different. In the CSI 100 dataset and CSI 300 dataset, many influential stocks successively survive several months and eventually, decline and perish. By summing the weights of stocks that affiliate the same sector according to GICS and normalizing them, we obtain the sector influence evolving along the time axis on CSI 100 in Fig. 12. We can observe a significant sector rotation effect that the influential sectors transfer from some

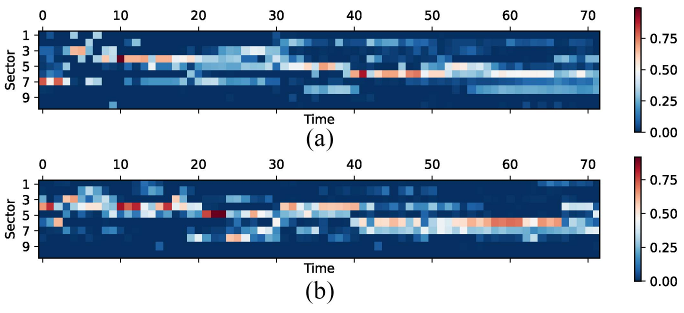


Fig. 12. Sector influence evolving along the time axis. (a) GERU. (b) clu-GERU.

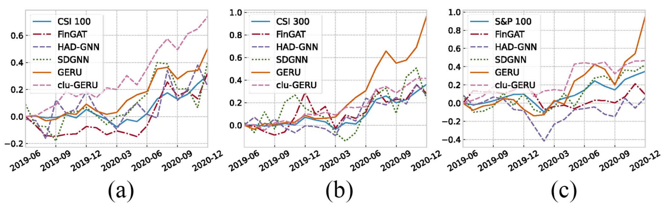


Fig. 13. Return ratio comparisons of different predictive models in the (a) CSI 100, (b) CSI 300, and (c) S&P 100 datasets, respectively. The notations CSI 100, CSI 300, and S&P 100 in these subfigures denote the three market indices.

sectors to others. However, the most influential stocks for the S&P 100 dataset are scattered over the time axis, as shown in Fig. 11 (c), which means that the transfer speeds of influential stocks are breakneck. Such difference between the two markets reveals that the financial market of developed countries has a more complicated and changeable investment pattern than the developing countries [39], [40].

5) Simulation Trading: To examine GERU's practical applicability, we designed a simulation trading on the CSI 100, CSI 300, and S&P100 datasets. The trading strategy is simple. Based on the binary classification probability from the models with their corresponding optimal hyperparameters, we selected ten stocks with the highest prediction probabilities of being positive class (going up) to combine a portfolio with an equal split of budget. Due to the different chosen time points of monthly features and the trading costs, we adopted different trading settings on the Chinese and American markets. For CSI 100 dataset and CSI 300 dataset, we conducted the trading on the last trading day of each month with a 0.2% trading cost and held the portfolio for one month. For S&P 100 dataset, we conducted the trading on the first trading day of each month with a 0.02% trading cost and held the portfolio for one month. Fig. 13 compares the return ratio of different predictive models on simulation trading. As shown in Fig. 13, GERU and clu-GERU consistently achieve the best performance on return ratio across different datasets, which demonstrates their superior profitability. Although simulation trading simplifies real-world trading, its results are still promising.

VI. Conclusion and Future Work

This article studied the stock movement prediction problem and proposed a novel dynamic graph-based prediction framework, GERU. The GERU can jointly learn the dynamic dependencies between stocks from the historical stock features and capture both topology and node attribute evolutions in dynamic graphs. Our proposed ADGL/clu-ADGL can generate sparse and meaningful time-varying stock graphs. The ADGL also provides a new perspective to utilize the learned attention matrixes, i.e., we can directly use the learned attention matrixes as graph topologies to capture the latent dependencies between time series. The proposed clu-ADGL can maintain comparable performance with the vanilla ADGL and have high efficiency in memory and computation. The GERU's prediction performance was verified through experiments on the five real-world datasets. The simulation trading also demonstrates GERU's profitability in stock investments. However, due to the heavy computational burden, our method could not provide real-time forecasts. Therefore, it is more suitable for low-frequency such as weekly or monthly prediction tasks.

There exist many possibilities for future research. Our system only uses historical trading information, while many other factors, such as social media and influence stock movements. Social media contains varied information sources, including natural language, images, and videos. Such multimodal signals have new opportunities for market predictions [41]. In future work, we could develop a new graph-based learning method to utilize trading data, interfirm relationships, and multimodal social media data.

References

[1] M. M. Ferdaus, R. K. Chakrabortty, and M. J. Ryan, "Multiobjective automated type-2 parsimonious learning machine to forecast time-varying stock indices online," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 5, pp. 2874–2887, May 2022, doi: 10.1109/TSMC.2021.3061389.

[2] W. Jiang, "Applications of deep learning in stock market prediction: Recent progress," *Expert Syst. Appl.*, vol. 184, Dec. 2021, Art. no. 115537, doi: 10.1016/j.eswa.2021.115537.

[3] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, Mar. 2017, doi: 10.1109/TNNLS.2016.2522401.

[4] X. Gabaix, P. Gopikrishnan, V. Plerou, and H. E. Stanley, "A theory of power-law distributions in financial market fluctuations," *Nature*, vol. 423, no. 6937, pp. 267–270, 2003, doi: 10.1038/nature01624.

[5] H. Tian, X. Zheng, and D. D. Zeng, "Analyzing the dynamic sectoral influence in Chinese and American stock markets," *Phys. Stat. Mech. Appl.*, vol. 536, Dec. 2019, Art. no. 120922, doi: 10.1016/j.physa.2019.04.158.

[6] S. K. Stavroglou, A. A. Pantelous, H. E. Stanley, and K. M. Zuev, "Hidden interactions in financial markets," *Proc. Nat. Acad. Sci.*, vol. 116, no. 22, pp. 10646–10651, 2019, doi: 10.1073/pnas.1819449116.

[7] C. J. Kock, "When the market misleads: Stock prices, firm behavior, and industry evolution," *Org. Sci.*, vol. 16, no. 6, pp. 637–660, 2005, doi: 10.1287/orsc.1050.0141.

[8] D. Kim and Y. Qi, "Accruals quality, stock returns, and macroeconomic conditions," *Account. Rev.*, vol. 85, no. 3, pp. 937–978, 2010, doi: 10.2308/accr.2010.85.3.937.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386.

[10] P. W. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*.

[11] Y. Chen, Z. Wei, and X. Huang, "Incorporating corporation relationship via graph convolutional neural networks for stock price prediction," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manag.*, Turin, Italy, Oct. 2018, pp. 1655–1658.

[12] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T.-S. Chua, "Temporal relational ranking for stock prediction," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, pp. 1–30, 2019, doi: 10.1145/3309547.

[13] R. Kim, C. H. So, M. Jeong, S. Lee, J. Kim, and J. Kang, "HATS: A hierarchical graph attention network for stock movement prediction," 2019, *arXiv:1908.07999*.

[14] J. Ye, J. Zhao, K. Ye, and C. Xu, "Multi-graph convolutional network for relationship-driven stock movement prediction," in *Proc. 25th Int. Conf. Pattern Recognit.*, Milan, Italy, Jan. 2021, pp. 6702–6709, doi: 10.1109/ICPR48806.2021.9412695.

[15] G. G. Creamer, Y. Ren, and J. V. Nickerson, "Impact of dynamic corporate news networks on asset return and volatility," in *Proc. Int. Conf. Soc. Comput.*, 2013, pp. 809–814, doi: 10.1109/SocialCom.2013.121.

[16] X.-Q. Sun, H.-W. Shen, and X.-Q. Cheng, "Trading network predicts stock price," *Sci. Rep.*, vol. 4, no. 1, pp. 1–6, 2014, doi: 10.1038/srep03711.

[17] H. Cao, T. Lin, Y. Li, and H. Zhang, "Stock price pattern prediction based on complex network and machine learning," *Complexity*, vol. 2019, May 2019, Art. no. 4132485, doi: 10.1155/2019/4132485.

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[19] H. Tian, X. Zheng, K. Zhao, M. W. Liu, and D. D. Zeng, "Inductive representation learning on dynamic stock co-movement graphs for stock predictions," *Inf. J. Comput.*, vol. 34, no. 4, pp. 1940–1957, 2022, doi: 10.1287/ijoc.2022.1172.

[20] J. V. D. M. Cardoso and D. P. Palomar, "Learning undirected graphs in financial markets," in *Proc. 54th Asilomar Conf. Signals Syst. Comput.*, 2020, pp. 741–745, doi: 10.1109/IEEECONF51394.2020.9443573.

[21] D. Marinazzo, M. Pellicoro, and S. Stramaglia, "Kernel-Granger causality and the analysis of dynamical networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 77, no. 5, 2008, Art. no. 56215, doi: 10.1103/PhysRevE.77.056215.

[22] 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 *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, Aug. 2020, pp. 753–763.

[23] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *Proc. 33rd Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 33, Dec. 2020, pp. 17804–17815. [Online]. Available: https://proceedings.neuips.cc/paper/2020/file/ce1aad92b939420fc17005e5461e6f48-Paper.pdf

[24] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, Feb. 2021, pp. 4027–4035.

[25] Y. He, Q. Li, F. Wu, and J. Gao, "Static-dynamic graph neural network for stock recommendation," in *Proc. 34th Int. Conf. Sci. Stat. Database Manag.*, Copenhagen, Denmark, Jul. 2022, pp. 1–4.

[26] Y.-L. Hsu, Y.-C. Tsai, and C.-T. Li, "FinGAT: Financial graph attention networks for recommending top-*K*K profitable stocks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 469–481, Jan. 2023, doi: 10.1109/TKDE.2021.3079496.

[27] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 6000–6010. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[28] M. Geva, R. Schuster, J. Berant, and O. Levy, "Transformer feed-forward layers are key-value memories," 2020, *arXiv:2012.14913*.

[29] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[30] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient content-based sparse attention with routing transformers," *Trans. Assoc. Comput. Linguist.*, vol. 9, pp. 53–68, 2021, doi: 10.1162/tacl_a_00353.

[31] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[32] A. Shrivastava and P. Li, "Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS)," in *Proc. 27th Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 2. Montreal, QC, Canada, Dec. 2014, pp. 2321–2329. [Online]. Available: https://proceedings.neurips.cc/paper/2014file/310ce61c90f3a46e340ee8257bc70e93-Paper.pdf

[33] A. Hasanzadeh, E. Hajiramezanali, K. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Variational graph recurrent neural networks," in *Proc. 32nd Int. Conf. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2019, pp. 10701–10711. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/a6b8deb7798e7532ade2a8934477d3ce-Paper.pdf

[34] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with LSTM neural networks," in *Proc. Int. Joint Conf. Neural Net. (IJCNN)*, May 2017, pp. 1419–1426, doi: 10.1109/IJCNN.2017.7966019.

[35] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2627–2633, doi: 10.24963/ijcai.2017/366.

[36] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T.-S. Chua, "Enhancing stock movement prediction with adversarial training," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 5843–5849, doi: 10.24963/ijcai.2019/810.

[37] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–10. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[38] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," 2019, *arXiv:1903.02428*.

[39] S. Kundu and N. Sarkar, "Return and volatility interdependences in up and down markets across developed and emerging countries," *Res. Int. Bus. Finance*, vol. 36, pp. 297–311, Jan. 2016, doi: 10.1016/j.ribaf.2015.09.023.

[40] W. Wang, "Research on consumption upgrading and retail innovation development based on mobile Internet technology," in *Proc. J. Phys. Conf. Ser.*, Mar. 2019, Art. no. 42070, doi: 10.1088/1742-6596/1176/4/042070.

[41] Q. Li, J. Tan, J. Wang, and H. C. Chen, "A multimodal event-driven LSTM model for stock prediction using online news," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 10, pp. 3323–3337, Oct. 2021, doi: 10.1109/TKDE.2020.2968894.

[42] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–10. [Online]. Available: https://openreview.net/forum?id=SJiHXGWAZ

Hu Tian received the B.S. degree in mathematics from the School of Mathematics and Physics, Chinese University of Geosciences, Wuhan, China, in 2018. He is currently pursuing the Ph.D. degree in social computing with the Institute of Automation, Chinese Academy of Sciences, Beijing, China.

His research interest includes machine learning and intelligent system, particularly in machine learning techniques and applications, such as graph representation learning, FinTech, and social computing.

Xingwei Zhang received the B.S. degree in electronic engineering from the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China, in 2014, and the Ph.D. degree in computer application technology from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2023.

He is currently an Assistant Research Professor with the Institute of Automation, Chinese Academy of Sciences. His research interests include artificial intelligence, Internet or Things, information security, and cross-modal retrieval.

Xiaolong Zheng (Member, IEEE) received the B.S. degree from China Jiliang University, Hangzhou, China, in 2003, the M.S. degree from Beijing Jiaotong University, Beijing, China in 2006, and the Ph.D. degree in social computing from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2009.

He is currently a Research Professor with the Institute of Automation, Chinese Academy of Sciences. His research interest includes artificial intelligence, complex networks, security informatics, social computing, and recommender system.

Daniel Dajun Zeng (Fellow, IEEE) received the B.S. degree in economics and operations research from the University of Science and Technology of China, Hefei, China, in 1990, and the M.S. and Ph.D. degrees in industrial administration from Carnegie Mellon University, Pittsburgh, PA, USA, in 1994 and 1998, respectively.

He is currently a Research Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include intelligence and security informatics, infectious disease informatics, social computing, recommendation systems, software agents, and applied operations research and game theory.