Inductive Representation Learning on Dynamic Stock Co-Movement Graphs for Stock Predictions

Hu Tian
The State of Key Laboratory of Management and Control for Complex System, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China; School of Artificial Intelligence, University of Chinese Academy of Science, Beijing, China, tianhu2018@ia.ac.cn

Xiaolong Zheng
The State of Key Laboratory of Management and Control for Complex System, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China; School of Artificial Intelligence, University of Chinese Academy of Science, Beijing, China, xiaolong.zheng@ia.ac.cn

Kang Zhao
Department of Business Analytics, Tippie College of Business, The University of Iowa, Iowa City, IA 52242, USA, kang-zhao@uiowa.edu.

Maggie Wenjing Liu
School of Economics and Management, Tsinghua University, Beijing, 100084, China, liuwenjing@sem.tsinghua.edu.cn.

Daniel Dajun Zeng
The State of Key Laboratory of Management and Control for Complex System, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China; School of Artificial Intelligence, University of Chinese Academy of Science, Beijing, China, dajun.zeng@ia.ac.cn

Abstract

Co-movement among individual firms' stock prices can reflect complex inter-firm relationships. This paper proposes a novel method to leverage such relationships for stock price predictions by adopting inductive graph representation learning on dynamic stock graphs constructed based on historical stock price co-movement. To learn node representations from such dynamic graphs for better stock predictions, we propose the Hybrid-Attention Dynamic Graph Neural Network, an inductive graph representation learning method. We also extended mini-batch gradient descent to inductive representation learning on dynamic stock graphs so that the model can update parameters over mini-batch stock graphs with higher training efficiency. Extensive experiments on stocks from different markets and trading simulations demonstrate that the proposed method significantly improves stock predictions. The proposed method can have important implications for the management of financial portfolios and investment risk.

Keywords: graph representation learning; deep learning; predictive models; business intelligence

1. Introduction

In financial markets, firms are increasingly connected via various relationships (Kwan 1996, Rosenberg and Rudd 1982). One effect of such interconnectedness is the dynamic co-movement phenomenon (Campbell et al. 1997), which depicts the synchronous price movement of firms in stock markets. Researchers have identified several reasons for such co-movement, including common industry or sector affiliations (Chan et al. 2007) or business partnerships via supply chain networks (Agarwal et al. 2017, Rios et al. 2020), in which connected firms share similar operating performance or are economically dependent. In addition, such stock price correlations are also dynamic, as the correlation between two stocks varies over time (Schwert 1989) due to economic conditions (Kim and Qi 2010), industry-wide cycles (Ehling and Heyerdahl-Larsen 2017), and individual firm behaviors (Kock 2005).

Intuitively, better stock movement predictions aid investment decisions, such as asset allocations (Ban et al. 2016). However, existing approaches in stock prediction, including traditional time-series analysis methods from finance literature (Yang et al. 2012) or machine-learning-based methods from computing research (Q. Li et al. 2016), primarily consider the task to be a single-stock prediction problem. In other words, the prediction of a stock's movement only leverages the stock's own features in the past.[1]

As recent research has revealed the predictability of stock prices based on inter-firm relationships (Agarwal et al. 2017), studies have adopted graph-based learning methods (Feng et al. 2019, Si et al. 2014) and used social or business relationships between firms to convert stock prediction into a graph-based problem, where the prediction for one stock can learn from other stocks in a graph. For instance, researchers have identified stocks whose prices are highly correlated with a focal stock and exploited the historical prices of these stocks to predict the

[1] Features could relate to the stock price (e.g., opening and closing prices), trading activities (e.g., trading volume), and other technical indicators (e.g., moving average). We introduce the features used in the experiments in Section 5.1.

focal stock movement (Feng et al. 2019).

However, existing network-based approaches also have several limitations. *First*, these approaches treated all stocks in a graph equally when predicting the movement of a focal stock. In contrast, business relationships between firms are multifaceted, including those based on common sectors, supply chain partners, and shareholders. Such complexity, along with the dynamic co-movement phenomenon, suggests that not all stocks in a graph would contribute equally to the prediction of a focal stock (Chi et al. 2010). *Second*, these methods simply aggregate historical price data from different time periods in the past without differentiating their importance to the prediction. Nevertheless, more recent stock information has a greater influence on predicting the future price than older information (Hu et al. 2018), and periodic patterns in stock movement may also help stock predictions (Elfeky et al. 2005, Yang et al. 2012). *Third*, these studies cannot handle dynamic stock graphs. Because they were based on a static graph with fixed structures, a model trained on one graph cannot be applied to another stock graph with a different structure (Veličković et al. 2018). When relationships between stocks frequently change over time, as in the dynamic price co-movement phenomenon, these methods are not applicable.

To address these limitations, we built dynamic stock co-movement graphs based on publicly available data and proposed an inductive representation learning method for such graphs. The approach first captures temporal relationships between stocks and constructs dynamic graphs using a moving time-window similarity measure of stock co-movement. To learn from such graphs for stock movement predictions, we proposed a Hybrid-Attention Dynamic Graph Neural Network (HAD-GNN), a deep learning framework that incorporates attention mechanisms for both time steps in the past and other nodes in a graph. We further extended the mini-batch gradient descent to inductive representation learning on such dynamic graphs and adopted a t-Batch Training Mechanism (t-BTM) for HAD-GNN, so that HAD-

GNN can be learned over mini-batch dynamic graphs efficiently.

Overall, the contributions of this paper are three-fold. *First*, from a data perspective, the paper represents the first attempt to construct dynamic stock co-movement graphs to predict individual stock prices. Such dynamic stock co-movement graphs are based on publicly available data and can directly reflect the ever-changing relationships between firms' stock performance at a very fine-grained temporal level. *Second*, methodologically, we proposed HAD-GNN, an attention-based inductive and scalable deep learning model for dynamic co-movement graphs. Moreover, HAD-GNN provides a powerful attention-based feature extraction framework, which is the first to consider the varying levels of influence from different historical periods and combine such temporal attention with node-level attention. *Third*, from a design science perspective, this is the first study to adopt the mini-batch gradient descent for inductive representation learning on dynamic graphs. We demonstrated that HAD-GNN can be efficiently trained with t-BTM over stock graphs whose structures change frequently. Besides stock price predictions, the proposed method can also be applied to the learning and inference of other dynamic graphs whose structures can change over time (e.g., collaboration networks).

The rest of this paper is organized as follows. Section 2 reviews the related literature. In Section 3, we introduce how to construct dynamic stock co-movement graphs. In Section 4, the HAD-GNN and *t*-BTM are described in detail. Section 5 presents the experimental results on three real-world stock market datasets. The paper concludes with discussions of future research directions in Section 6.

2. Related Work

Our research is conceptually and methodologically related to previous studies on deep learning-based stock predictions and graph-based stock predictions. In this section, we review related work in these areas and highlight our contributions.

2.1 Deep Learning-based Stock Predictions

Recently, deep learning methods have led to significant advancements in many areas of artificial intelligence, such as computer vision, natural language processing, and speech recognition. Deep learning has also been adopted for stock predictions. Based on the types of inputs for predictions, existing deep learning-based stock prediction approaches can be categorized into finance-based, web-based, and hybrid methods.

Finance-based methods generally use financial data, such as historical trading data (e.g., prices and volumes) and derivative indicators, as inputs to predict closing prices or movements of stocks. Different variations of deep learning methods have been adopted to mine such financial data. For instance, by employing possible long-term temporal dependencies within a time series, Qin et al. (2017) deployed a temporal attention mechanism to select relevant hidden states of encoders across all time steps. In addition, Zhang et al. (2017) modified the memory cell of Long Short-Term Memory (LSTM) to decompose the hidden states of cells into multiple frequency components and combined these components with the inverse Fourier transform to gain more accurate predictions in different future time ranges.

With the availability of large-scale data, including user-generated content from the internet, web-based methods use online content related to the stock markets, such as news stories and social media posts, to predict stock movements (X. Li et al. 2016). Natural language processing and text mining are the most commonly used methods for such online data. Ding et al. (2015) extracted major events from news stories and represented them as dense vectors using the Recurrent Neural Networks (RNN) for stock market prediction. To find valuable information from noisy online news for stock prediction, Hu et al. (2018) designed a hybrid-attention RNN by imitating the learning process of human beings.

Hybrid methods fuse both finance and online data for stock predictions (Kraus and Feuerriegel 2017, Wu et al. 2018, Xu and Cohen 2018). For example, Xu and Cohen (2018) combined features from tweets and historical price information into an LSTM-based neural variational inference framework to improve stock price prediction.

Overall, RNN-based models are the underlying architecture for most deep learning-based stock predictions, primarily because of the RNN's superior performance in modeling time sequence data. However, one major limitation is that these studies have considered each stock to be an independent entity. While the historical performance of a stock indeed provides signals for its future movement, studies in this stream ignored another important source of information—the relationships between firms.

2.2 Graph-based Stock Prediction

To address the limitations mentioned above, graph-based prediction methods incorporate related stock information to improve the prediction for a target stock. Graph-based methods can be traced back to the pairs trading strategy (Elliott et al. 2005), which matches a long position with a short position in two stocks with a high historical correlation. Stock graphs, where stocks are nodes and edges indicate relationships among stocks, represent an important model for studying stock market characteristics (Bonanno et al. 2001, Chi et al. 2010, Dhar et al. 2014). For stock price predictions, researchers have built stock graphs based on co-occurrence in social media (Si et al. 2014) and shared sector or supplier-customer relationships (Feng et al. 2019).

However, these stock graphs reflect various business relationships that only serve as coarse-grained, indirect, or noisy proxies for correlated financial performance. For example, co-search or co-mention of firms on the web may be biased toward firms that attract the public's attention or individual investors. The standard categorization of firms into industry sectors is also becoming more problematic (Hoberg and Phillips 2016) due to modern companies' expansion into multiple business types. For example, Amazon.com has businesses in retail, cloud computing, consumer electronic devices, and digital content distribution. Even two competing firms may have similar movements on some days due to the trends in their common sector (e.g., airlines during COVID-19) or opposite movements on other days when one of them faces a disruption (e.g., Boeing and Airbus when Boeing airplanes were grounded due to safety issues). Possible multiplex relationships between organizations further complicate the matter (Zhao et al. 2012). For example, two firms can be partners and competitors at the same time. Thus, a machine learning model must accommodate such multiplexity if it considers more than one type of business relationship. In addition, it is difficult to obtain large-scale, temporal, and detailed data for inter-firm relationships, such as the flows of products or services and the level of dependencies between supply chain partners, over a long period. Therefore, the first contribution of our work is that, for the task of stock price predictions, we took a more direct approach to capture correlated stock performance by building a stock co-movement graph based on publicly accessible data on stock price correlations in the past. We believe such a graph can represent the aggregated effects of different types of business relationships between firms over time. In addition to this stock graph, the later adoption of the graph attention mechanism allows the model to learn from data which ties are more important for a firm's stock movement prediction during a specific period so that we do not need to decide which specific types of business relationships to use in the model.

From a methodological perspective, although traditional time-series methods, such as vector autoregression, have been used to predict stock prices from stock graphs (Si et al. 2014), recent studies have incorporated more powerful deep learning methods, especially graph representation learning, into stock predictions using stock graphs (Feng et al. 2019). Graph representation learning aims at generating low-dimensional vector representations (e.g., embeddings) for nodes from a graph. Such representations preserve the nodes' structural characteristics and can be input into machine learning algorithms for network inference tasks.

At the same time, because stock prices change frequently, structures of stock co-movement graphs are naturally more dynamic than those of graphs based on relationships, such as business partnerships. However, existing deep learning methods for stock price predictions are based on transductive graph representation learning methods. Such methods include (1) walk-based methods, which sample node sequences via random or guided walks in a graph and generate node embeddings (Grover and Leskovec 2016, Perozzi et al. 2014), and (2) Graph Convolutional Networks, which develop various graph convolutions based on the spectral graph theory (Bruna et al. 2014, Defferrard et al. 2016, Kipf and Welling 2017). However, node embeddings generated by transductive approaches are dependent on graph structures. When the structure of a graph changes (e.g., adding or removing edges or changing the edge weight) or when dealing with a new graph, embeddings have to be relearned. Therefore, stock prediction methods based on transductive graph representation learning do not apply to dynamic stock co-movement graphs whose structures frequently change over time.

In contrast, inductive graph representation learning attempts to learn a set of models or aggregation functions from which node embeddings can be generated from their network neighbors. Hamilton et al. (2017) proposed GraphSAGE, a nonspectral approach that defines convolutions directly on groups of network neighbors at various distances. Other neural network mechanisms, such as self-attention, have also been redefined for graphs (Veličković et al. 2018). Such methods can compute node representations in an inductive and sometimes supervised manner. Thus, they often achieve better performance than transductive methods in network inference tasks. Nevertheless, inductive graph representation approaches incur low training efficiency when a large number of graphs are involved or when graph structures change frequently. We also find another limitation of the existing stock prediction models based on graph representation learning. They do not consider the possible periodic patterns in stock prices nor focus more on recent price movements, which can be more valuable for predicting the next move than distant price changes.

To address these issues, we based the HAD-GNN model on inductive graph representation learning. The proposed model employs time-based attention mechanisms for a time series in stock prices and integrates such mechanisms with node-level graph attention. We adopted a mini-batch gradient descent for the inductive learning setting to efficiently train the HAD-GNN on dynamic stock co-movement networks whose structures change daily.

3. Dynamic Stock Co-movement Graphs

To capture price co-movements among stocks, we constructed dynamic stock co-movement graphs. Intuitively, such graphs are based on similarities between historical prices of different stocks. We defined two types of co-movement graphs: the --Pearson Stock Co-Movement Graph (PSCMG) for predicting stock prices (i.e., a numerical prediction task) and the Manhattan Stock Co-Movement Graph (MSCMG) for predicting stock movement directions (i.e., a classification task). Due to the temporal dynamics in stock prices, price similarities also change over time. Thus, we used the moving time-window method introduced by Tian et al. (2019) to construct dynamic stock co-movement graphs.

We defined a node (stock) set $V = \{v_1, v_2, \cdots, v_L\}$, with L nodes. The feature vector of stock $v_i \in V$ at time t is $\mathbf{x}_t^i = \left(x_{t,1}^i, x_{t,2}^i, \cdots, x_{t,m}^i\right) \in \mathbb{R}^m$, where m is the number of features. The prediction is made daily, and each time step represents one trading day. In addition, p_t^i represents the closing price of v_i on trading day t. Please see Online Supplement A for the notation in this paper.

3.1 Pearson Stock Co-Movement Graphs

The PSCMGs are based on correlations of stock returns. Because stock return rates often follow highly skewed distributions, we applied logarithmic transformations so that log-transformed returns are normally distributed (Hudson and Gregoriou 2015, Singleton and Wingender 1986). This normality of distributions is desirable for many statistical analyses and machine learning applications. Equation 1 defines the logarithmic return rate for stock v_i at time t:

$$r_t^i = \log \frac{p_t^i}{p_{t-1}^i}. \tag{1}$$

Then the Pearson correlation coefficient $\rho_t^{ij}(\Delta t)$ between stock v_i and stock v_j at time t during the past Δt days is the Pearson correlation between their return rates r_t^i and r_t^j during Δt.

DEFINITION 1: PEARSON STOCK CO-MOVEMENT GRAPH (PSCMG). Given an observation time window of length Δt, we define the PSCMG at time t as an undirected graph $G_t^P(\Delta t) = \{V, E_t^P, \mathbf{X}_t\}$. In addition, $V = \{v_1, v_2, \cdots, v_L\}$ is a set of $L = |V|$ nodes, and $E_t^P \subseteq V \times V$ is the set of edges among nodes, where $e_{ij} = (v_i, v_j) \in E_t^P$ denotes an edge between nodes v_i and v_j. Further, $\mathbf{X}_t = \{X_t^i, v_i \in V\}$ denotes the set of features of all stocks in V. Additionally, $A_t^P \in \mathbb{R}^{L \times L}$ is the binary adjacency matrix for $G_t^P(\Delta t)$ based on a threshold value of $0 \leq \delta^P \leq 1$. If $\left|\rho_t^{ij}(\Delta t)\right| > \delta^P$, then edge e_{ij} exists between v_i and v_j, and $A_t^P(i,j) = 1$; otherwise $A_t^P(i,j) = 0$.

In other words, two nodes (stocks) are connected in $G_t^P(\Delta t)$ if their prices have high correlations (positive or negative) during Δt. We considered both positive and negative correlations because both types can potentially offer valuable prediction signals. For example, two firms with partnerships may have a high and positive price correlation (e.g., Boeing and Delta Airlines), whereas a high and negative correlation may exist between two competing firms (e.g., Coca-Cola vs. Pepsi). We incorporated both into the stock graph so that the model has the chance to learn the specific level of influence between a pair of connected stocks.

3.2 Manhattan Stock Co-Movement Graphs

We used the following definition to categorize stock price movements.

DEFINITION 2: DAILY MOVEMENT LABEL. Given price change threshold values

$\mu_1(1) \in [0, \infty)$ and $\mu_2(1) \in (0,1]$, we define the daily movement label for stock v_i at time t as l_t^i:

$$l_t^i = \begin{cases} 1, \text{if } p_t^i / p_{t-1}^i > 1 + \mu_1(1); \\ 0, \text{if } 1 - \mu_2(1) < p_t^i / p_{t-1}^i \leq 1 + \mu_1(1); \\ -1, \text{otherwise.} \end{cases} \tag{2}$$

The above definition separates stock price movements into three discrete categories. When the price return ratio of a stock rises above (or descends below) a certain threshold, we assign a positive "rise" (or a negative "fall") label to the stock on that day. Otherwise, we assign a neutral class label *0* to the stock. Then the Manhattan distance between stocks v_i and v_j until time t is

$$d_t^{ij}(\Delta t) = \sum_{s=t-\Delta t+1}^{t} \left| l_s^i - l_s^j \right|, \tag{3}$$

where Δt is the size of the observation time window, and $0 \leq d_t^{ij}(\Delta t) \leq 2 * \Delta t$. We used the Manhattan distance instead of the Euclidean distance because it has a lower computational cost and better performance in finding neighbors in stock graphs (Aggarwal et al. 2001).

DEFINITION 3: MANHATTAN STOCK CO-MOVEMENT GRAPH (MSCMG). Given the size Δt of the observation time window, we define MSCMG at time t as an undirected graph $G_t^M(\Delta t) = \{V, E_t^M, \mathbf{X}_t\}$. The MSCMG is similar to the PSCMG, except that its adjacency matrix $A_t^M \in \mathbb{R}^{L \times L}$ is based on the Manhattan distance instead of the Pearson correlation between stocks. With $\delta^M \in [0, 2 * \Delta t]$, if $\delta^M \leq d_t^{ij} \leq (2 * \Delta t - \delta^M)$, then $A_t^M(i,j) = 0$; otherwise $A_t^M(i,j) = 1$. In other words, similar to the PSCMG, stocks that have quite similar or very different movement labels during Δt are connected in $G_t^M(\Delta t)$.

The Pearson correlation coefficient and Manhattan distance are calculated based on the time series of the stock price within a moving observation time window with length Δt that ends at t. Thus, the structures of the two graphs naturally change over time, reflecting the dynamic nature of stock price co-movements. To simplify the notation, we use the same notation G_t in the remainder of the paper to represent $G_t^P(\Delta t)$ and $G_t^M(\Delta t)$ in numerical prediction and classification tasks, respectively.

4. Stock Predictions Based on Co-movement Graphs

In this section, we first formalize the problem of stock prediction. Then, we present the inductive learning framework HAD-GNN, which incorporates hybrid-attention encoders into inductive graph representation learning to better leverage dynamic stock co-movement graphs for stock predictions. Afterward, we introduce the t-batch training mechanism, which can be used to train the proposed learning framework efficiently.

4.1 Problem Formulation

In stock predictions, the target to predict can be a stock's numerical return rate or movement trend in the future. We formulated the τ-step prospective return rate prediction as an inductive node-regression task and the τ-step prospective movement prediction as an inductive node-classification task.

DEFINITION 4: τ-STEP AHEAD RETURN RATE PREDICTION. Given a future time window τ, we define the τ-step ahead return for stock v_i at time t as $r_t^i(\tau)$:

$$r_t^i(\tau) = \frac{p_{t+\tau}^i}{p_t^i} - 1. \tag{4}$$

Then, at time t, the goal is to predict the return rates τ days later for stocks in G_t using the HAD-GNN:

$$\hat{\mathbf{r}}_t(\tau)=\text{HAD-GNN}(G_t; \mathbf{\Theta}), \tag{5}$$

where $\hat{\mathbf{r}}_t(\tau) = (\hat{r}_t^1(\tau), \hat{r}_t^2(\tau), \cdots, \hat{r}_t^L(\tau))$ is the vector of predicted return rates for L stocks in G_t, and $\mathbf{\Theta}$ represents all the learnable parameters of the HAD-GNN.

DEFINITION 5: τ-STEP AHEAD MOVEMENT PREDICTION. Given the threshold values $\mu_1(\tau) \in [0, \infty)$ and $\mu_2(\tau) \in (0, 1]$ for a future time window τ, we define the τ-step ahead movement label for stock v_i at time t as $y_t^i(\tau)$, which categorizes a stock's price change τ days after t:

$$y_t^i(\tau) = \begin{cases} 1, \text{if } p_{t+\tau}^i / p_t^i > 1 + \mu_1(\tau); \\ 0, \text{if } 1 - \mu_2(\tau) < p_{t+\tau}^i / p_t^i \leq 1 + \mu_1(\tau); \\ -1, \text{otherwise.} \end{cases} \tag{6}$$

Then, at time t, we predict movement labels τ-day later for stocks in G_t using the HAD-GNN:

$$\hat{\mathbf{y}}_t(\tau)\text{=HAD-GNN}(G_t; \mathbf{\Theta}), \tag{7}$$

where $\hat{\mathbf{y}}_t(\tau) = (\hat{y}_t^1(\tau), \hat{y}_t^2(\tau), \cdots, \hat{y}_t^L(\tau))$ is the vector of predicted labels for L stocks in G_t, and $\mathbf{\Theta}$ represents all the learnable parameters of the HAD-GNN.

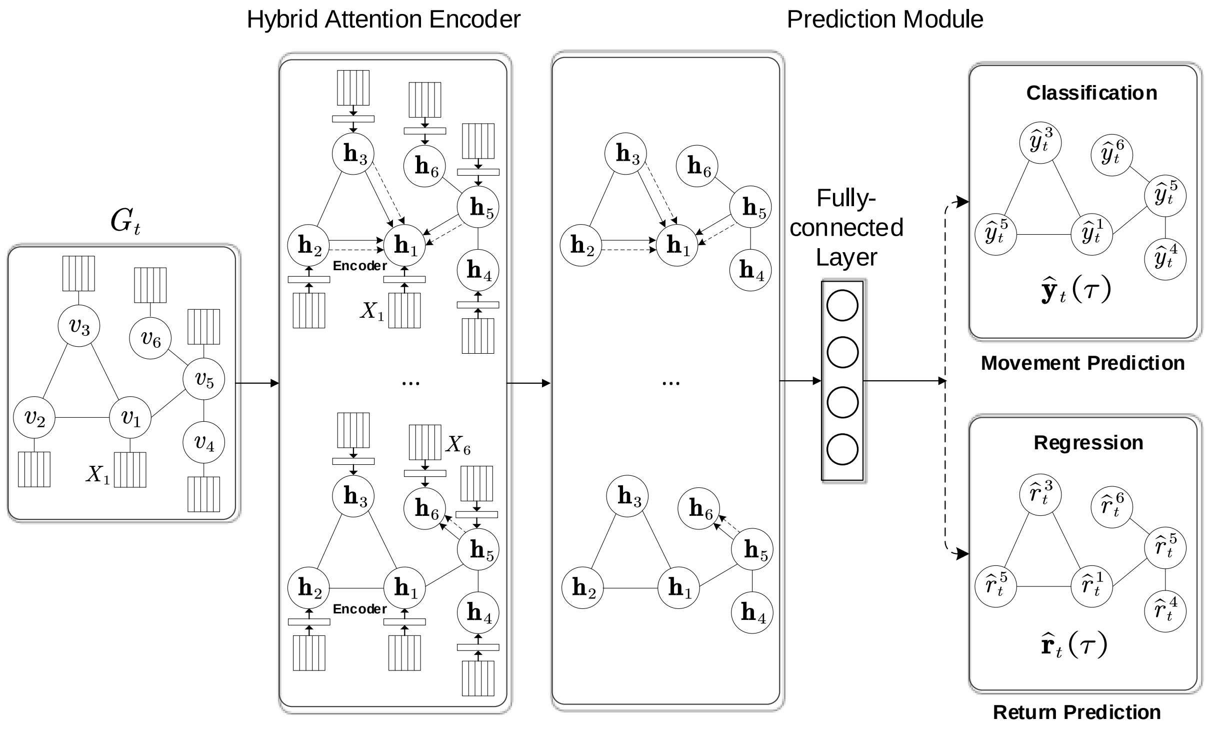

Fig. 1. Architecture of the HAD-GNN illustrated for sample stock graph G_t with six nodes and six edges.

4.2 Graph Hybrid-Attention Networks

The HAD-GNN (Fig. 1) consists of two components: the hybrid-attention encoder and prediction module. The hybrid encoder generates embedding vectors for each stock by considering the time sequence of each stock's historical features and the influence of neighboring stocks on dynamic stock graphs. The prediction module takes a stock's embedding vector from the hybrid encoder as input and generates movement or return rate predictions for the stock. We describe these two components in this subsection.

4.2.1 Hybrid-Attention Encoder

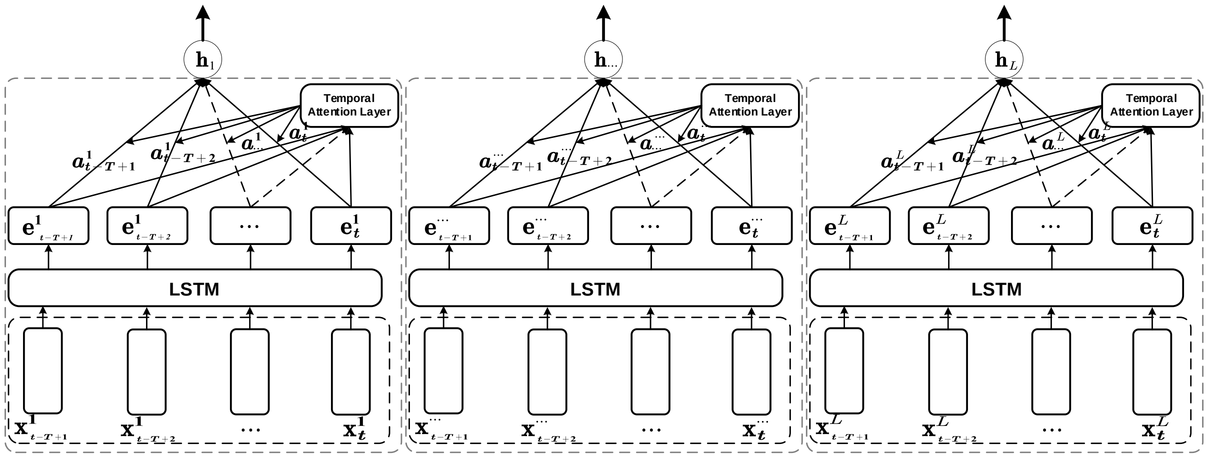

Fig. 2. Illustration of the temporal attention-based long short-term memory (LSTM) for the hybrid-attention encoder.

We designed a hybrid-attention encoder with both time-level and node-level feature extractions. The goal is to capture the influence of features from different time steps in the past and from different peer stocks for predicting a focal stock. At the time level, we built a shared temporal attention-based LSTM to encode historical features from nodes (i.e., stocks). The temporal attention mechanism performs feature extractions from features for different time steps when the length of the observation period T becomes long (Qin et al. 2017, Raffel and Ellis 2015). As illustrated in Fig. 2, we first encoded historical features at time t by inputting the T-lag ($T \geq 1$) historical feature matrix $X_t^i = [\mathbf{x}_{t-T+1}^i, \mathbf{x}_{t-T+2}^i, \cdots, \mathbf{x}_t^i] \in \mathbb{R}^{m \times T}$ for stock v_i into the LSTM, where $i \in [1, L]$, m represents the number of features, and T is the temporal length of the observation period when features are extracted:

$$[\mathbf{e}_s^i]_{s=t-T+1}^{t} = \mathrm{LSTM}(X_t^i). \tag{8}$$

The output of the LSTM encoder $\mathbf{e}_s^i \in \mathbb{R}^{m'}$ is the latent vector representation with dimension m' for historical features from time s.

To distinguish the contribution of historical features at different moments from $t - T + 1$ to t, we introduced a temporal attention layer, which can adaptively assign weights to vector $\mathbf{e}_s^i$ and combine them into an expressive high-level vector via a weighted sum. The numerical weight (i.e., importance) of vector $\mathbf{e}_s^i$ is generated and normalized via the temporal attention layer as follows:

$$\tilde{a}_s^i = \tanh\left(W_{TA}\mathbf{e}_s^i + \mathbf{b}_{TA}\right), \tag{9}$$

where $\mathbf{b}_{TA} \in \mathbb{R}^T$ and $W_{TA} \in \mathbb{R}^{T\times m'}$ are parameters to learn for temporal attention (TA), and the activation function $\tanh(\cdot)$ operates element-wise on a given vector. In addition, $\tilde{a}_s^i$ is transformed (i.e., normalized) via the softmax function into a_s^i so that the value of the weight falls within the internal [0,1]. Vector $\mathbf{a}^i = [a_{t-T+1}^i, \cdots, a_t^i]$ assigns a weight for $\mathbf{e}_s^i$ for each time step between $t - T + 1$ and t. Thus, we obtain a temporal-attention-based representation $\mathbf{h}_i$ for each stock $v_i \in V$ in stock graph G_t:

$$\mathbf{h}_i = \sum_{s=t-T+1}^{t} a_s^i \mathbf{e}_s^i \,. \tag{10}$$

As all values of $\mathbf{h}_i \in \mathbb{R}^{m'}$ are for prediction at time t, we omitted the superscript t for simplicity. The temporal attention-based LSTM transforms historical features of a stock into a high-level latent vector representation. In addition, according to Eqs. 8, 9, and 10, all parameters of the temporal attention-based module are independent of the graph structure and can be shared by all stocks for all observation periods with length T. For simplicity, we used $\mathbf{\Theta}_{TA}$ to denote parameters of the temporal attention encoder, including parameters of LSTM, W_{TA}, and $\mathbf{b}_{TA}$.

Inspired by Graph Attention Networks (GAT) (Veličković et al. 2018), we also adopted node-level graph attention mechanisms (illustrated in Fig. 3) to model the different effects of neighboring stocks on a focal stock. The mechanism takes latent representations $\{\mathbf{h}_i\}_{i=1}^{L}$

generated by the temporal attention-based LSTM as input for all stocks in G_t. For each connected stock pair $v_i, v_j \in V$, a linear transformation (i.e., a one-layer neural network) is applied to the latent representations $\mathbf{h}_i$ and $\mathbf{h}_j$, respectively:

$$\tilde{\mathbf{h}}_i = W_{GA}\mathbf{h}_i, \tilde{\mathbf{h}}_j = W_{GA}\mathbf{h}_j, \tag{11}$$

where $W_{GA} \in \mathbb{R}^{m' \times m'}$ denotes graph attention (*GA*) parameters to be learned. Then, $\tilde{\mathbf{h}}_i$ and $\tilde{\mathbf{h}}_j$ are concatenated as a new vector, which is input into another one-layer neural network, parameterized by a weight vector $\mathbf{u}_{GA}$ that is learned. After the dot product with $\mathbf{u}_{GA}$ and a nonlinear transformation (e.g., LeakyReLU), the concatenated vector is converted into a numerical coefficient, which is input into a softmax function to generate n_{ij}. The normalized attention coefficient n_{ij} describes the importance of stock v_j (represented by $\mathbf{h}_j$) to the prediction of stock v_i (represented by $\mathbf{h}_i$). The process can be summarized as follows:

$$n_{ij} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{u}_{GA}^{\intercal}[\tilde{\mathbf{h}}_i; \tilde{\mathbf{h}}_j])\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}(\mathbf{u}_{GA}^{\intercal}[\tilde{\mathbf{h}}_i; \tilde{\mathbf{h}}_k])\right)}, \tag{12}$$

where $\mathbf{u}_{GA}^{\intercal}$ is the transpose of $\mathbf{u}_{GA}$, $\mathcal{N}_i$ is the first-order neighborhood (i.e., direct neighbors) of node v_i (including v_i) in stock graph G_t, and $[\cdot;\cdot]$ concatenates two vectors. The activation function LeakyReLU is a variation of the rectified linear unit (ReLU) (Nair and Hinton 2010). Designed to fix the "dying ReLU" problem, it can often make training faster (Maas et al. 2013) and was adopted by the original GAT model (Veličković et al. 2018). The LeakyReLU has demonstrated better performance than ReLU in analyzing images and speech (Xu et al. 2015, Zhang et al. 2017). Additional experiments (Fig. C1 in Online Supplement C) reveal that the proposed model's performance is robust when using ReLU as the activation function.

With attention coefficients for each neighbor of v_i, we obtain a more comprehensive vector representation of v_i via a weighted sum of all latent representations of the neighboring stocks $\mathbf{h}_i' \in \mathbb{R}^{m'}$:

$$\mathbf{h}'_i = \sum_{j \in \mathcal{N}_i} n_{ij} \mathbf{h}_i \,. \tag{13}$$

By traversing all stocks in G_t and executing the transformations listed in Eqs. 11, 12, and 13, we obtain vector representations for all stocks in G_t. Parameters of the node-level graph attention apply to every node. They are independent of the graph structure and can be shared by all nodes across all time steps.

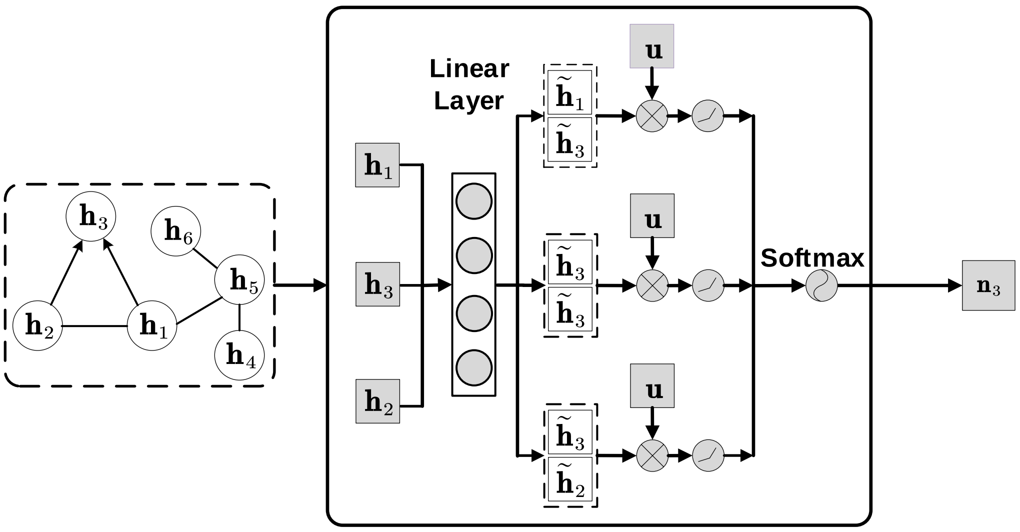

Fig. 3. Illustration of the node-level graph attention mechanism of the hybrid-attention encoder, with node v_3, its latent representation $\mathbf{h}_3$, and its node-level attention coefficient vector $\mathbf{n}_3$ as an example. Boxes with dotted boundaries represent concatenations of vectors. The process of learning node-level graph attention is illustrated in Eqs. 12, 13, and 14.

After transformations by the hybrid-attention encoder, all raw features of stocks over time are encoded as vector representations $\mathbf{H}' = [\mathbf{h}'_1, \mathbf{h}'_2, \cdots, \mathbf{h}'_L]$. However, $\mathbf{h}'_i$ only aggregates information from the first-order neighborhood of stock v_i (i.e., direct neighbors). Researchers have found that information from second-order neighborhoods (i.e., two-hop neighbors or neighbors of neighbors) of a focal stock can also improve predictive powers (Fan et al. 2019). Thus, we added another graph attention layer, parametrized by W'_{GA} and $\mathbf{u}'_{GA}$, to transform $\mathbf{h}'_i$ into vector $\mathbf{h}''_i \in \mathbb{R}^{m'}$. The transformation is performed in the same way that $\mathbf{h}_i$ is transformed into $\mathbf{h}'_i$ (described above), so that the output of the second graph attention layer $\mathbf{H}'' = [\mathbf{h}''_1, \mathbf{h}''_2, \cdots, \mathbf{h}''_L]$ incorporates signals from both first- and second-order neighbors of individual stocks. For simplicity, we used $\mathbf{\Theta}_{GA} = \{W_{GA}, \mathbf{u}_{GA}, W'_{GA}, \mathbf{u}'_{GA}\}$ to denote the parameters of this two-layer node-level graph attention encoder.

4.2.2 Prediction Module

For stock v_i, its vector representation $\mathbf{h}''_i$ generated by the hybrid-attention encoder serves as input for the prediction module, which consists of a fully connected (FC) neural network. Depending on the predictive task, the prediction module can use different activation and loss functions.

As a classification task, stock movement predictions can use an FC layer with the activation function ReLU, which is more suitable for classification (Nair and Hinton 2010):

$$\mathbf{o}_i = \mathrm{ReLU}(W_O\mathbf{h}''_i + \mathbf{b}_O), \tag{14}$$

where $\mathbf{o}_i$ is the output for stock v_i from the FC layer parameterized by $W_O \in \mathbb{R}^{C\times m'}, \mathbf{b}_O \in \mathbb{R}^C$, and C represents the number of classes or labels to predict. The softmax activation function normalizes elements within vector $\mathbf{o}_i$ into confidence scores, and the class with the highest score is chosen as the prediction for stock v_i. Standard cross-entropy over all classes and all stocks can be used as the basic loss function to train the classification model.

As a regression task, stock return rate predictions use an FC layer with the activation function *tanh*. This function maps any real value into the [-1,1] interval and is more suitable for predicting return rates whose values fall into the same interval:

$$\hat{r}^i = \tanh(W_R\mathbf{h}''_i + \mathbf{b}_R), \tag{15}$$

where $\hat{r}^i$ is the predicted return of stock v_i from the FC layer parameterized by $W_R \in \mathbb{R}^{m'}, \mathbf{b}_R \in \mathbb{R}$. The standard mean absolute error between the predicted and actual return rates for all stocks serves as the basic loss function for this prediction task.

For both stock movement and return rate prediction tasks, regularization terms are also added to the basic loss functions to prevent overfitting, a common practice in machine learning research (Goodfellow et al. 2016). Therefore, during the training period $[1, T_R]$, the final loss function is as follows:

$$\mathcal{L} = \sum_{t=1}^{T_R} \mathcal{L}^t + \frac{\gamma}{2} \|\mathbf{\Theta}\|_2^2 , \tag{16}$$

where $\mathcal{L}^t$ represents the basic loss function at time t for a specific prediction task, $\mathbf{\Theta}$ refers to the set of all parameters of the HAD-GNN, $\|\cdot\|_2$ is the L_2 norm, and γ is a hyperparameter that controls the weight of the regularization term. The objective function is a regularized loss function commonly used in machine learning research (Goodfellow et al. 2016). The first part attempts to minimize prediction errors, whereas the second part (i.e., the regularization term) aims to reduce the model complexity.

4.3 t-Batch Training Mechanism for the Hybrid-attention Dynamic Graph Neural Network

To solve the optimization problem defined in Eq. 16, we can use the stochastic gradient descent, batch gradient descent, or mini-batch gradient descent algorithms. Batch gradient descent computes the gradient over all stock graphs, which demands a tremendous amount of memory and easily lands the batch gradient trajectory in a saddle point (Goodfellow et al. 2016). In contrast, stochastic gradient descent computes the gradient on each stock graph of the dataset and helps a model escape from saddle points or local minima. However, it suffers from a longer training time due to frequent updates to the model (Goodfellow et al. 2016). Mini-batch gradient descent algorithms represent a middle-ground solution between the batch and stochastic gradient descent. It divides stock graphs into small batches to evaluate and update models. Specifically, for an inductive model, such as the HAD-GNN, where all stock graphs G_t share the same set of parameters $\mathbf{\Theta}$, the reduced model complexity makes the mini-batch gradient descent an even better fit to efficiently train the HAD-GNN.

We adopted the mini-batch gradient descent and called this approach the t-BTM for the HAD-GNN. Specifically, we regarded stock graph G_t at time t as a graph-level instance. Then the training set can be denoted as $\mathcal{G}_{\mathrm{R}} = \{G_t\}_{t=1}^{T_R}$. Due to time dependencies between dynamic stock graphs (Kumar et al. 2019), the t-BTM temporally generates a sequential t-batch $\{G_{t-S_B+1}, G_{t-S_B+2}, \cdots, G_t\}$ at time t ($t \geq S_B$) from $\mathcal{G}_{\mathrm{R}}$ and allows the HAD-GNN to be trained using the gradient descent over the t-batch. In addition, S_B represents the t-batch size. If $S_B = 1$, the t-BTM is the same as the stochastic gradient descent, and if $S_B = T_R$, which is the length of the training set, the t-BTM becomes the batch gradient descent. Figure 4 illustrates the architecture of the t-BTM.

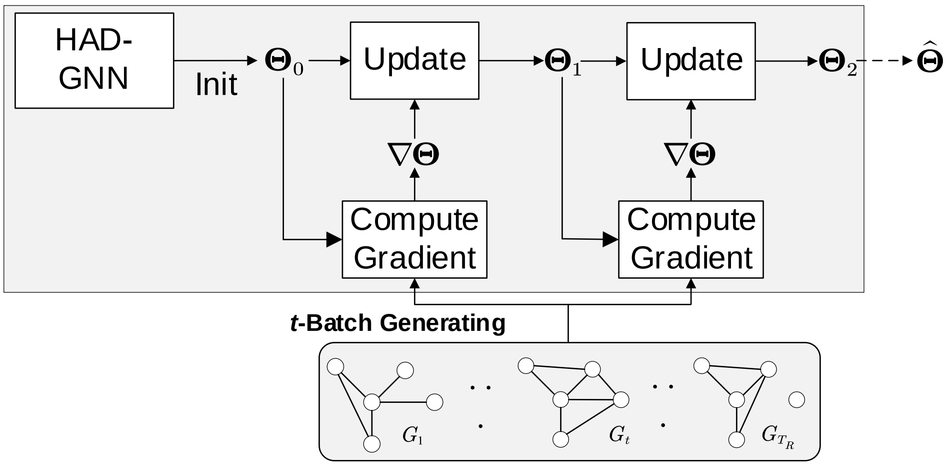

Fig. 4. Illustration of the t-batch training mechanism for the HAD-GNN.

Similar to other deep learning methods, the t-BTM trains the HAD-GNN by calculating the loss and updating the model parameters over the t-batch stock graphs, which can be efficiently accelerated with graphic processing units (GPUs). In this study, we selected the Adam optimizer (Kingma and Ba 2015), a variant of the gradient descent, as the update method for the model parameters because of its high training speed (Ruder 2016). The overall training process for the HAD-GNN is summarized in Algorithm 1 in Online Supplement A.

5. Results

This section presents results from experiments on real-world datasets to evaluate the performance of the proposed HAD-GNN, along with the ablation, efficiency, and sensitivity analyses. The practical value of the model in trading practices is also demonstrated via simulated trading.

5.1 Datasets and Experiment Setup

We collected datasets from Yahoo! Finance for three major stock markets in the US, China, and Australia. These stocks represent major companies in these market indices: Standard and Poor's 500 (S&P 500), China Securities Index 300 (CSI 300), and Australian Securities Exchange 300 (ASX 300). Table 1 provides detailed statistics for these datasets. Although the three markets may differ in trading mechanisms and regulations, the underlying motivation for this model should still be valid—the prediction of one stock can benefit from learning from other stocks whose prices have been correlated with the focal stock.

Table 1. Detailed information on three stock datasets.

Market	USA	China	Australia
Index	S&P 500	CSI 300	ASX 300
Data period	2/20/2013–12/28/2018	11/27/2013–3/21/2018	3/5/2013–12/28/2018
No. of stocks	475	153	229
No. of trading days	1476	1054	1476
T_R	792	560	792
T_V	198	140	198
T_S	486	354	486

After removing stocks traded on fewer than 98% of all trading days during the corresponding data collection periods, we retained 475 (S&P 500), 153 (CSI 300), and 229 (ASX 300) stocks from the three datasets. As for features, the raw data for a stock on a given trading day contains five features: the opening price, high price, low price, closing price, and trading volume. Following standard practices in time-series forecasting (Chatfield and Xing 2019), we preprocessed the raw stock closing price using exponential smoothing with a smoothing factor of 0.9 and normalization. For the predictions, we also incorporated seven technical indicators (listed in Table 2) widely used by investors as additional features. Each dataset was split into three time intervals: $[1, T_R]$ as training (53.3% of the data), $[T_R + 1, T_R +$

$T_V]$ as validation (13.3%), and $[T_R + T_V + 1, T_R + T_V + T_S]$ as testing (33.4%). Datasets and code used in this research have been made publicly available online[2].

We conducted two types of experiments: stock movement classifications using the MSCMG and stock return rate regressions using the PSCMG. We designed the τ-day-ahead prediction tasks for each dataset using three τ values: $\tau = 1, 5,$ and 9.

Table 2. Seven technical indicators in the prediction model.

Technical indicator	Definition
Moving Average Convergence/Divergence	https://en.wikipedia.org/wiki/MACD
Relative Strength Index	https://en.wikipedia.org/wiki/Relative_strength_index
Stochastic Oscillator %K	https://en.wikipedia.org/wiki/Stochastic_oscillator
Williams %R	https://en.wikipedia.org/wiki/Williams_%25R
On-Balance Volume	https://en.wikipedia.org/wiki/On-balance_volume
Price Rate of Change	https://www.investopedia.com/terms/p/pricerateofchange.asp
$y_{t-5}^i(5)$	5-step ahead movement label at time $t - 5$ defined in Eq.6.

To set up τ-day-ahead stock movement predictions, we chose threshold values $\mu_1(\tau)$ and $\mu_2(\tau)$ so that the three classes (+1, 0, and -1) have balanced prior distributions. This practice is common in machine learning research (Xie and Qiu 2007) because imbalanced datasets pose challenges for most machine learning algorithms (Krawczyk 2016). In practice, while traders can choose threshold values they want to predict (e.g., an extremely high or low magnitude of movement), we still recommend that they maintain a balanced class distribution to avoid highly skewed prior distributions that are difficult for machine learning algorithms to learn from.

Table 3(a). Parameters for stock movement predictions.

Dataset	τ	Nonlearning parameters					Learning parameters		
		$\mu_1(\%)$	$\mu_2(\%)$	Δt	T	δ^M	t-batch size	Learning rate	m'
S&P 500	1	0.50	0.50	15	10	5	10	1e-4	64
	5	1.35	0.85						
	9	2.00	1.15						
CSI 300	1	0.70	0.60	15	10	6	10	1e-4	64
	5	1.40	1.40						
	9	2.80	1.70						
ASX 300	1	0.55	0.60	15	10	6	10	1e-4	64
	5	1.55	1.45						
	9	2.24	1.93						

Table 3(b). Parameters for stock return rate predictions (HAD-GNN with different τ shares the same

[2] https://github.com/Hugo-CAS/Dynamic-Stock-Co-Movement-Graphs-for-Stock-Predictions

hyperparameters).

Dataset	Nonlearning parameters			Learning parameters		
	Δt	T	δ^P	t-batch size	Learning rate	m'
S&P 500	15	10	0.80	10	5e-3	64
CSI 300	15	10	0.80	10	5e-3	64
ASX 300	15	10	0.75	10	5e-3	64

To construct the MSCMG, the observation time window Δt was set to 15 days. The nonlearning parameters Δt, T, and δ^M are involved in building stock graphs and formulating predictions. We fixed the values of Δt and T to 15 and 10, respectively, for all datasets because we found that the performance of the HAD-GNN is stable when the values vary in some intervals (Section 5.4). Tables 3(a) and 3(b) summarize these parameters in the experiments for τ-day-ahead stock movement predictions and return rate predictions, respectively. Another parameter for the MSCMG is the edge weight threshold δ^M, which was chosen for each dataset using validation sets, along with its counterpart edge weight threshold δ^P for the PSCMG (details in Table 3(b)).

Tables 3(a) and 3(b) summarize the hyperparameters for learning node representations with the HAD-GNN. The t-batch size in the t-BTM, the learning rate used in the Adam optimizer, and the dimensionality of the hidden layer m' were chosen based on the validation sets.

Benchmark methods include individual-based prediction methods and graph-based prediction methods, both suited for classification and regression tasks. Individual-based methods include random guess (RG), the classic time-series prediction method autoregressive integrated moving average (ARIMA), the traditional machine learning method support vector machine (SVM) (support vector regression for return prediction), and two state-of-the-art deep learning models: LSTM-Att (Hu et al. 2018) and StockNet (Xu and Cohen 2018). Graph-based methods include the transductive node representation learning approach DeepWalk (Perozzi et al. 2014) and two state-of-the-art inductive graph representation learning methods: GraphSAGE (Hamilton et al. 2017) and GAT (Veličković et al. 2018). Our experiments used two convolutional layers for GraphSAGE and GAT to incorporate information from two-hop neighbors, like the HAD-GNN. All graph-based methods, including the proposed method, are based on the same dynamic MSCMG/PSCMG. Table B1 in Online Supplement B describes these benchmark methods and their parameters.

As for performance metrics, we used the standard accuracy and macro-F1 for movement predictions and the mean absolute error (MAE) for the return rate predictions. All models were trained on the training sets with parameter tuning on the validation sets. After selecting the hyperparameters, the models were retrained on the union of training sets and validation sets. The performance was finally evaluated on the testing sets.

5.2 Predictive Performance

Due to space limitations, this section only reports the predictive performance for stocks in the S&P 500. The average performance for stocks in the CSI 300 and ASX 300 is similar and is reported in Online Supplement C. The predictions are for individual stocks, and we aggregated the predictive performance for individual stocks in one market for easier comparisons.

Table 4. Comparison of the average performance on stock movement predictions for S&P 500 stocks (best-performers are bolded).

τ	1		5		9	
Evaluation metrics	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
RG	22.07^{***}	16.58^{***}	22.26^{***}	16.68^{***}	22.59^{***}	16.80^{***}
ARIMA	33.91^{***}	25.57^{***}	31.29^{***}	29.86^{***}	31.78^{***}	26.26^{***}
SVM	35.17^{***}	25.56^{***}	33.17^{***}	26.57^{***}	31.71^{***}	26.30^{***}
LSTM-Att	37.38^{**}	27.65^{***}	33.53^{***}	23.56^{***}	32.82^{***}	23.36^{***}
StockNet	37.43^{**}	26.17^{***}	33.21^{***}	24.90^{***}	31.94^{***}	23.13^{***}
Rn-LSTM	35.75^{***}	24.06^{***}	33.94^{***}	29.33^{***}	32.39^{***}	25.18^{***}
DeepWalk	34.43^{***}	25.12^{***}	32.57^{***}	26.19^{***}	31.91^{***}	25.10^{***}
GraphSAGE	37.12^{**}	24.67^{***}	34.10^{***}	21.11^{***}	34.28^{**}	23.67^{***}
GAT	36.54^{**}	26.75^{***}	34.87^{**}	25.98^{**}	34.54^{**}	24.73^{***}
HAD-GNN	**39.52**	**32.91**	**36.04**	**33.20**	**35.85**	**32.05**

Note: **: p-value < 0.05, ***: p-value < 0.001. All experiments were repeated five times.

Table 4 compares the performance for all methods for τ-day-ahead movement predictions for stocks in the S&P 500 dataset. The proposed HAD-GNN method dominates all benchmark methods across the three τ values on both performance metrics. Specifically, compared to a random guess, the improvements in the accuracy for the HAD-GNN are 79.07% ($\tau = 1$), 62.47% ($\tau = 5$), and 58.70% ($\tau = 9$). The improvements in macro-F1 reach 98.49% ($\tau = 1$), 99.04% ($\tau = 5$), and 90.77% ($\tau = 9$). Compared with the nonrandom guess methods, the improvements in accuracy for the HAD-GNN range from 5.58% to 16.54% ($\tau = 1$), from 3.36% to 13.62% ($\tau = 5$), and from 3.79% to 13.06% ($\tau = 9$). The improvements in macro-F1 scores from the HAD-GNN are from 19.02% to 23.03% ($\tau = 1$), from 11.19% to 57.27% (τ =5), and from 21.86% to 38.56% ($\tau = 9$). The improvement is also statistically significant with all p-values below 0.05 for the pairwise t-tests between the HAD-GNN and benchmark methods.

Table 5 summarizes the performance for all methods for τ-day-ahead return rate predictions on S&P 500 stocks. Again, the proposed HAD-GNN has the best performance across the three τ values. Specifically, compared with a random guess, the HAD-GNN reduces the MAE by 38.49% ($\tau = 1$), 35.46% ($\tau = 5$), and 34.25% ($\tau = 9$). Compared with the nonrandom guess methods, the HAD-GNN reduces the MAE by 1.12% to 18.38% ($\tau = 9$), 3.39% to 10.14% ($\tau = 9$), and 1.27% to 6.69% ($\tau = 9$). The p-values of pairwise t-tests between the HAD-GNN and benchmark methods are below 0.05, except for the comparison with GraphSAGE ($p < 0.1$when τ = 1) and the comparison with GAT ($p < 0.1$ when τ = 1 and 9).

Table 5. Comparison of the average performance on stock return predictions for S&P 500 stocks (best-performers are bolded).

τ	1	5	9
Evaluation metrics	MAE	MAE	MAE
RG	1.725E-2***	3.886E-2***	5.215E-2***
ARIMA	1.300E-2***	2.791E-2***	3.675E-2***
SVR	1.265E-2***	2.714E-2**	3.591E-2**
LSTM-Att	1.113E-2***	2.634E-2**	3.587E-2**
StockNet	1.139E-2***	2.684E-2**	3.620E-2***
Rn-LSTM	1.092E-2**	2.672E-2**	3.588E-2**
DeepWalk	1.149E-2**	2.712E-2**	3.598E-2**
GraphSAGE	1.073E-2*	2.596E-2**	3.583E-2**
GAT	1.073E-2*	2.614E-2**	3.473E-2*
HAD-GNN	**1.061E-2**	**2.508E-2**	**3.429E-2**

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

The improvement of the proposed method over individual-based methods demonstrates the power of considering co-movement relationships between stocks. The superior performance of the proposed method over graph-based benchmark methods can be attributed to the novel design of the HAD-GNN, which provides inductive representation learning on dynamic stock co-movement graphs with a hybrid-attention mechanism. As one would expect, unsupervised representation learning based on DeepWalk does not perform as well as other supervised methods, including the proposed method. Moreover, the proposed model outperforms the Rn-LSTM, which arbitrarily picks the top-three correlated stocks with a focal stock and concatenates the movements of the three stocks to predict the focal stock. This outcome demonstrates the value of choosing similar stocks based on dynamic co-movement graphs and using node-level graph attention to determine which neighbors of a focal stock can provide better signals for predictions.

In addition, GraphSAGE can be considered a simplification of the HAD-GNN without the hybrid-attention encoder. Thus, the improvement of the HAD-GNN over GraphSAGE comes from feature extraction based on the hybrid-attention encoder. In addition, because the GAT only considers graph attention, our model's better performance over GAT highlights the value of a hybrid-attention mechanism that also incorporates temporal attention.

5.3 Ablation Analyses

To further evaluate how dynamic stock co-movement graphs and node-level attention in the proposed model contributes to the performance of the model, we compared the proposed HAD-GNN with three of its variations for τ-day-ahead movement and return rate predictions.

- The HAD-GNN-SG uses one static stock co-movement graph instead of dynamic graphs in the HAD-GNN. Specifically, the HAD-GNN-SG builds one MSCMG/PSCMG whose observation time window is the entire training period (i.e.,

$\Delta t = T_R$).

- The HAD-GNN-RG is based on a series of dynamic random stock graphs without considering stock price correlations. At each time step, the model randomly chooses three stocks as neighbors of each stock.
- The HAD-GNN-CW simplifies the HAD-GNN by replacing the node-level graph attention mechanism with a simple weighting strategy based on the price co-movement. For movement predictions on the MSCMG, this variation replaces the node-level attention n_{ij} with a normalized weight that is inversely proportional to d^{ij}, the Manhattan distance between stocks v_i and v_j. For return rate predictions on the PSCMG, n_{ij} is replaced with a normalized weight proportional to the Pearson correlation coefficients ρ^{ij} between stocks v_i and v_j. In other words, the influence from neighboring stocks is directly based on the historical price co-movement instead of being learned via an attention mechanism.

Table 6. Performance of the HAD-GNN vs its variations on τ-day-ahead movement predictions for S&P 500 stocks.

τ	1		5		9	
Evaluation metrics	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
HAD-GNN-CW	37.28**	29.17**	35.11**	29.95**	35.15*	30.27**
HAD-GNN-SG	36.13***	25.43***	34.24***	26.41***	34.30**	24.91***
HAD-GNN-RG	34.55***	24.86***	34.07***	29.06***	33.32**	25.26***
HAD-GNN	**39.52**	**32.91**	**36.04**	**33.20**	**35.85**	**32.05**

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

Table 7. Performance of the HAD-GNN vs. its variations on τ-day-ahead return rate predictions for S&P 500 stocks.

τ	1	5	9
Evaluation metrics	MAE	MAE	MAE
HAD-GNN-CW	1.077E-2*	2.686E-2**	3.589E-2**
HAD-GNN-SG	1.089E-2*	2.719E-2**	3.603E-2**
HAD-GNN-RG	1.093E-2**	2.792E-2***	3.595E-2**
HAD-GNN	**1.061E-2**	**2.508E-2**	**3.429E-2**

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

Tables 6 and 7 reveal that using random stock graphs, such as in the HAD-GNN-RG, yields the worst performance, highlighting the value of predictive signals from stock co-movement graphs. In addition, methods based on dynamic co-movement graphs (the HAD-GNN and HAD-GNN-CW) perform better than the HAD-GNN-SG based on a single and static co-movement graph. This result demonstrates that dynamics in stock co-movement should be captured for better predictive performance.

In addition, the fact that the HAD-GNN outperforms the HAD-GNN-CW suggests that the node-level graph attention learned from the data works better than the simple weighting heuristics. To further illustrate the discrepancies between the HAD-GNN and HAD-GNN-CW, we calculated Pearson correlation coefficients between learned node-level graph attention coefficients (in the HAD-GNN) and the heuristic weight (as in the HAD-GNN-CW) for each time step during the training period for the S&P 500 dataset. The correlation coefficients change over time (see Fig. D1 in Online Supplement D), with the average values being 0.62 ($SD = 0.03$) and 0.59 ($SD = 0.10$) for movement and return rate predictions, respectively. In other words, graph attention coefficients learned by this framework are only moderately correlated with simple measures of stock price co-movement over time, yet the supervised approach to learning graph attention can provide significantly better predictive performance.

5.4 Sensitivity Analyses

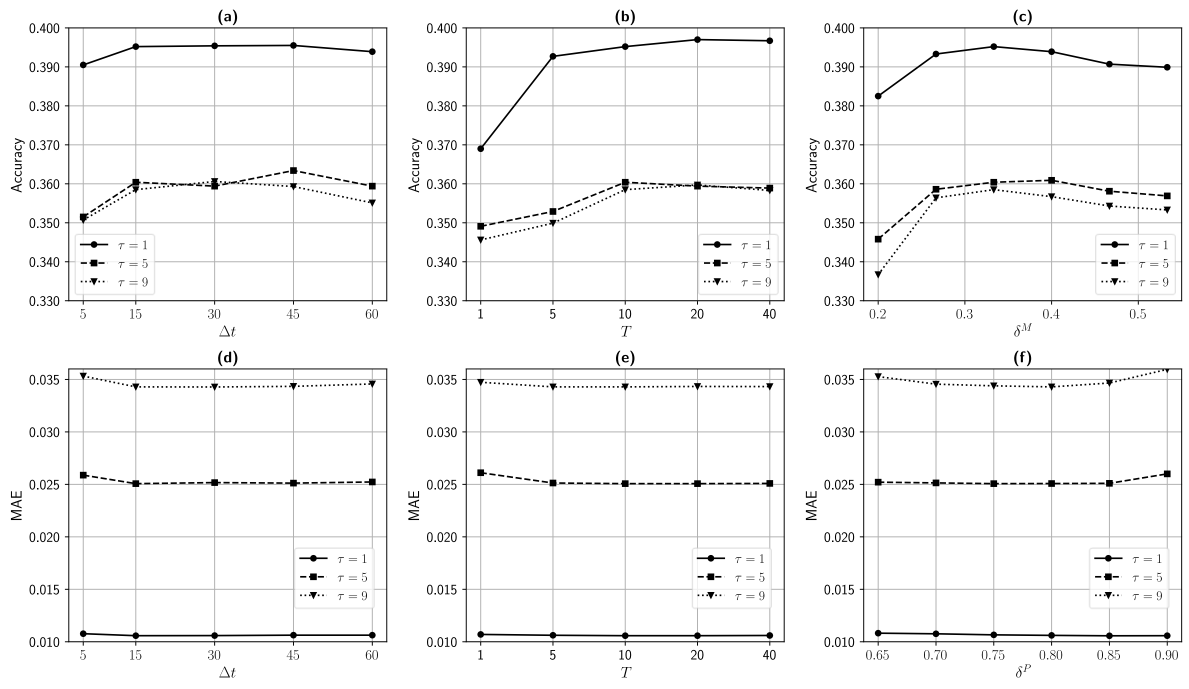

Fig. 5. Effect of nonlearning hyperparameters on stock movement (a-c) and return rate (d-f) predictions.

This subsection evaluates the robustness of the proposed model to changes in the hyperparameters. Each sensitivity experiment changes the value of one target hyperparameter while the others remain the same. Again, we only present the S&P 500 stock results, although the results are consistent for the other two datasets. The proposed model has two types of hyperparameters: nonlearning parameters involved in building stock graphs and formulating predictions and learning-related parameters related to predictive model training.

Figure 5 reports the effects of three nonlearning hyperparameters. The Δt values in panels (a) and (c) represent the sizes of the observation time window to construct the MSCMG and PSCMG, respectively. In panels (b) and (e), T is the temporal length of the historical features for the prediction model. In panels (c) and (f), δ^M and δ^P are the edge weight threshold values for the MSCMG and PSCMG, respectively. For Δt and T, which measure how far we should look back in time, lower values (e.g., Δt = 5 and T = 1 or 5) lead to lower performance, likely because there is insufficient historical data from which to learn. However, once Δt moves beyond 15 and T is above 10, further increases in the values do not improve or even hurt the performance. In other words, including historical information from too long ago is unnecessary for stock predictions in the very near future (e.g., τ = 1, 5, or 9 days). Similarly, too large or too small values of δ^M and δ^P lead to worse performance. When the thresholds are too high, the resulting stock graphs are very sparse with few edges, and the proposed model may lose potentially valuable information from moderate co-movement patterns. In contrast, lower thresholds mean many edges between stocks and a higher noise level from such stock graphs. Online Supplement D provides further discussion on how to choose values for these nonlearning parameters.

The proposed model is less sensitive to learning-related hyperparameters. Figures 6(b) and 6(e) demonstrate that the proposed model is insensitive to different learning rates. The performance increases slightly when the mini-batch size increases from 1 but becomes stable after the size becomes greater than 10 (panels (b) and (e)). Panels (c) and (f) suggest that the dimensionality of hidden layers does not need to be more than 32. More details on learning-related hyperparameters are included in Online Supplement D.

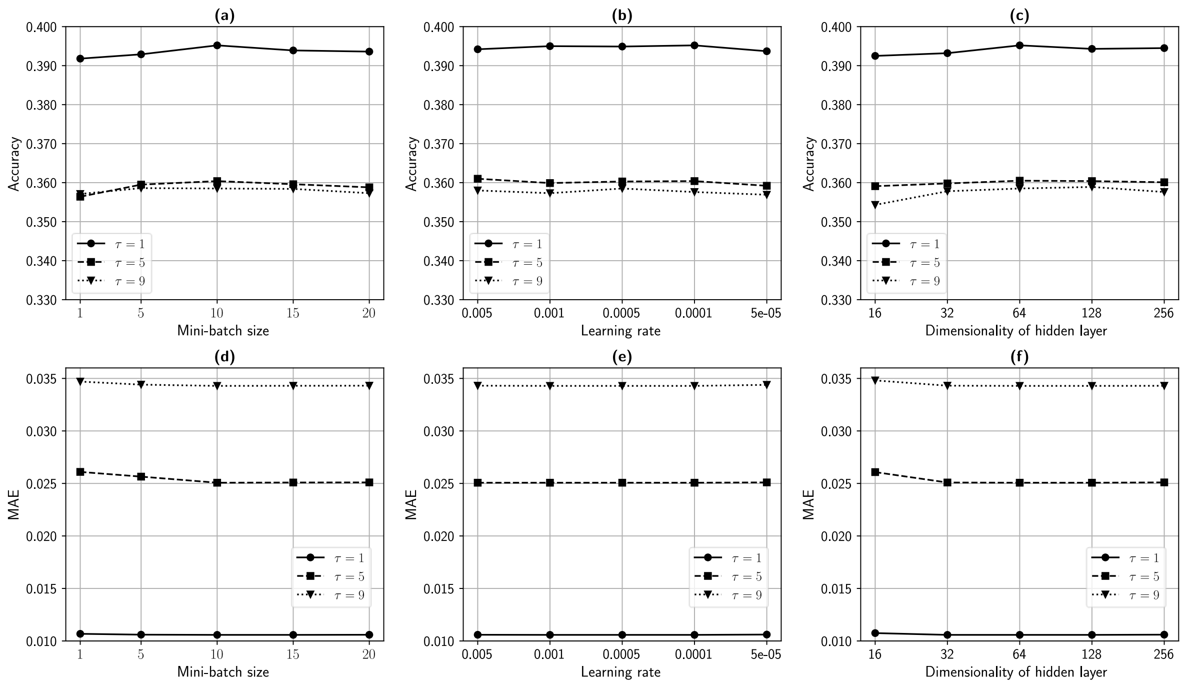

Fig. 6. Effect of learning-related hyperparameters on stock movement (a-c) and return rate (d-f) predictions.

In practice, HAD-GNN can be retrained over time to get updated parameters and adapt to everchanging market conditions. Therefore, Online Supplement D illustrates how retraining our model with a rolling window scheme can further improve the performance of HAD-GNN, and how users of HAD-GNN can make decisions on the frequency of model updating.

5.5 Efficiency of the t-Batch Training Mechanism

As discussed in Section 4.3, we adopted the mini-batch gradient descent in the t-BTM for inductive representation learning on dynamic graphs because computational efficiency is an important issue, especially for highly dynamic stock co-movement graphs. Thus, we compared the running time of the HAD-GNN with various mini-batch sizes. When the mini-batch size is 1, the HAD-GNN essentially becomes HAD-GNN with stochastic gradient descent. We excluded the batch gradient descent from the comparison because it requires a tremendous amount of memory. Each experiment ran 100 epochs with the same parameters on the same hardware (80 Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20 GHz, Tesla V100 SXM2 with 16 GB). Figure 7 depicts the average running time of one epoch. For one-day-ahead movement predictions, t-BTM makes the HAD-GNN 4.2 to 11.7 times faster than the stochastic gradient descent with a mini-batch size of 5 to 30. For one-day-ahead return rate predictions, the t-BTM is 4.2 to 14.0 times faster than the stochastic gradient descent. Overall, the t-BTM represents a more efficient approach for inductive graph representation learning and can scale to highly dynamic graphs.

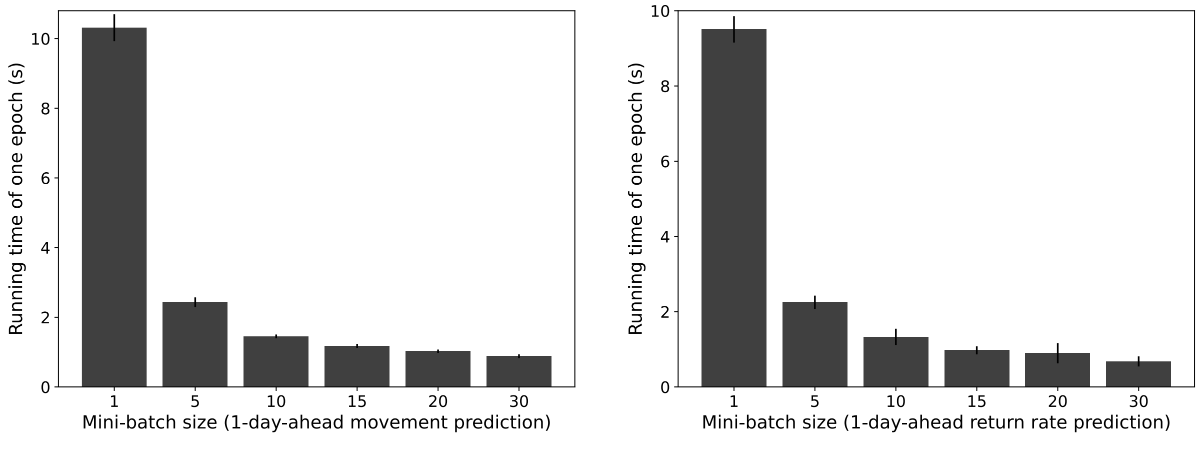

Fig. 7. Running time for one epoch with various mini-batch sizes in the t-BTM.

5.6 Trading Simulations

Besides performance evaluations using metrics widely adopted in machine learning research, this subsection demonstrates the practical value of the proposed model via trading simulations on stocks in the S&P 500. Admittedly, simulating stock trading is challenging because real-world practices involve many factors beyond stock prices, including volatility, transaction costs, portfolio management, and other factors. However, as the predictive model focuses on stock price movements, the simulations adopt a widely used and simple strategy in the literature of stock predictions: buying a stock predicted to rise in price with the highest probability (Feng et al. 2019).

For each stock, the model provides three probabilities for classes +1, 0, and -1. For each stock that is predicted as class +1 (i.e., the probability of being class +1 is greater than the probability for the other two classes), its buying score is the ratio between its predicted probability of being class +1 (increasing) and its predicted probability of being class -1 (decreasing). A higher buying score means a higher chance of rising prices compared to other stocks. In the simulation, a trader equally splits the budget to buy stocks with the top-20 buying scores from stocks predicted to be in the +1 class by τ-day-ahead movement predictions with the HAD-GNN and then holds for τ trading days. Such a rolling horizon allows the trader to adjust the portfolio. Besides equal splitting of the budget, we also tested another strategy to allocate an investment based on the market cap of stocks[3] (with the suffix "cap") (Tütüncü and Koenig 2004). Transaction costs and short selling are not considered. An example of simulated trading actions using this strategy is presented in Online Supplement E.

For comparison, the simulations also included GAT, the best performer among the benchmark methods, and Markowitz efficient frontier (MKF) a classic portfolio optimization

[3] Monthly market capitalization data for stocks was collected from https://companiesmarketcap.com/.

method (Markowitz 1968; details in Online Supplement E). In practice, a sparse Markowitz portfolio that selects a small number of assets from an asset pool is a widely used method to control management fees. In this paper, we created a sparse Markowitz portfolio with 20 stocks following the method proposed by Bertsimas and Cory-Wright (2018). They imposed a ridge regularization term to reformulate the sparse portfolio optimization problem as a convex binary optimization problem, solvable via an efficient outer-approximation procedure.

Investment performance was evaluated with the standard return ratio and Sharpe ratio (Sharpe 1966). The Sharpe ratio is a risk-adjusted return measure, and we used the two-year US Treasury Note as the risk-free return rate. We also added the market index of the S&P 500 as a benchmark.

Table 8 compares performance measures for different trading strategies at the end of the testing period, and Fig. E2 (Online Supplement E) illustrates how the return ratios from different strategies change over time. Strategies based on the HAD-GNN consistently achieve the best performance on return measures across different rolling horizons (τ = 1, 5, and 9).

Table 8. Simulated investment performance of different trading strategies at the end of the testing period.

τ	1		5		9	
Method	Return ratio	Sharpe ratio	Return ratio	Sharpe ratio	Return ratio	Sharpe ratio
S&P 500 index	0.110	0.026	0.093	0.063	0.110	0.094
MKF	0.098	0.024	0.113	0.086	0.127	0.118
GAT	0.140	0.032	0.111	0.056	0.127	0.100
GAT-cap	0.147	0.030	0.128	0.070	0.126	0.090
HAD-GNN	0.162	0.032	**0.414**	**0.174**	0.214	**0.140**
HAD-GNN-cap	**0.262**	**0.052**	0.319	0.131	**0.219**	0.132

Overall, even though the trading simulations are only simplications of real-world trading scenarios and omit many other important factors (see Online Suppplement E for additional analyses on volatility), the results are still auspicious. The improvements over the market index, a traditional portfolio optimization method, and another state-of-the-art deep learning predictive model support the potential value of the proposed model in real-world stock trading.

6. Conclusions

This paper proposed the HAD-GNN, an inductive graph representation learning approach on dynamic stock graphs for stock prediction. Based on historical stock price co-movement patterns, we constructed co-movement graphs to represent relationships between stocks. Our model design incorporates weight signals from neighboring stocks and historical time steps to enable learning and inference on such dynamic stock co-movement graphs. More importantly, it is an inductive method that can scale to dynamic graphs whose structures frequently change over time. We extended the mini-batch gradient descent to inductive representation learning to improve the training efficiency of the HAD-GNN model on dynamic stock graphs. Extensive experiments on real-world datasets from three stock markets demonstrated that the proposed model provides significant improvements in predicting stock movements and return rates.

The three major components of the HAD-GNN (the hybrid-attention encoder, dynamic stock co-movement graphs, and t-BTM) were further evaluated via comparisons with benchmark methods, ablation analyses, and computational efficiency analyses. Besides its dominant performance on traditional performance metrics for predictive modeling, the proposed model also prevails in trading simulations. The model leads to much higher returns than a group of benchmark methods and has the potential to aid in real-world trading.

The HAD-GNN can extract task-specific latent features for financial tasks, such as building investment portfolios and predicting risks. Moreover, with hybrid attention from both the node and temporal levels, the HAD-GNN represents a novel, generalizable, and scalable inductive framework to learn from dynamic graphs. Beyond stock graphs in this paper, many social networks also feature dynamic structures with node behaviors or attributes that change over time. The HAD-GNN can be used for learning from and for inference for such a combination of dynamic graphs and time sequence data (e.g., to predict whether an individual will share a piece of information in an information diffusion network or predict whether a customer will buy a new product in a viral marketing campaign).

There are also interesting directions for future research. For instance, we did not explicitly model temporal dependencies between the stock co-movement graphs for consecutive time steps in this work. We conjecture that incorporating the relationships between stock co-movement graphs using graph RNNs (Wu et al. 2019) may improve predictions. In addition, the proposed model only used financial features related to stock prices, whereas research has identified other sources of information beyond financial data, such as news stories (Mao and Wei 2016) and user-generated content in social media (Bali et al. 2018). It would be interesting to integrate such information into dynamic co-movement graphs to generate more expressive vector representations for financial predictions.

7. Acknowledgments

This work was supported the Ministry of Health of China under Grant No. 2017ZX10303401-002, the Ministry of Science and Technology of China under Grant No. 2020AAA0108401, and the Natural Science Foundation of China under Grant Nos: 71472175, 71602184, 71621002, and 71991462. Note that Hu Tian and Xiaolong Zheng have an equal contribution, and Xiaolong Zheng is the corresponding author of this article.

References

Agarwal A, Leung ACM, Konana P, Kumar A (2017) Cosearch attention and stock return predictability in supply chains. *Inf. Sys. Res.* 28(2):265–288.

Aggarwal CC, Hinneburg A, Keim DA (2001) On the surprising behavior of distance metrics in high dimensional space. *Int. Conf. Database Theory.*, 420–434.

Bali TG, Bodnaruk A, Scherbina A, Tang Y (2018) Unusual news flow and the cross section of stock returns. *Manag. Sci.* 64(9):4137–4155.

Ban GY, El Karoui N, Lim AE (2016) Machine learning and portfolio optimization. *Manag. Sci.* 64(3):1136–1154.

Bertsimas D, Cory-Wright R (2018) A scalable algorithm for sparse portfolio selection. *arXiv preprint arXiv:1811.00138*.

Bonanno G, Lillo F, Mantegna R, et al. (2001) High-frequency cross-correlation in a set of stocks. *Quantitative Finance* 1(1):96–104.

Brown RG (2004) *Smoothing, forecasting and prediction of discrete time series* (Courier Corporation).

Bruna J, Zaremba W, Szlam A, Lecun Y (2014) Spectral networks and locally connected networks on graphs. *Int. Conf. Learning Representations (ICLR2014), CBLS, April 2014*. http–openreview.

Campbell JY, Mackinlay AC, Lo AW, Press PU (1997) The econometrics of financial markets. *J. Empirical Finance* 3(1):15–102.

Cao S, Lu W, Xu Q (2016) Deep neural networks for learning graph representations. *30th AAAI Conf. Artificial Intelligence*.

Chan LKC, Lakonishok J, Swaminathan B (2007) Industry classifications and return comovement. *Financial Analysts J.* 63(6):56–70.

Chatfield C, Xing H (2019) *The analysis of time series: an introduction with R* (CRC Press).

Chi KT, Liu J, Lau FC (2010) A network perspective of the stock market. *J. Empirical Finance* 17(4):659–667.

Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. *Adv. Neural Inf. Process. Sys*. 3844–3852.

Dhar V, Geva T, Oestreicher-Singer G, Sundararajan A (2014) Prediction in economic networks. *Inf. Sys. Res.* 25(2):264–284.

Ding X, Zhang Y, Liu T, Duan J (2015) Deep learning for event-driven stock prediction. *24th Int. Joint Conf. Artificial Intelligence*.

Ding X, Zhang Y, Liu T, Duan J (2016) Knowledge-driven event embedding for stock prediction. *Proc. COLING 2016, 26th Int. Conf. Computational Linguistics: Technical Papers*. 2133–2142.

Ehling P, Heyerdahl-Larsen C (2017) Correlations. *Manag. Sci.* 63(6):1919–1937.

Elfeky MG, Aref WG, Elmagarmid AK (2005) Periodicity detection in time series databases. *IEEE Trans. Knowl. Data Eng.* 17(7):875–887.

Elliott RJ, Van Der Hoek* J, Malcolm WP (2005) Pairs trading. *Quantitative Finance* 5(3):271–276.

Fan X, Li B, Li C, SIsson S, Chen L (2019) Scalable deep generative relational model with high-order node dependence. *Adv. Neural Inf. Process. Sys*. 12658–12668.

Feng F, He X, Wang X, Luo C, Liu Y, Chua TS (2019) Temporal relational ranking for stock prediction. *ACM Trans. Inf. Sys. (TOIS)* 37(2):27.

Goodfellow I, Bengio Y, Courville A, Bengio Y (2016) *Deep learning* (MIT Press, Cambridge).

Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. (ACM), 855–864.

Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Systems*. 1024–1034.

Hoberg G, Phillips G (2016) Text-based network industries and endogenous product differentiation. *J. Political Econ.* 124(5):1423–1465.

Hu Z, Liu W, Bian J, Liu X, Liu TY (2018) Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. *Proc. 11th ACM Int. Conf. Web Search and Data Mining*.

(ACM), 261–269.

Hudson RS, Gregoriou A (2015) Calculating and comparing security returns is harder than you think: A comparison between logarithmic and simple returns. *Int. Rev. Financial Anal.* 38:151–162.

Kim D, Qi Y (2010) Accruals quality, stock returns, and macroeconomic conditions. *Accounting Rev.* 85(3):937–978.

Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. *Int. Conf. Learning Representations*.

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. *Int. Conf. Learning Representations (ICLR)*.

Kock CJ (2005) When the market misleads: Stock prices, firm behavior, and industry evolution. *Organ. Sci.* 16(6):637–660.

Kraus M, Feuerriegel S (2017) Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Sys.* 104:38–48.

Krawczyk B (2016) Learning from imbalanced data: open challenges and future directions. *Progress Artif. Intell*. 5(4):221–232.

Kumar S, Zhang X, Leskovec J (2019) Predicting dynamic embedding trajectory in temporal interaction networks. *Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*. 1269–1278.

Kwan SH (1996) Firm-specific information and the correlation between individual stocks and bonds. *J. Financial Econ.* 40(1):63–80.

Li Q, Chen Y, Jiang LL, Li P, Chen H (2016) A tensor-based information framework for predicting the stock market. *ACM Trans. Inf. Sys. (TOIS)* 34(2):11.

Li X, Chen K, Sun SX, Fung T, Wang H, Zeng DD (2016) A commonsense knowledge-enabled textual analysis approach for financial market surveillance. *INFORMS J. Comput.* 28(2):278–294.

Maas AL, Hannun AY, Ng AY (2013) Rectifier nonlinearities improve neural network acoustic models. *Proc. ICML*. 3.

Mao MQ, Wei KCJ (2016) Cash-flow news and the investment effect in the cross section of stock returns. *Manag. Sci.* 62(9):2504–2519.

Markowitz H (1968) Portfolio selection. *Yale university press*.

Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. *ICML*.

Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. (ACM), 701–710.

Qin Y, Song D, Cheng H, Cheng W, Jiang G, Cottrell GW (2017) A dual-stage attention-based recurrent neural network for time series prediction. *Proc. 26th Int. Joint Conf. Artificial Intelligence*. (AAAI Press), 2627–2633.

Raffel C, Ellis DP (2015) Feed-forward networks with attention can solve some long-term memory problems. *ArXiv preprint arXiv:1512.08756*.

Rios, J., Zhao, K., Street, W. N., & Blackhurst, J. (2020). Stock Price Movement Cross-Predictability in Supply Chain Networks (SSRN Scholarly Paper ID 3737658). Social Science Research Network. https://doi.org/10.2139/ssrn.3737658

Rosenberg B, Rudd A (1982) Factor-related and specific returns of common stocks: Serial correlation and market inefficiency. *J. Finance* 37(2):543–554.

Ruder S (2016) An overview of gradient descent optimization algorithms. *ArXiv preprint arXiv:1609.04747*.

Schwert GW (1989) Why does stock market volatility change over time? *J. Finance* 44(5):1115–1153.

Sharpe WF (1966) Mutual fund performance. *J. Business* 39(1):119–138.

Si J, Mukherjee A, Liu B, Pan SJ, Li Q, Li H (2014) Exploiting social relations and sentiment for stock prediction. *Proc. 2014 Conf. Empirical Methods in Natural Language Processing*. 1139–1145.

Singleton JC, Wingender J (1986) Skewness persistence in common stock returns. *J. Financial Quantitative Anal.,* 335–341.

Tian H, Zheng X, Zeng DD (2019) Analyzing the dynamic sectoral influence in Chinese and American stock markets. *Physica A: Statistical Mech. Appl.,* 120922.

Tütüncü RH, Koenig M (2004) Robust asset allocation. *Ann. Oper. Res.* 132(1):157–187.

Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. *Int. Conf. Learning Representations*.

Wu H, Zhang W, Shen W, Wang J (2018) Hybrid deep sequential modeling for social text-driven stock prediction. *Proc. 27th ACM Int. Conf. Information and Knowledge Management*. (ACM), 1627–1630.

Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2019) A comprehensive survey on graph neural networks. *arXiv:1901.00596 [cs, stat]*.

Xie J, Qiu Z (2007) The effect of imbalanced data sets on LDA: A theoretical and empirical analysis. *Pattern Recognit.* 40(2):557–562.

Xu B, Wang N, Chen T, Li M (2015) Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.

Xu Y, Cohen SB (2018) Stock movement prediction from tweets and historical prices. *Proc. 56th Annual Meeting Assoc. for Computational Linguistics (Vol. 1: Long Papers)*. 1970–1979.

Yang Y, Padmanabhan B, Liu H, Wang X (2012) Discovery of periodic patterns in sequence data: a variance-based approach. *INFORMS J. Comput.* 24(3):372–386.

You J, Ying R, Ren X, Hamilton WL, Leskovec J (2018) GraphRNN: Generating realistic graphs with deep auto-regressive models. *ICML*. 5694–5703.

Zhang L, Aggarwal C, Qi GJ (2017) Stock price prediction via discovering multi-frequency trading patterns. *Proc. 23rd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. (ACM), 2141–2149.

Zhang X, Zou Y, Shi W (2017) Dilated convolution neural network with LeakyReLU for environmental sound classification. 2017 *22nd Int. Conf. Digital Signal Processing*. (IEEE), 1–5.

Zhao K, Yen J, Ngamassi LM, Maitland C, Tapia AH (2012) Simulating inter-organizational collaboration network: a multi-relational and event-based approach. *Simulation* 88(5):617–633.