2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00017

Relational Temporal Graph Convolutional Networks for Ranking-Based Stock Prediction

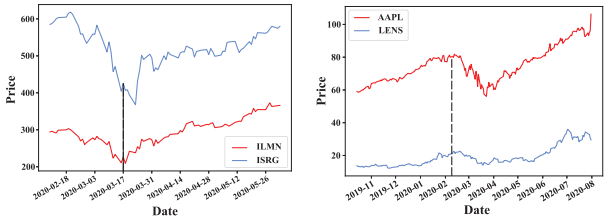
Zetao Zheng[†] Jie Shao[†‡*] Jia Zhu[§] Heng Tao Shen[†‡]
[†]University of Electronic Science and Technology of China, Chengdu, 611731, China
[‡]Sichuan Artificial Intelligence Research Institute, Yibin, 644000, China
[§]Zhejiang Normal University, Jinhua, 321004, China
ztzheng@std.uestc.edu.cn shaojie@uestc.edu.cn jiazhu@zjnu.edu.cn shenhengtao@hotmail.com

***Abstract*—Stock prediction is an attractive topic in fintech. However, traditional solutions for stock prediction have two drawbacks: (1) Some focus on the temporal patterns of stocks and model each stock as an independent individual but neglect their relations. Some models consider the relations among stocks, but work in a two-step format (i.e., capturing the temporal patterns first and then considering the relation dependency), which makes them complex and inefficient; (2) They model the stock prediction as a regression (predicting stock price) or classification task (predicting stock trend), which cannot optimize the target of investment, i.e., selecting the best stocks from the exchange market with the highest expected revenue in the future. To fully utilize the relations among stocks and achieve the highest revenue, a relation-temporal graph convolutional network (RT-GCN) is proposed. We first model the relations among stocks and their daily features into a relation-temporal graph. Then, we apply RT-GCN and three relation-aware strategies to realize the relation-temporal feature extraction for each stock. Finally, the features are fed for score calculation in a learning-to-rank way, and the stock with the highest score represents the highest investment revenue in the future. Extensive experiments demonstrate the effectiveness and efficiency of our method.**

***Index Terms*—stock prediction, graph-based learning, stock ranking**

I. Introduction

Stock trading has attracted the attention of many investors in recent years. Whether an investor can gain profit in the stock market depends heavily on whether he/she can buy or trade the stock at an appropriate time. Therefore, accurate prediction of the trend of stocks is the key to make the right investment decisions. With computer technology assistance, considerable efforts have been devoted to developing AI techniques (e.g., deep learning approaches) for stock movement prediction. For instance, state frequency memory (SFM) [1] is an excellent model for capturing short-long-term temporal frequency and obtains impressive performance on stock prediction. Liu et al. [2] proposed multi-scale two-way deep neural network (MTDNN) to automatically learns multi-scale patterns from wavelet-based and downsampling-based information by using eXtreme gradient boosting [3] and recurrent neural network (RNN) [4], respectively. DA-RNN [5] is a novel model to capture long-term temporal dependencies with a dual attention mechanism appropriately. However, the aforementioned works only consider the predicted stock as an individual and ignore the connection among the stocks. It is common knowledge that corporations (stocks) are connected broadly via various relationships with the development of financial market. The price of the target stock would be affected by corporations that are related. For example, stock Illumina (ILMN) is related to the stock Intuitive Surgical (ISRG) because they both belong to the medicine industry. As shown in Figure 1(a), their stock prices increased rapidly after the outbreak of COVID-19. Another example is the stock Apple (AAPL) and Lens Technology (LENS). The supplier-customer relation between them has a larger impact on the stock price of LENS, especially in the period of releasing a new version of iPhone, as Figure 1(b) displays. Therefore, in this work, we propose to take the relations of related stocks into account when predicting the stock movement instead of treating them as individual ones.



(a) Same industry relation (b) Supplier-consumer relation

Fig. 1. Two examples of stock price history to illustrate the impact of company relations on the stock price.

Significant works about the combination with relations among stocks have been done in related work. For example, Deng et al. [6], Li et al. [7] and Li et al. [8] propose to consider the extra information such as events or news of related stocks when predicting the stock trend. Their works are more inclined to explore the impact of related event/new natures (positive or negative) on stock movement. Moreover, we argue that such prediction methods are suboptimal to guide stock selection, since they regard the stock prediction as either a classification or regression task rather than selecting the top stocks with the highest expected revenue. Compared with the previous works, Feng et al. [9], Sawhney et al. [10] and Hsu et al. [11] propose to rank the stocks according to the ranking score and

Corresponding author: Jie Shao. This work was supported by the National Natural Science Foundation of China (grants No. 62276047 and No. 61832001).

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00017

return the top-N stocks with the highest score, which indicates the highest investment revenue, to help investors obtain the highest profit. Their work extracts the stock features in a two-step format, capturing the temporal patterns first and then considering the relational dependency. More specifically, Feng et al. [9] use long short-term memory (LSTM) to capture the temporal features and then devise a graph neural network to capture the relational dependency; Sawhney et al. [10] model the complex relations among stocks through a hypergraph and capture the temporal features by using the Hawkes attention mechanism [12]. Hsu et al. [11] use gated recurrent unit (GRU) to extract the temporal patterns first and then use the graph attention network to achieve the interactions among the stocks. Although these models have achieved impressing performance, calculating the relational temporal features of the stock using different modules is an inefficient option.

In this work, we propose to model the relations among stocks and their daily features into a relation-temporal graph and propose RT-GCN to extract the relation-temporal features for each stock cooperating with three graph propagation strategies, and finally score each stock according to the features. The higher scores indicate higher investment revenue. After modeling the stocks and their daily features into a relation-temporal graph, the graph neural network can operate on it and synchronously capture the feature evolution on temporal dimension and the relation dependency on spatial dimension, which is beyond the reach of other methods we mentioned above [9]–[11]. Furthermore, the pure convolution structure makes our model more efficient. Experiments on three real-world stock markets NASDAQ, NYSE and CSI demonstrate that our model achieves impressive performance not only on investment revenue but also training speed compared with other models. Our technical contributions are three-fold:

- To fully leverage each stock's relational and temporal information, we model the stock relations and the temporal feature of each stock as a relation-temporal graph, which is the basis for the application of RT-GCN and the premise of the efficiency of our model.
- We propose RT-GCN for ranking-based stock prediction task. This model is built with pure convolutional structures to simultaneously capture the temporal dynamics and relational dependencies, enabling faster training speed. Such an improvement is significant for the short-term stock trading system.
- We design three relation-aware propagation strategies in RT-GCN to meet the demands in the stock prediction task.
- Extensive experiments conducted on the NASDAQ, NYSE, and CSI datasets demonstrate that RT-GCN can outperform state-of-the-art methods in terms of effectiveness and efficiency.

The rest of the paper is organized as follows. We briefly introduce the related work in Section II. Section III introduces the preliminary knowledge about the task of ranking-based stock prediction. After that, Section IV presents our proposed model. The experimental evaluations are reported in Section V. Finally, this study is concluded in Section VI.

II. Related Work

Generally, the related work of our research can be divided into the following three categories: stock prediction, graph neural network, and graph-based stock prediction.

A. Stock Prediction

Traditional stock prediction methods such as technical analysis [13], [14] take much effort to select useful indicators and then predict the stock's movement according to these selected indicators. Normally, fundamental variables such as opening price, closing price, trading volume, and highest price are typically chosen as indicators to predict stocks' trend. However, such methods require background knowledge about economics and finance to analyze which indicators are appropriate for the stock. Besides, these handcraft indicators are often criticized for their poor generalization capacity [15]. Compared with traditional models, deep-learning-based models are more potent in discovering temporal dependency without requiring substantial human feature engineering. There have already existed some deep-learning-based stock predictions. Bao et al. [16] propose to obtain the high-level denoising features from Wavelet transform [17] as well as stacked autoencoders, and feed these high-level features to LSTM to obtain the next day's closing price. MTDNN [2] automatically learns multi-scale patterns from wavelet-based and downsampling-based information by using eXtreme gradient boosting [3] and RNN [4], respectively. Zhang et al. [1] propose to capture multi-frequency features from historical market data to make short and long-term predictions using state frequency memory (SFM) recurrent network. Qin et al. [5] propose dual-stage attention-based RNN (DA-RNN), capturing long-term temporal dependencies appropriately with an attention mechanism. Unlike the above-described deep-learning-based methods that train the models by minimizing the gap between predicted values and truth values, reinforcement-learning-based methods learn dynamically by adjusting actions based on feedback interacting with the environment to maximize the reward. Carta et al. [18] exploit an ensemble Q-learning strategy to minimize the over-fitting problem and maximize the return profit. Liu et al. [19] propose an adaptive model which introduces imitation learning to leverage classical trading strategies to improve trading agents automatically. These deep-learning-based and reinforcement-learning-based models are time-series sequence input and achieve excellent performance, but they ignore the stocks' relations information and treat each stock as an isolated individual.

B. Graph Neural Network

The target of graph-based learning is to learn an optimal representation for each node and edge to achieve the best performance for a specific task. Compared with the sequence-based deep model, graph-based structure neural network has a more powerful expression ability [20] as it represents each node with a fixed dimension vector that includes the information aggregated from the node's neighborhood. Due to its convincing performance, GNN has been applied in various

research areas. For example, Rhee et al. [21] leverage graph convolution and relation network for breast cancer subtype classification. Schlichtkrull et al. [22] adopt graph neural network in knowledge graph representation learning. It achieves highlight performance on dealing with the highly multi-relational data characteristic of realistic knowledge bases. Yan et al. [23] propose a robust model called spatial-temporal graph convolutional networks automatically learning both the spatial and temporal patterns and achieve impressive performance in skeleton-based action recognition. The introduction about the combination of GNN and stock prediction will be described in the next subsection.

C. Graph-based Stock Prediction

Chen et al. [24] demonstrate that incorporating relations among stocks can make prediction more accurate since it is beneficial to capture the evolution of stream data among different stocks. Substantial efforts have been made to incorporate graph structure for stock prediction in recent years. By utilizing the external knowledge, Deng et al. [6] and Li et al. [7] propose a knowledge-driven temporal convolutional network (KDTCN) and an event-driven LSTM model for stock trend prediction, respectively. They improve the stock prediction by combining the price values and the structured events extracted from financial news. Li et al. [8] propose an LSTM relational graph convolutional network to make use of the connection among stocks, and at the same time, explore the impact of overnight news on the stock movement. It should be noted that the works [6]–[8] referred above concentrate on the impact of external information such as events or news on the stock movement. Different from [6], [8], our datasets do not contain any side information such as news or events. Hsu et al. [11] devise a multi-task model to recommend profitable stocks, and at the same time, predict the stock movement by utilizing the fully-connected sector graph. Feng et al. [9] propose relational stock ranking (RSR) to formulate stock prediction as a ranking task and demonstrate the potential of learning-to-rank methods for predicting stocks. RSR encodes the temporal pattern of stock into sequential embeddings by applying LSTM and then revises the embeddings by accounting for stock relations. These revised embeddings are fed into a fully connected network to calculate the ranking score. Sawhney et al. [10] propose spatio temporal hypergraph attention network for stock ranking (STHAN-SR) to model the complex relations between stocks through a hypergraph and capture the temporal features by using the Hawkes attention mechanism [12]. Although [9]–[11] have achieved impressive performance, the LSTM structure adopted in [9], [11] seriously limits the framework's training speed and the relational features as well as the temporal features are calculated separately in [10], which makes it inconvenient.

Different from [9]–[11], our work is more simple and efficient. We model the relations among the stocks and their temporal features into a relation-temporal graph. The simple convolutional kernel operates on the relation-temporal graph to extract relational and temporal features for stock prediction without requiring the other models to achieve the feature extraction. Our model is purely built with convolutional structures to enable much faster training speed, and this improvement is significant for short-term stock trading system.

TABLE I
A SUMMARY OF NOTATIONS.

Notation	Description
X_t	Stock feature at time-step t
N	Number of stocks
T	Length of days used to predict the stock
D	Dimension of features of each stock
$\mathcal{A}$	Relation matrix of the stocks
K	Number of relation types
G_{RT}	Relational temporal graph
G_R	Relational graph
Θ	Convolution filters
F	Number of relational convolution filters
H	Number of temporal convolution filters

III. PRELIMINARIES

This section introduces the definition of the ranking-based stock prediction, relation-temporal graph, and graph convolution formulation. We summarize the frequently used notations in Table I.

A. Ranking-based Stock Prediction

Traditional stock prediction is to learn a prediction function $y = f(X_t)$ which maps the stock feature $X_t = [x_{(t-T+1)}, ..., x_t] \in \mathbb{R}^{T\times D}$ into the target label space at time-step t, T (window size) represents the length of the time sequence before prediction time-step $t+1$, and D is the dimension of features at each time-step. Different from traditional prediction which treats different stocks as isolated ones, the ranking-based stock prediction aims at learning a ranking function $r_{t+1} = R(X_t, \mathcal{A})$ to map a bunch of stock's feature X_t into a ranking list while considering the stock relations $\mathcal{A}$ simultaneously. Here $X_t \in \mathbb{R}^{T\times N\times D}$, N is the number of stocks, and T and D are the same definitions as traditional stock prediction. $\mathcal{A} \in \mathbb{R}^{N\times N\times K}$, K indicates the type of relations, and the pairwise relation between two stocks is encoded as a multi-hot binary vector. For instance, suppose we have three relations named supplier-customer, funder-by, same-industry, and two stocks (companies) i, j. Given that stock (company) j is a supplier and funder of stock (company) i, we can encode their relations as a multi-hot vector: $a_{ij} = a_{ji} = [1, 1, 0]$. In the ranking-based stock prediction mechanism, the higher score of the stock in the ranking list, the higher return investment profit would obtain in the time-step $t+1$.

B. Relation-temporal Graph

In this section, we describe the relation-temporal graph definition, denoted as G_{RT} hereinafter. In our work, we utilize G_{RT} to form a hierarchical representation of a bound of stocks and their daily features. The relational temporal graph can be vividly illustrated in Figure 2. G_{RT} is formulated on a

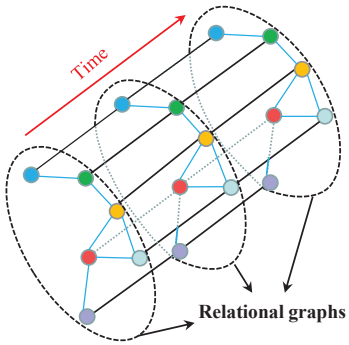


Fig. 2. Illustration of relation-temporal graph where RT-GCN operates on. This relational temporal graph consists of three relational graphs connected by temporal edges (denoted as solid black lines). In each relational graph, the stocks are connected by relational edges (denoted as solid blue lines).

sequence of relational graphs, denoted as G_R, $G_R \in G_{RT}$, where each node in a relational graph corresponds to a stock. Two types of edges exist in this graph, namely relational edges and temporal edges. Relational edges connect the stocks based on the prior knowledge obtained from Wikipedia and the NASDAQ website, denoted as solid blue lines. The temporal edges connect the same stocks across consecutive time-steps, denoted as solid black lines. The numbers of nodes and edges are fixed, i.e., no nodes or edges are dynamically added during the training and testing. To better understand the relational temporal graph, we can treat it as a cylinder. Each relational graph can be seen as a plane of the cylinder, while these planes form the cylinder by temporal edge connections. Each plane is connected by multiple points (stocks) via the relational edges.

C. Graph Convolution

Because the traditional convolution operation is not applicable to general graphs due to the non-Euclidean structure of the graph [25]. Two practical approaches have been explored currently and successfully apply the convolution operation on the graph. One is to expand the spatial definition of spatial graph convolution [26], named spatial graph convolution, which rearranges the nodes into certain grid forms that normal convolutional operations can process. The other is to operate on the spectral domain with graph Fourier transforms [27]. Compared with the former approach, the latter one is applied more widely because the reliable theory foundation support and promising progress has been made to reduce the computational complexity from $O(n^2)$ to $O(n)$ [28], [29].

Typical graph convolution can be described as Eq. (1) proposed by [28] as it simplifies the parameters of older version graph convolutional operation [29] by only considering 1-st Chebyshev polynomials:

$$g_\theta * x \approx \theta(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x, \tag{1}$$

where g_θ is the graph convolution operator. By stacking multiple convolutional layers as Eq. (1) we can build a deep graph neural network. However, too many convolutional layers can lead to exploding/vanishing gradients [28]. To alleviate this problem, Kipf and Welling [28] introduce the following normalization trick: $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, with $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. It is worth noting that the above definition is specific for a single feature x and single filter θ. This definition can be generalized to F filters and a graph with C-dimensional features for each node, as Eq. (2) describes:

$$Z = f(X, A) = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta, \tag{2}$$

where $X \in \mathbb{R}^{N\times C}$, and $\Theta \in \mathbb{R}^{C\times F}$. The convolutional definition in Eq. (2) is what we finally adopt in our work.

IV. Our Proposed Model

In this section, we elaborate on the architecture of our proposed framework. An illustration of our model can be found in Figure 3. Given a bound of stocks with their sequential historical data, we construct G_{RT} with the stocks as graph nodes, external extracted relations and consecutive time connection as graph edges. The input to the RT-GCN model is the historical data of each stock. Multiple layers of relational temporal graph convolution operations will be applied to the input data and generate higher-level feature maps on each node of G_{RT}. Each relation-temporal graph convolution layer consists of a relation convolution structure and a temporal convolution structure, and the input data will first go through the relational convolution and then the temporal convolution. The relation convolution structure aims to enrich the representation of each node by aggregating its neighbor information and temporal convolution structure to compress temporal dimension by extracting crucial information across consecutive time-steps. The average pooling layer is added after the RT-GCN layer to generate the representation for each stock. The representation will be fed to a fully connected layer to calculate the ranking score. The whole model is trained in an end-to-end manner with back-propagation. We will go over each module in detail in the following.

A. Relation-temporal Graph Construction

In our work, we utilize G_{RT} to form hierarchical representation of a bound of stocks and their daily features. More specifically, we construct an undirected relation-temporal graph $G_{RT} = (V, E)$ with N stocks and T time-steps featuring both intra-stock and inter-time connections. The node set of G_{RT} can be described as $V = \{v_{ti}|t = 1, ..., T, i = 1, ..., N\}$. The edge set E is composed of two subsets. The first subset depicts the intra-stocks connection at each time-step, denoted as $E_S = \{v_{ti}v_{tj}|(i, j) \in G_R\}$. The second subset contains the inter-time edges, which connect the same stocks in consecutive time-steps as $E_T = \{v_{ti}v_{(t+1)i}\}$.

G_{RT} is constructed in two steps. First, the stocks within one time-step are connected with relational edges according to the prior knowledge extracted from external resources, e.g., Wikipedia and sector-industry relations from the NASDAQ website[1]. Wikipedia is a free and open knowledge base that

[1]https://www.nasdaq.com/screening/industries.aspx

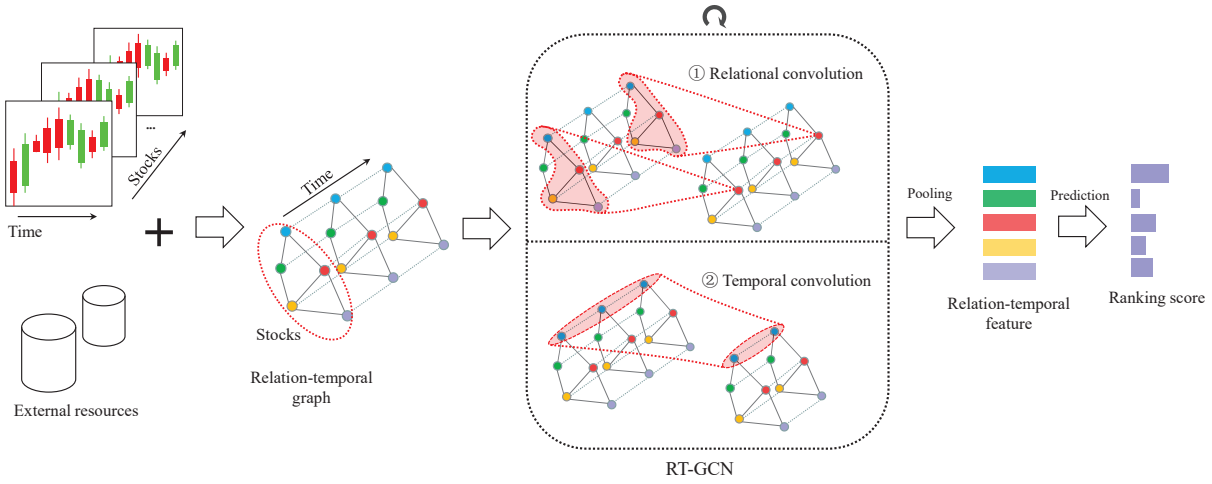


Fig. 3. Illustration of our proposed framework. The cycle sign above the RT-GCN module represents that the RT-GCN network can be stacked to become a deeper network.

contains a large number of facts about the relationship among companies (stocks) in a triple format. We connect the stocks according to the facts in Wikipedia. Studies [9], [10] show that cooperating with stock correlations can have significant improvement and stocks belonging to a common industry experience synchronous price movement trends. Following the previous works [9], [10], we also collect the sector-industry information from the NASDAQ website and connect the stocks that belong to the same industry. Next, to model the temporal feature evolution of each stock, every same stock in different time-steps will be connected, as the solid black lines shown in Figure 2. In this case, the same stock of different time-steps can establish connections and its features evolution can be captured by the neural network. As the input of RT-GCN, the feature vector on a node is flexible which can be the combination of open price, closing price, highest price, or n-day moving average or other useful features.

B. Relational Graph Convolution

We have defined the primary graph convolution in Section III-C. In this section, we will elaborate how to apply GCN to our relation-temporal graph.

As we mentioned above, we encode the relations between two stocks into a multi-hot vector. How to effectively combine the vector between the stock pair into GCN is the key of our model. We propose three different strategies to make full use of these relationships between stocks. These three strategies are achieved by different relation-aware function $\mathcal{R}$, which can return different weighted adjacency matrix A with the relational matrix $\mathcal{A}$ as input. In the following, we detail how we design these three strategies.

1) Uniform Strategy: The first strategy is the simplest one, which can be expressed as Eq. (3). It ignores the different relations between stocks and treats the different relations between all stock pairs as the same, for these different relationships play a familiar role of propagating information in the graph.

$$A_{ij} = \mathcal{R}(\mathcal{A}) = \begin{cases} 1 & if \quad sum(\mathcal{A}_{ij}) > 0 \\ 0 & if \quad sum(\mathcal{A}_{ij}) = 0 \end{cases}. \tag{3}$$

Recall that $\mathcal{A} \in \mathbb{R}^{N \times N \times K}$ is the relation matrix defined in Section III-A, and $\mathcal{A}_{ij} \in \mathbb{R}^{K}$, $A \in \mathbb{R}^{N \times N}$. With the defined adjacency matrix A, the GCN model can operate on each G_R. Note that all $G_R \in G_{RT}$ at different time-steps share the same A during the convolution process.

2) Weight Strategy: However, the first strategy is crude as it neglects that different relations may result in varying impacts between two stocks. To overcome this drawback, we apply a non-uniform coefficient when propagating the information on the graph as Eq. (4):

$$A_{ij} = \mathcal{R}(\mathcal{A}) = \mathcal{A}_{ij}^{T} w + b, \tag{4}$$

where $w \in \mathbb{R}^{K}$ and b are parameters to be learned. As strategy 1, all $G_R \in G_{RT}$ at different time-steps also share the same adjacency matrix A. This strategy is more relational-wise as it can return different weights with different relational vectors as input.

3) Time-sensitive Strategy: A common limitation for the above two strategies is that all relation graphs at different time-steps share the same weight. However, the stock market is highly dynamic such that a stock's status and the strength of a relation may continuously evolve at different time-steps. A fixed weight for all relational graphs at different time-steps will undoubtedly limit our model capability. For instance, in the previous example of Figure 1(b), the Apple Inc. stock price has a more considerable impact on the Lens stock in the period of releasing a new version of product than usual. To address this

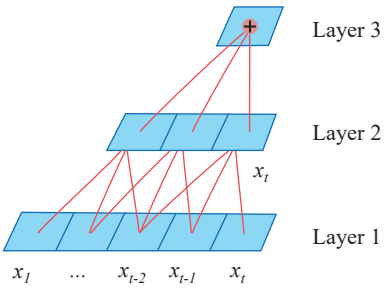


Fig. 4. Illustration of temporal convolution.

limitation, we define the time-sensitive propagation process as Eq. (5), which consider the temporal information:

$$A(t)_{ij} = \mathcal{R}(X, \mathcal{A}) = \underbrace{\frac{X(t)_i^T X(t)_j}{\sqrt{n}}}_{time-correlation} \times \underbrace{\mathcal{A}_{ij}^T w + b}_{relation-importance} . \tag{5}$$

Here, $A = \left[A(1), A(2) \dots A(T)\right] \in \mathbb{R}^{T \times N \times N}$, $X(t)_i, X(t)_j$ are the features of stocks i and j at time-step t, and n is the dimension of $X(t)_i$. $w \in \mathbb{R}^K$ and b are trainable parameters. The relation-aware function in this strategy is determined by two terms: time-correlation and relation-importance. Specifically, the first term measures the correlation between two stocks by using their features at the current time-step. We adopt the scaled dot-product [30] to calculate this term inspired by [31], which utilizes it to calculate the weight of two nodes in a graph. The second term is the same as Eq. (4) to return the relation weight for a relational vector. Different from the above two strategies, this strategy can assign a unique weighted adjacency matrix for every G_R in G_{RT}. It is a superior-wise strategy and more suitable for stock prediction than the other two strategies as it considers both the temporal correlation and relation importance between two stocks.

C. Temporal Convolution

We build up a temporal convolution network (TCN) module by referring the temporal convolution architecture proposed in [23], [32]. TCN uses a 1-D filter to capture temporal dependency by using causal convolutions [33], as Figure 4 displays, where an output at time t is convolved only with elements from time t and earlier in the previous layer, so that there can be no leakage from the future to the past. For example, x_t in layer 2 is only convolved with x_{t-2}, x_{t-1} and x_t in layer 1, and does not depend on any of the future at time-steps $t+1$ and $t+2$ in layer 1. In this way, there can be no leakage from the future to the past. To capture longer distance dependency of the input feature, we change the filter moving strides to expand the receptive field corporation with zero padding [34]. Eq. (6) gives the mathematical expression of TCN on the $t-th$ element X_t operated by a filter $\mathcal{F} \in \mathbb{R}^k$:

$$\mathcal{O}(X_t) = (\mathcal{F} * X) = \sum_{i=0}^{k-1} \mathcal{F}(i) * x_{t-k+i}. \tag{6}$$

We also add the residual connection [35] after the TCN layer to ease network training and apply weight normalization [36] to convolution filters. In addition, a spatial dropout [37] was added after each TCN layer for regularization.

D. Pooling and Prediction

After the RT-GCN processing, we can obtain the relation-temporal feature $Z \in \mathbb{R}^{H \times N \times F}$ for our relation-temporal graph, H is the final temporal dimension after processing by the temporal convolution component, N is the number of stocks and F is the feature dimension of stock after processing by the relational convolution component. In the following we use an average pooling layer with stride=H to generate the representation $Z' \in \mathbb{R}^{N \times F}$ for each stock. Finally, the representation of each stock would be fed into a fully connected layer to calculate the ranking score $\hat{r}_{t+1}$. We optimize our model using a mean squared error loss τ_{reg} to minimize the difference between the predicted and actual return ratios cooperated with pairwise ranking-aware loss τ_{rank} to maintain the relative order of the ranked stocks.

$$\tau_{reg} = ||\hat{r}_{t+1} - r_{t+1}||^2, \tag{7}$$

$$\tau_{rank} = \sum_{i=0}^{N} \sum_{j=0}^{N} ReLU(-(\hat{r}_i^{t+1} - \hat{r}_j^{t+1})(r_i^{t+1} - r_j^{t+1})), \tag{8}$$

where r_{t+1} is the ground-truth, $\hat{r}_{t+1}$ is the predicted ranking score, and $ReLU$ is the activation function. Thus, the final loss function for our model can be written as Eq. (9):

$$\tau = \tau_{reg} + \alpha \tau_{rank} + \lambda ||\beta||^2, \tag{9}$$

where β is the set of all learnable parameters, λ is a weight for $L2$ regularization and α is the balancing parameter between the regression loss and ranking loss.

V. Experiments

A. Data Collection and Processing

We conduct our experiments on three stock markets NASDAQ Stock Exchange (NASDAQ), New York Stock Exchange (NYSE) and Chinese A-share market. For the NASDAQ and NYSE datasets, we use the stock lists collected by [9]. For Chinese A-share market, we collect the constituent stocks from China Securities Index 300 (CSI). However, some stocks have been delisted because of the fierce competition in the market. Therefore, we only leave the stocks that only have transaction records between 02/01/2015 and 12/31/2020, obtaining 854, 1,405 and 242 stocks for NASDAQ, NYSE and CSI respectively. For these stocks, we collect two kinds of data: (1) Historical closing price data. Following [2], [9], we only use the closing price and its n-day moving averages as features. (2) Relational data, including the industry relations such as biotechnology, computer software, nursing service and wiki relations between these companies such as supplier-consumer relation and funder-by relation.

TABLE II
STATISTICS OF HISTORICAL DATA.

Market	Stocks	Training date	Training days	Testing date	Testing days
NASDAQ	854	2015/01-02 – 2020/02/28	1295	2020/03/02 – 2020/12/31	207
NYSE	1405	2015/01-02 – 2020/02/28	1295	2020/03/02 – 2020/12/31	207
CSI	242	2015/01-02 – 2020/04/30	1295	2015/05-01 – 2020/12/31	139

1) Stock Historical Closing Price: We collect the historical closing price data of stocks by using a publicly available API yahoo finance[2]. After finishing the collection, we further process these sequential data according to the following four steps. Step 1: Normalized closing price. We follow the previous works [38], [39] that normalize the closing price via dividing by its closing price at the last period of the input time sequence to avoid the leakage of future information, i.e., the closing price at t-th period is normalized by $p^t = \frac{p_t}{p^T}$, where p^T is the closing price at the last period of the input time sequence. Step 2: Calculating more features. Besides the closing price, we calculate the 5, 10, 20 days moving averages of closing price which represent the weekly and half-month trends of each stock. Step 3: Calculating return ratio. According to the definition of ranking-based stock prediction in Section III-A, our purpose is to predict the ranking list of stocks at $T+1$ trading day based on the feature of the previous T trading days, so we need to have a metric to evaluate the stock revenue. Following [9], we set return ratio of a stock as ground-truth because it indicates the expected revenue of a stock, and the higher return ratio indicates higher revenue compared with the previous trading day. The return ratio of the stock i at trading day t can be calculated as Eq. (10):

$$r_i^{t+1} = (p_i^{t+1} - p_i^t)/p_i^t, \tag{10}$$

where p_i^t is the closing price at day t. Step 4: Separating sequential data. For all datasets, we chronologically split the sequential price data into two time periods for training and testing, respectively. The data statistics are summarized as Table II displays.

2) Stock Relational Data: As Figure 1(a) shows, the stocks under the same industry could perform similar trend. For example, the stocks in medical sector have increased rapidly since the COVID-19 outbreak. To capture such an implicit pattern, we filter the industry relations of the available stocks from the dataset provided in [9] and set them as the kinds of relational edges for G_{RT}. If two stocks are under the same industry, we regard this industry as a relation between these two stocks. For instance, Facebook Inc. and Twitter Inc. are under the same industry name Technology Services-Internet Software/Services, which can be represented as (Facebook; Technology Services:Internet Software,Services; Twitter).

We also enrich the relations for G_{RT} by setting another relation infomation, wiki company-based relations, as relational edges. Wikipedia [40] is one of the biggest and most active open domain knowledge bases with rich sources of entity relations. It contains lots of company entities and company relations, which might reflect the impact across stocks. Wiki relations can be queried by using public API wikimedia[3]. There may be multiple relations between two stocks, e.g., GOOGLE Inc. and ALPHABET Inc. have two relations (ALPHABET; Technology Services:Internet Software; GOOGLE) and (ALPHABET; own by GOOGLE), and we encode these multiple relations between two stocks as a multi-hot vector. We summarize the number of relation types and the relation ratio of stock pairs in Table III. Compared with the other two datasets, CSI has no wiki relations since most Chinese companies are not recorded on Wikipedia, and the relationships found among these Chinese companies are incomplete. It should be noted that the relations are collected before the test period to prevent the leakage of future information.

TABLE III
STATISTICS OF WIKI RELATION AND INDUSTRY RELATION DATA.

Market	Wiki-relation		Industry-relation	
	Types	Relation ratio	Types	Relation ratio
NASDAQ	41	0.3%	97	5.4%
NYSE	28	0.4%	108	6.9%
CSI	-	-	24	6.7%

B. Experimental Setup

1) Evaluation Protocols: Following [9], [42], we adopt a daily buy-sell trading strategy to evaluate our model performance on stock ranking regarding the revenue. For the fairness of the experiments, we obey all the trading assumptions proposed in [9], [10], i.e., buying the top-N stocks at trading day t and selling all these stocks at day $t+1$. Our work focuses on selecting the stocks with the highest return ratio accurately and does not consider the stocks' shares to buy or the amount of capital invested.

2) Evaluation Baselines: The baselines compated with our model can be divided into classification-based (CLF), regression-based (REG), reinforcement learning-based (RL) and ranking-based (RAN) methods. To ensure the significance of the experimental results, all the methods use the same features to predict the day-level return ratio with the same length of window size T. RT-GAT is implemented by replacing the relational graph convolution (Section IV-B) with a graph attention network [31]. We construct the graph for RT-GAT

[2]https://github.com/ranaroussi/yfinance

[3]https://www.mediawiki.org/wiki/Wikibase/DataModel/JSON

TABLE IV
PERFORMANCE COMPARISON OF ALL BASELINES. '-' MEANS THAT THE METRIC CANNOT BE CALCULATED BECAUSE THE STOCK PROFIT RANKING CANNOT BE CALCULATED.

	Models	NASDAQ				NYSE				CSI			
		MRR	IRR-1	IRR-5	IRR-10	MRR	IRR-1	IRR-5	IRR-10	MRR	IRR-1	IRR-5	IRR-10
CLF	ARIMA [14]	-	0.12	0.20	0.23	-	0.10	0.17	0.28	-	-0.13	-0.09	0.07
	A-LSTM [41]	-	0.23	0.40	0.59	-	0.43	0.51	0.69	-	-0.07	0.15	0.11
REG	SFM [1]	0.032	0.11	0.28	0.36	0.029	0.13	0.23	0.31	0.007	-0.21	0.10	0.17
	LSTM [16]	0.027	0.13	0.22	0.28	0.013	0.09	0.15	0.19	0.011	-0.33	0.07	0.12
RL	DQN [18]	0.041	0.20	0.34	0.58	0.033	0.12	0.33	0.60	0.009	0.11	0.13	0.15
	iRDPG [19]	0.049	0.29	0.37	0.46	0.040	0.20	0.28	0.57	0.010	0.12	0.14	0.15
RAN	Rank_LSTM [16]	0.041	0.35	0.42	0.82	0.029	0.30	0.64	0.63	0.016	-0.24	0.14	0.19
	RSR_I [9]	0.043	0.81	0.93	0.88	0.032	0.66	0.98	1.01	0.015	-0.04	0.29	0.33
	RSR_E [9]	0.055	0.89	0.83	0.84	0.048	0.88	0.92	0.95	0.021	0.29	0.26	0.28
	RT-GAT [31]	0.049	0.11	0.67	0.87	0.044	0.48	0.87	0.97	0.027	0.27	0.30	0.25
Ours	RT-GCN (U)	0.055	0.45	0.71	0.82	0.040	0.65	0.88	0.96	0.017	0.13	0.15	0.19
	RT-GCN (W)	0.059	0.63	0.86	0.98	0.047	0.80	0.95	0.99	0.022	0.29	0.26	0.35
	RT-GCN (T)	**0.061**	**1.25**	**0.97**	**1.03**	**0.056**	**0.92**	**1.10**	**1.13**	**0.031**	**0.35**	**0.35**	**0.38**
Improvement		10.9%	40.4%	4.3%	17.0%	16.7%	4.5%	12.2%	11.9%	14.8%	20.7%	16.7%	15.2%
p-value		3.05e-4	9.16e-5	6.10e-5	3.05e-4	1.52e-4	4.27e-4	7.63e-4	3.05e-5	3.05e-5	1.53e-4	6.10e-5	4.82e-4

by connecting a pair of nodes (i.e., stocks) having at least one type of relations. Among these baselines, ARIMA [14], A-LSTM [41], SFM [1], LSTM [16], DQN [18], iRDPG [19] and Rank_LSTM [16] cannot utilize the relational data. RSR_I [9], RSR_E [9] and RT-GAT [31], as graph-based methods, could leverage the relational data as our model does.

3) Evaluation Metrics: The purpose of the ranking-based stock prediction task is to rank the relative order of stocks accurately. Therefore, we employ mean reciprocal rank (MRR) and cumulative investment return ratio (IRR) as two metrics to evaluate our model performance. MRR is a widely used metric for ranking tasks. We calculate the MRR result of the top-1 stock in a ranking list over the testing days. IRR is our primary metric as it directly reflects the effect of stock investment. It is calculated by summing over the return ratio of the selected top-N stocks in the ranking list on each testing day. We select the top-1, top-5 and top-10 stocks to calculate the IRR results, denoted as IRR-1, IRR-5 and IRR-10. Larger values of MRR and IRR indicate better performance.

4) Training Setup: All the experiments were conducted on a machine with 32GB of RAM and 2 NVIDIA TITAN GPUs. To eliminate the randomness and have a statistically significant result, we run all models fifteen times and average the performance. All baselines are optimized with the tuning strategy reported in their papers to obtain the best performance. We implement the proposed RT-GCN with PyTorch and also apply the Adam optimizer with a learning rate of 0.001. The same tuning strategy and grid search are employed to select the optimal hyperparameters on all graph-based methods. Specifically, we tune two hyperparameters for graph-based models, the window size T and the α value in Eq. (9), within $\{5, 10, 15, 20\}$ and $\{0.01, 0.1, 0.2\}$, respectively. λ in Eq. (9) is set to 0.01. We conduct all the following experiments with one layer of RT-GCN as too many layers could cause overfitting. The code and data to reproduce our experiments are available[4].

[4]https://github.com/zhengzetao/RTGCN

C. Experimental Results

1) Comparison with Baselines: The performance comparison is displayed in Table IV, where the model with the best performance is denoted in bold and strongest baselines are highlighted with an underline. The improvements are measured between RT-GCN (T) with the strongest baselines. The classification-based methods only output three results (up, neutral, down) but cannot rank the stocks according to the return ratio, so we randomly select top-N stocks to calculate IRR. The three different strategies of our model described in Section IV-B (uniform, weighted and time-sensitive) are denoted as RT-GCN (U), RT-GCN (W) and RT-GCN (T), respectively. We can see that, our proposed model achieves higher returns than all baselines across all datasets. Generally, RL-based and ranking-based methods are more profitable than classification and regression methods, because these two kinds of methods are inherently optimized for higher profit. This verifies our premise of formulating the stock prediction as a learning-to-rank problem and selecting the most profitable stocks to trade is right. Furthermore, RSR_E and RSR_I are better than our proposed RT-GCN (U) and RT-GCN (W) on most metrics, but this does not mean that our model is invalid. Compared with RT-GCN (U) and RT-GCN (W), the two RSR models are more delicate because they consider the relation evolution at different time-steps. It is more fair to compare RSR with our proposed RT-GCN (T), which also considers the relation evolution at different time-steps. From Table IV, we can observe that RSR_E and RSR_I perform worse than RT-GCN (T) because RT-GCN (T) utilizes scaled dot-product [30], a more effective way to calculate the correlation among stocks. RT-GAT is competitive compared with most baselines but cannot completely surpass RT-GCN (U) and RT-GCN (W). The reason is that GAT calculates the weights between stocks through the temporal features of stocks without considering the multiple relations between them, which impedes the performance of RT-GAT. Moreover, we also notice that each of

TABLE V
PERFORMANCE COMPARISON WITH STHAN-SR AND RSR ON THE NASDAQ AND NYSE DATASETS. '-' INDICATES THE RESULT CANNOT BE OBTAINED FROM THE PAPER OR REPRODUCED FROM THE SOURCE CODE.

Models	NASDAQ-II			NYSE-II		
	MRR	IRR-5	IRR-10	MRR	IRR-5	IRR-10
RSR_I [9]	0.032	0.13	0.22	0.045	0.10	0.12
RSR_E [9]	0.032	-	-	0.043	-	-
STHAN-SR [10]	-	0.44	-	-	0.33	-
RT-GCN (T)	**0.040**	**0.48**	**0.50**	**0.053**	**0.37**	**0.48**
p-value	0.002	0.008	0.001	0.003	0.001	0.0

our three designed strategies performs better than the next. Specifically, the RT-GCN (W) performs better than RT-GCN (U) in four metrics because RT-GCN (W) has considered the effect of different relationships on the propagation of information in the graph, or in other words, it gives different weights to different relationships. Furthermore, RT-GCN (T) is the best of our three strategies as it has considered the weight evolution at different time-steps.

Additionally, we use the paired Wilcoxon signed rank test [43] to verify the significance of our method surpassing the strongest baseline, and the p-values reported in Table IV reflect the significance level. The statistical significance analysis is based on 15 pairs of experimental results from RT-GCN (T) and the strongest baseline. All the results are obtained by running the model 15 times with the same parameter setting. The confidence level of 0.05 is a 'rule of thumb' which has been adopted in many studies [44], [45] as well as in our work. In other words, when the p-value is smaller than 0.05, it is considered statistically significant that our model outperforms the baseline. From Table IV, we can observe that all the p-values are less than 0.05, so it is concluded that the experimental results are of statistical significance.

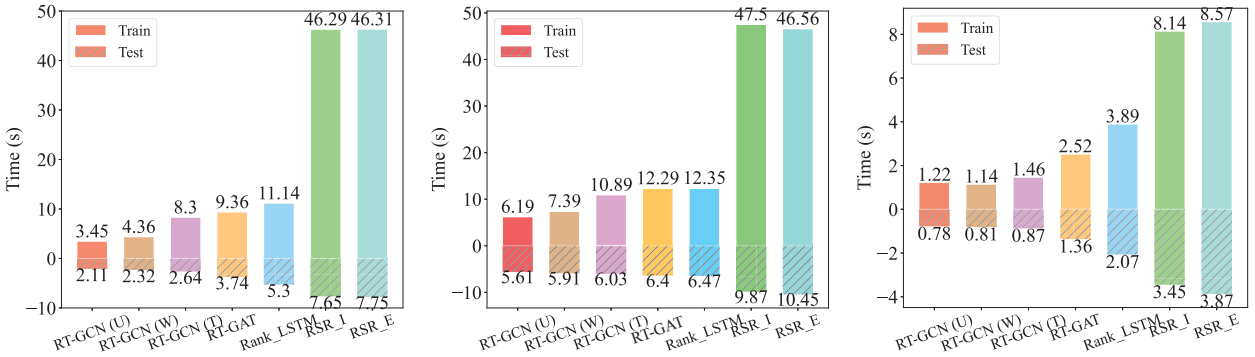
Furthermore, to better verify the superiority of our model, we evaluate our model with the same data, i.e., relational data (only industry relations) and sequential price data published in [9] (denoted as NASDAQ-II and NYSE-II to make distinction) and the same parameter setting, i.e., the same window size T and learning rate. All the results displayed in Table V are taken directly from [9], [10]. The best performance is denoted in bold and strongest baselines are highlighted with an underline. We run our model with the published dataset 15 times and obtained 15 results, and then we calculate the statistical significance of our results by utilizing one-sample Wilcoxon signed rank test [43], which can calculate the statistical significance of our 15 results being greater than the specific value (i.e., the result of the baseline). The results in Table V mean that our model statistically outperforms RSRs and STHAN-SR (all p-values smaller than 0.05), which further indicates that modeling based on a relation-temporal graph is more advanced than processing in a two-step format adopted in RSRs [9] and STHAN-SR [10].

2) Speed Comparison: Besides the excellent performance in the above evaluation metrics, our model performs excellently in terms of model efficiency. As shown in Figure 5, we have the following observation (for a fair comparison, we only compare with the ranking-based methods because they need to rank and consider the relationship of the stocks): the training time and testing time are significantly reduced compared with other baselines, no matter on which dataset. Specifically, on the NASDAQ dataset, our model achieves up to 3.2× and 13.4× faster than Rank_LSTM and RSR in training speed, respectively. Regarding testing speed, our model achieves up to 2.5× and 3.6× faster than Rank_LSTM and RSR. RT-GAT is close to RT-GCN (T) in both training speed and testing speed, and is faster than Rank_LSTM and RSR, which further demonstrates that modeling the temporal features and relations of stocks into a relation-temporal graph is efficient. On the other two datasets, the training speed and testing speed of our model are also greatly reduced. Due to space constraints, we do not analyze them in detail. The reason why our model performs more efficiently than Rank_LSTM is that, our model is implemented only by a simple convolution operation without requiring other modules to extract the features. The reason why our model is more efficient than RSR is that the RSR model has to wait for the LSTM model to finish the calculation of the temporal features before calculating the relation dependency, which seriously slows down its efficiency.

3) Return Ratio Analysis: Figure 6 illustrates the procedure of return ratio change during the test period. As can be seen, IRR-1 suffers severe fluctuations compared with IRR-5 and IRR-10 (e.g., in Figure 6(b), IRR-1 rises rapidly on the 30-th trading day, and drops rapidly on the 160-th trading day). The result of IRR-1 indicates that selecting only one stock from hundreds of stocks is a highly risk operation. Once the ranking order is accurate, buying and selling stocks would achieve a higher return ratio, and vice versa. Compared with IRR-1, IRR-5 and IRR-10 rise smoothly. The reason is that IRR-5 and IRR-10 consider multiple stocks and the investments are well diversified to reduce the risk.

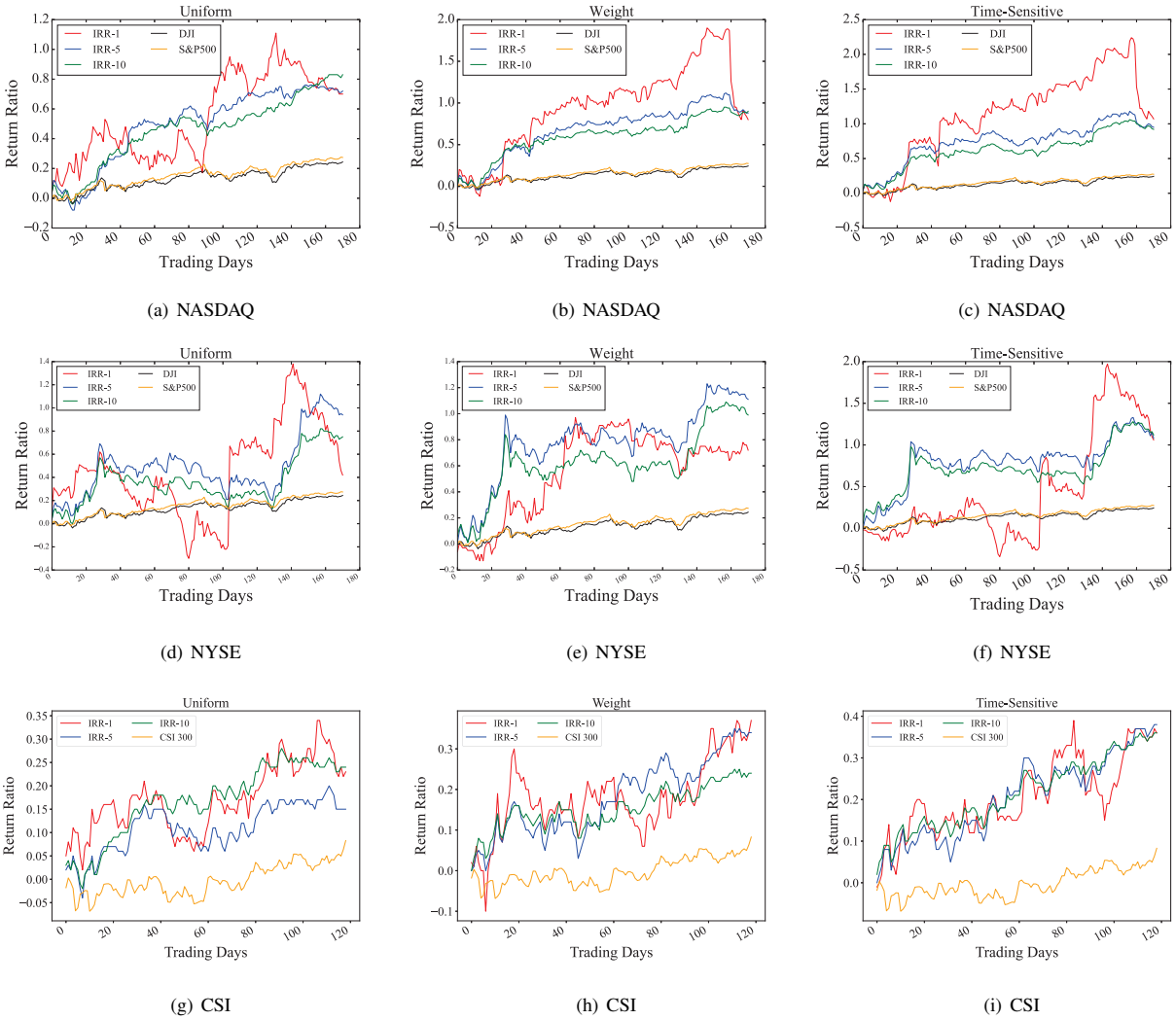
To show the effectiveness of our model, we also compare with Dow Jones Industrial Average Index (DJI) and S&P 500 for the NASDAQ and NYSE datasets, and compare with China Securities Index 300 (CSI 300) for the CSI dataset. S&P 500, DJI and CSI 300 are three popular market indices that investors concern with. We collect the values of S&P 500[5], DJI[6] and CSI 300[7] during the test period from public financial websites. The results are shown in Figure 6. We can observe that the return ratios obtained by the three strategies of our model are all higher than the three market indices. Generally speaking, when the return ratio of the model is greater than the market index, it means that the model is useful [9]. This also justifies the competitiveness of our model.

[5]https://www.macrotrends.net/2526/sp-500-historical-annual-returns
[6]https://dqydj.com/dow-jones-return-calculator
[7]https://hk.investing.com/indices/csi300-historical-data



(a) NASDAQ (b) NYSE (c) CSI

Fig. 5. Training and testing speed comparison of different models. The shaded part is the testing speed.



(a) NASDAQ (b) NASDAQ (c) NASDAQ

(d) NYSE (e) NYSE (f) NYSE

(g) CSI (h) CSI (i) CSI

Fig. 6. Performance comparison of three strategies regarding three metrics (IRR-1, IRR-5, IRR-10) and three market indexes (DJI, S&P 500 and CSI 300).

TABLE VI
PERFORMANCE COMPARISON BETWEEN WIKI RELATIONS AND INDUSTRY RELATIONS ON THE NASDAQ AND NYSE DATASETS.

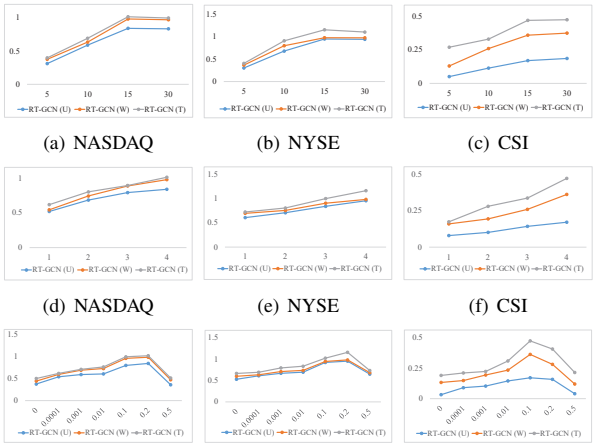
Models	Wiki-relation				Industry-relation			
	MRR	IRR-1	IRR-5	IRR-10	MRR	IRR-1	IRR-5	IRR-10
	NASDAQ							
Rank_LSTM	0.040	0.36	0.44	0.83	0.040	0.36	0.44	0.83
RT-GCN (U)	0.037	0.21	0.58	0.62	0.051	0.33	0.77	0.79
RT-GCN (W)	0.036	0.43	0.66	0.71	0.053	0.35	0.79	0.87
RT-GCN (T)	**0.049**	**0.70**	**0.90**	**0.97**	**0.055**	**0.41**	**0.91**	**0.88**
	NYSE							
Rank_LSTM	0.040	0.36	0.44	0.83	0.040	0.36	0.44	0.83
RT-GCN (U)	0.037	0.21	0.58	0.62	0.051	0.33	0.77	0.79
RT-GCN (W)	0.036	0.43	0.66	0.71	0.053	0.35	0.79	0.87
RT-GCN (T)	**0.049**	**0.70**	**0.90**	**0.97**	**0.055**	**0.41**	**0.91**	**0.88**

D. Ablation Study

In this section, we study (1) whether different relational graphs will affect the performance of the model, which is beneficial for constructing a more effective relational graph in the future; (2) the impact of relational convolution module and temporal convolution module on RT-GCN. We detail these two ablation studies in the following.

1) Impact of Relation Types: Performance comparison between wiki relations and industry relations on the NASDAQ and NYSE datasets can be seen in Table VI, from which we have the following observations. (1) The proposed RT-GCN with three strategies performs better than Rank_LSTM on two kinds of relational data. This result verifies the effectiveness of modeling relational data compared with the model that only considers temporal features. Moreover, (2) RT-GCN with industry relation performs better on two datasets than RT-GCN with wiki relations. It could be attributed that the relation ratios (e.g., 5.4% in NASDAQ and 6.9% in NYSE, respectively) of industry relations are larger than that (e.g., 0.3% in NASDAQ and 0.4% in NYSE, respectively) of wiki relations. The larger the relation ratio, the wider the information can be propagated on the graph. Consequently, it also suggests the worth of completing the relation between stocks, making the connection between stocks close to the real world. (3) On the NASDAQ dataset, RT-GCN (T) with wiki relations performs better than RT-GCN (T) with industry relations in the IRR-1 and IRR-10 metrics, which is contrary to observation (2). The lower performance of IRR-1 indicates that selecting only one stock from hundreds of stocks is a highly risk operation. Once the ranking order is inaccurate, the investment return ratio would suffer severe fluctuations, and the reason for the lower IRR-10 is that NASDAQ is tech-dominated and most stocks in it are connected by the industry relations. These stock prices actually decreased due to a sharp drop during the test period from March 2020 to April 2020, resulting in less number of tech stocks selected in the top 10, and thus lower IRR-10.

2) Impact of Relation Graph Convolution and Temporal Convolution: We conduct another experiment to explore the role of relational convolution module or temporal convolution module in our model. In this experiment, we build another two models, R-Conv and T-Conv, by removing temporal convolution module or relational convolution module, respectively.



(a) NASDAQ (b) NYSE (c) CSI

(d) NASDAQ (e) NYSE (f) CSI

(g) NASDAQ (h) NYSE (i) CSI

Fig. 7. Performance comparison of our model with different window sizes (a)-(c), number of features (d)-(f) and α values (g)-(i) when training. The x-axis represents the value of hyperparameter, and the y-axis represents the value of IRR.

R-Conv removes the temporal convolution module and only leaves the relational convolution module with our proposed uniform strategy, which aims at testing the performance of our model without temporal convolution. T-Conv removes the relational convolution module and leaves the temporal module to test the performance of our model without relational convolution. Table VII shows the performance of R-Conv and T-Conv compared with RT-GCN (U). We can have the following observation from the Table VII: the performance of R-Conv is the worst while T-Conv is better than R-Conv. However, both of these two models are outperformed by RT-GCN (U). R-Conv has considered the relational dependency among stocks, but stock prediction is a task that depends more on the effectiveness of temporal features. That is the reason why T-Conv performs better than R-Conv. Compared with T-Conv, RT-GCN (U) not only extracts the temporal dependency through temporal convolution but enrichs the features of each stock by aggregating the information from neighbors. That is why RT-GCN (U) performs better than T-Conv.

E. Hyperparameter Analysis

We aim at understanding how RT-GCN performs by varying the values of different hyperparameters, including the training window size, the number of features used in the model, and the balancing parameter that determines the contributions of two different losses. All the following comparison experiments are conducted following the same rule: keeping the other hyperparameters fixed when varying any of these hyperparameters. The following experiments are only conducted on the time-sensitive strategy due to its excellent performance.

1) Training Window Size (Time Span): Figure 7(a)-(c) show the performance is affected by changing window size (i.e., the length of time sequence used for prediction, also known as time span). We can find that using about 15 days of sequence

TABLE VII
PERFORMANCE COMPARISON OF R-CONV, T-CONV AND RT-GCN (U).

Market	NASDAQ				NYSE				CSI			
	MRR	IRR-1	IRR-5	IRR-10	MRR	IRR-1	IRR-5	IRR-10	MRR	IRR-1	IRR-5	IRR-10
RT-GCN (U)	**0.049**	**0.70**	**0.68**	**0.82**	**0.033**	**0.38**	**0.97**	**0.73**	**0.017**	**0.13**	**0.14**	**0.17**
R-Conv	0.024	0.35	0.37	0.58	0.016	0.20	0.53	0.52	0.009	0.06	0.08	0.10
T-Conv	0.033	0.54	0.49	0.70	0.021	0.31	0.71	0.68	0.013	0.10	0.12	0.15

TABLE VIII
PERFORMANCE COMPARISON BETWEEN WIKI RELATIONS AND INDUSTRY RELATIONS ON THREE DATASETS.

Num.	Feature combination
1	closing price
2	closing price, 5 days moving average
3	closing price, 5 days, 10 days moving averages
4	closing price, 5 days, 10 days, 20 days moving averages

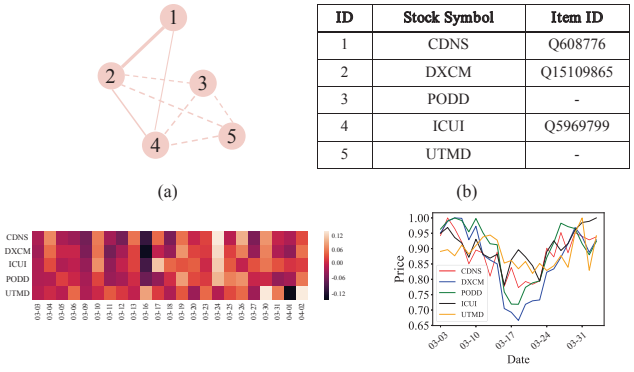
length for training leads to higher IRR. A small number days (i.e., 5 or 10) for training can result in worse performance. There is no significant change when the window size is longer than 15 days.

2) Feature Number: The results exhibited in Figure 7(d)-(f) show that the feature number of 4 leads to better performance. We use the closing price, 5 days moving average, 10 days moving average, and 20 days moving average as our candidate features. The feature combination is shown in Table VIII. The model performs better as the number of features increases. The possible reason could be that more features can result in the better fitting. Hence, we would suggest using the 4 as the feature number in RT-GCN. The more valuable features will be explored in the future and applied to RT-GCN.

3) Balancing Parameter α: We present the performance of $\alpha \in \{0, 0.0001, 0.001, 0.01, 0.1, 0.2, 0.5\}$ in Figure 7(g)-(i). It can be found that α = 0.2 leads to the best performance for NASDAQ and NYSE datasets and α = 0.1 results in best performance for CSI dataset. A small α indicates the model emphasizes more on regression loss than ranking loss when optimizing. We can also find that when we consider only the regression loss (i.e., α = 0) or give too much weight to the ranking loss (i.e., α = 0.5), the model performs worse. The results imply that properly fusing the regression loss and ranking loss can perform better. We suggest setting α = 0.1 or 0.2 when applying to different datasets based on the results.

F. Case Study

In this section, we qualitatively analyze the performance of RT-GCN (T) by visualizing the prediction of five stocks in the NASDAQ test set from March 3 to April 3. From Figure 8, we can observe that RTGCN (T) can excellently predict the return ratio in a temporal dimension. Specifically, on March 4, our model predicts that the return ratio of the stocks will rise and actually they are (the color becomes lighter on March 4). The prediction on March 16 is also accurate, but we do not analyze it due to space limit. We observe that our model can also capture the relation dependency on the spatial



ID	Stock Symbol	Item ID
1	CDNS	Q608776
2	DXCM	Q15109865
3	PODD	-
4	ICUI	Q5969799
5	UTMD	-

Fig. 8. An example of the daily return ratio prediction of our model. (a) is part of the relational graph about the five stocks. The edge width indicates the weight (learned by our model) between two stocks, and the larger weight the wider edge. The solid edges indicate the wiki relations, and the dotted edges indicate the industry relations. (b) gives the details of the stocks in (a). Item Id is the entity id of the stock in the Wikipedia database. (c) is the heatmap of the predicted return ratio of the five stocks in (a). (d) is the ground-truth price (normalized) curve of the stocks in (a).

dimension. For example, the prediction of CDNS, CXDM and ICUI perform similarly (better viewed from the color of the heatmap) because the three stocks are more closely connected (better viewed from the width of the edges in (a)).

VI. CONCLUSION

We propose a relation-temporal graph convolutional network (RT-GCN) for ranking-based stock prediction to fully utilize the relations among stocks and achieve the highest revenue from stock investment. We first model the relations among stocks and their daily features into a relation-temporal graph. Then, we apply the graph convolutional network and three proposed relation-aware strategies to extract the relation-temporal feature for each node (stock). Finally, the features are fed to calculate the ranking score in a learning-to-rank way, and the stock with highest score represents highest investment revenue in the future. Experimental results on NASDAQ, NYSE and CSI demonstrate the effectiveness and efficiency of our proposed model.

This work focuses on building relations among stocks, modeling their dependency and studying how one stock affects others. Once the model can capture the dependency among stocks, external information such as news and tweets can enrich the features and predict stock trends more accurately, which could be our future work.

REFERENCES

[1] L. Zhang, C. C. Aggarwal, and G. 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, Halifax, NS, Canada, August 13 - 17, 2017*, 2017, pp. 2141–2149.

[2] G. Liu, Y. Mao, Q. Sun, H. Huang, W. Gao, X. Li, J. Shen, R. Li, and X. Wang, "Multi-scale two-way deep neural network for stock trend prediction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 2020, pp. 4555–4561.

[3] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, pp. 785–794.

[4] L. Medsker and L. C. Jain, Eds., *Recurrent Neural Networks: Design and Applications*. CRC Press, 1999.

[5] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 2017, pp. 2627–2633.

[6] S. Deng, N. Zhang, W. Zhang, J. Chen, J. Z. Pan, and H. Chen, "Knowledge-driven stock trend prediction and explanation via temporal convolutional network," in *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 2019, pp. 678–685.

[7] Q. Li, J. Tan, J. Wang, and H. 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, 2021.

[8] W. Li, R. Bao, K. Harimoto, D. Chen, J. Xu, and Q. Su, "Modeling the stock relation with graph network for overnight stock movement prediction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 2020, pp. 4541–4547.

[9] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T. Chua, "Temporal relational ranking for stock prediction," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, pp. 27:1–27:30, 2019.

[10] R. Sawhney, S. Agarwal, A. Wadhwa, T. Derr, and R. R. Shah, "Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 2021, pp. 497–504.

[11] Y. Hsu, Y. Tsai, and C. Li, "Fingat: Financial graph attention networks for recommending top-k profitable stocks," *IEEE Trans. Knowl. Data Eng.*, 2021. [Online]. Available: https://doi.org/10.1109/TKDE.2021.3079496

[12] M. C. S. Carreira, "Exponential kernels with latency in hawkes processes: Applications in finance," *CoRR*, vol. abs/2101.06348, 2021.

[13] J. J. Murphy, *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. Penguin, 2019.

[14] J. Wang and J. Leu, "Stock market trend prediction using arima-based neural networks," in *Proceedings of International Conference on Neural Networks (ICNN'96), Washington, DC, USA, June 3-6, 1996*, 1996, pp. 2160–2165.

[15] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 3, pp. 653–664, 2017.

[16] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PloS one*, vol. 12, no. 7, p. e0180944, 2017.

[17] A. Popoola and K. Ahmad, "Testing the suitability of wavelet preprocessing for TSK fuzzy models," in *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2006, Vancouver, BC, Canada, July 16-21, 2006*, 2006, pp. 1305–1309.

[18] S. Carta, A. Ferreira, A. S. Podda, D. R. Recupero, and A. Sanna, "Multi-dqn: An ensemble of deep q-learning agents for stock market forecasting," *Expert Syst. Appl.*, vol. 164, p. 113820, 2021.

[19] Y. Liu, Q. Liu, H. Zhao, Z. Pan, and C. Liu, "Adaptive quantitative trading: An imitative deep reinforcement learning approach," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 2020, pp. 2128–2135.

[20] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[21] S. Rhee, S. Seo, and S. Kim, "Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 2018, pp. 3527–3534.

[22] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, 2018, pp. 593–607.

[23] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 7444–7452.

[24] J. Chen, F. Lécué, J. Z. Pan, and H. Chen, "Learning from ontology streams with semantic concept drift," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 2017, pp. 957–963.

[25] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2021.

[26] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 2014–2023.

[27] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[29] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 3837–3845.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 5998–6008.

[31] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[32] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, 2016, pp. 47–54.

[33] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, 2016, p. 125.

[34] A. H. Waibel, T. Hanazawa, G. E. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 37, no. 3, pp. 328–339, 1989.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778.

[36] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016.

[37] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[38] K. Xu, Y. Zhang, D. Ye, P. Zhao, and M. Tan, "Relation-aware transformer for portfolio policy learning," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 2020, pp. 4647–4653.
[39] Y. Zhang, P. Zhao, Q. Wu, B. Li, J. Huang, and M. Tan, "Cost-sensitive portfolio selection via deep reinforcement learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 236–248, 2022.
[40] D. Vrandecic and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Commun. ACM*, vol. 57, no. 10, pp. 78–85, 2014.
[41] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T. Chua, "Enhancing stock movement prediction with adversarial training," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 2019, pp. 5843–5849.
[42] M. Dixon, D. Klabjan, and J. H. Bang, "Classification-based financial markets prediction using deep neural networks," *Algorithmic Finance*, vol. 6, no. 3-4, pp. 67–77, 2017.
[43] D. Rey and M. Neuhäuser, "Wilcoxon-signed-rank test," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Springer, 2011, pp. 1658–1659.
[44] Y. He, X. Chu, and Y. Wang, "Neighbor profile: Bagging nearest neighbors for unsupervised time series mining," in *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, 2020, pp. 373–384.
[45] O. Yürüten, J. Zhang, and P. Pu, "Decomposing activities of daily living to discover routine clusters," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, 2014, pp. 1348–1354.