Contents lists available at ScienceDirect

Information Fusion

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

Full length article

Attentive gated graph sequence neural network-based time-series information fusion for financial trading

Wei-Chia Huang [a], Chiao-Ting Chen [b], Chi Lee [c], Fan-Hsuan Kuo [c], Szu-Hao Huang [c,*]

[a] *Institute of Information Management, National Yang Ming Chiao Tung University, 1001 University Road, Hsinchu 30010, Taiwan*
[b] *Department of Computer Science, National Yang Ming Chiao Tung University, 1001 University Road, Hsinchu 30010, Taiwan*
[c] *Department of Information Management and Finance, National Yang Ming Chiao Tung University, 1001 University Road, Hsinchu 30010, Taiwan*

ARTICLE INFO

Keywords:
Graph neural network
Relational network
Robo-advisor

ABSTRACT

With the advances in financial technology (FinTech) in recent years, the finance industry has attempted to enhance the efficiency of their services through technology. The financial service most commonly desired by people is a robo-advisor, which is a virtual expert that advises people on how to make good decisions related to financial market trading. Some mathematical models may be difficult to apply to prediction problems involving multiple cross correlation. To overcome this issue, the rules of and implicit correlation between the collected data must be determined through algorithms to forecast the future market situation. By combining financial information from the financial market, we apply an information fusion approach to collect data. Also, a relational model can be developed using a deep neural network, which can be represented using a graph structure, to determine the real market situation. However, the structural information of the graph may be lost if a traditional deep learning module is used. Therefore, in this paper, we present a visual-question-answering-like deep learning fusion model based on the graph structure and attention mechanism. This model was used to study the interaction between the time-series data of various financial variables. The relationships among financial commodities were formulated, and the graph representation was learned through graph neural networks. Moreover, the importance of each commodity was determined through the attention mechanism. The proposed method improved the accuracy of trend and volatility prediction up to 6% on S&P500 and developed a pairs trading application.

1. Introduction

The primary goals of financial operations include making additional money, controlling or lowering risks, and creating trust. Therefore, an increasing number of financial companies and banks have begun to introduce deep learning into their services or systems. The advanced technology used to enhance financial services is termed financial technology (FinTech). In the field of FinTech, the most common application is to use historical records for predicting future prices. Such prediction can facilitate appropriate asset allocation and profit maximization.

Historical prices are a type of time-series data; therefore, many people have begun to use machine learning for time-series value prediction. However, this complex market is affected by a large number of events in which forecasting of future market dynamics is challenging. Investor to the stock market may require researching various relevant factors and extracting useful information for reliable forecasting. Therefore, fusion can be viewed as a method of integrating data or features and based on combined methods that can help each other. we briefly categorize fusion into information fusion, feature fusion, and model fusion.

According to [1], support vector machine, which is a machine learning method, can be used to estimate future closing prices because these prices are a complicated result of market conditions, which can be influenced by many factors. Information provided by the media and service suppliers indicates that the most popular service desired by customers is a robo-advisor, which is a virtual expert that advises people on how to make good decisions related to financial market trading, including the selection of strategies, operations, or portfolios, for maximizing customer profit. In addition to the aforementioned advantage, robo-advisors provide digital financial advice based on mathematical rules or algorithms; thus, robo-advisors can provide 24/7 service without any human intervention.

The performance of a robo-advisor depends on the modeling of the cross correlation between different commodities. However, various

* Corresponding author.
E-mail addresses: tyc30827.iim06g@nctu.edu.tw (W.-C. Huang), rsps971130.cs09@nycu.edu.tw (C.-T. Chen), richielee1030.dif06@nctu.edu.tw (C. Lee), shirley0509.hs06@nycu.edu.tw (F.-H. Kuo), szuhaohuang@nycu.edu.tw (S.-H. Huang).

https://doi.org/10.1016/j.inffus.2022.10.006
Received 18 March 2022; Received in revised form 20 September 2022; Accepted 7 October 2022
Available online 13 October 2022

definitions of correlation exist, including mathematical and statistical definitions. Moreover, some mathematical models are invalid or difficult to apply in the case of problems involving multiple cross correlation. Graphs have a strong ability to represent relationships between different factors. In investigations of the interaction relationship embedded in financial time-series data, traditional deep learning models perform well on data in the Euclidean space, such as images and sequences, but cannot completely retain the structural information of graphs. Therefore, poor training results are obtained with these models for graph data. Consequently, a specialized deep model must be designed for processing the structure of graphs. Therefore, in this paper, we propose a novel robo-advisor prototype with a deep neural network framework and visual question answering (VQA) system.

Therefore, in this paper, we propose a novel robo-advisor prototype with a deep neural network framework and visual question answering (VQA)-system-like form. A VQA system, which is combined with CNN and RNN, is a model that observes images and answers questions related to the images by analyzing the relationship of items in the images. In our case, if we provide historical financial data, such as price data, over a certain period to the robo-advisor and ask it some questions about these data, it analyzes and learns the implied meaning or correlation between the data with algorithms or networks designed in the system. Then, the robo-advisor summarizes the prediction result for the future market condition and can eventually answer the questions or queries input by users. With the addition of the VQA-based system, our model can learn to recognize the pattern of the correlation between multiple financial time-series sequence data and can engage in various trading strategies or assist users in making decisions. Consequently, users can perform risk management with greater ease and convenience. Furthermore, since those data were raw data, and were collected separately, they could also defined as data fusion.

The proposed model learns the cross correlation between different commodities, which is represented as a complex relation network. Therefore, we used two novel deep learning models in this study: the improved relation network and attention-based gated graph sequence neural network (GGSNN). The improved relation network was used as the benchmark model for the GNN. By using the aforementioned deep learning networks, we avoided the necessity of developing a complex mathematical model and automatically mapped features into a high-level representation by executing a nonlinear transform with a large amount of training data. We conducted a series of experiments to evaluate the performance of the aforementioned models and demonstrated a financial application, namely pairs trading with the VQA-like system.

The proposed model can be used for market condition forecasting, and the obtained results can be used in an experiment on pairs trading. The major contributions of this paper are as follows:

- A VQA system is proposed to answer questions about the future market condition automatically by using keywords in the questions to form the question embedding. The question set of a model can be extended and jointly trained with a model based on sequential learning in the natural language domain. For risk management and profit maximization, we use financial indicators such as the company scale, EPS, and dividends as the basis for judgment.
- The performance of reasoning tasks is improved and the relation model is set as a benchmark for time-series tasks by conducting one-dimensional (1D) and two-dimensional (2D) convolution. The relations extracted from the convolution layer are tagged with relatively temporal annotation to learn the temporal relation.
- An improved gated GNN with a novel multihead graph attention mechanism is applied to the edges and nodes of a graph. This network highlights the information of important nodes and the weighted aggregate of the messages from neighbor nodes. A gated recurrent unit (GRU)-like updater is used to learn the state of the graph.

The remainder of this paper is organized as follows. Section 2 describes the related work; Section 3 describes the Attentive GGSNN and deep relation network (DRN); Section 4 presents the experiment results; and Section 5 summarizes the conclusion and future research directions.

2. Related works

This section presents a review of studies on two topics: deep learning for modeling time-series relationships and graph neural networks.

2.1. Deep learning for modeling time-series relationships

Obtaining the correlation among multiple time series is difficult. Most mathematical theories can determine the correlation between only two time series. Therefore, most studies on the correlation among multiple time series have adopted machine learning algorithms [2–4], which have certain advantages. If one possesses a sufficiently large quantity of data, superior analysis results can be achieved using deep learning methods because these methods enable automatic feature engineering through neural networks. Wang et al. [5] invented an end-to-end joint self-attention model to make house prediction more precisely. Huang et al. [6] designed a novel sampling strategy to reduce the immense computational resources required for RL. Jang et al. [7] proposed a new option pricing and delta-hedging framework based on deep learning. Moews et al. [8] proposed the use of deep neural networks (DNNs) that apply stepwise linear regressions with exponential smoothing in preparatory feature engineering. In these networks, regression slopes are used as trend strength indicators for a given time interval. Chen et al. [9] presented a multimodal reinforcement trading system which includes a deep neural network and multimodal deep recurrent neural network. Li et al. [10] produces a more refined measurement of the informativeness of peer engagement by training four deep learning models for sentiment classification and conducting comparative evaluations. Li et al. [11] combined the traditional ARIMA model with a deep learning model for forecasting high-frequency financial time series. Bao et al. [12] used deep learning modules, such as stacked autoencoders and long–short-term memory (LSTM) modules, for financial time-series prediction. Sagheer et al. [13] use deep LSTM approach to deal with the limitation of traditional method and get more accurate results. Fischer and Krauss [14] used LSTM networks to predict out-of-sample directional movements for the constituent stocks of the S&P 500 from 1992 until 2015. They reported a common pattern in trading. Because of the introduction of electronic trading and the automation that followed, the trading volume has increased sharply, and trading involves large-scale time-series data. To address this problem, Tsantekidis et al. [15] used an RNN-based deep learning model to predict price movements from large-scale high-frequency time-series data. Mo and Wang [16] used a DNN with a custom loss function to forecast long-term cross correlations between two time-series sequences. However, when solving the problem of portfolio construction, deep learning methods do not consider the relationship between assets under inspection. Thus, Yun et al. [17] proposed a two-stage deep learning framework, namely the Grouped-ETFs Model (GEM), for learning interasset and group features at each stage. In addition, traditional investment and portfolio theories have limitations regarding decision-making and portfolio construction. In response to this problem, Vo et al. [18] introduced a Deep Responsible Investment Portfolio (DRIP) model that contains a multivariate bidirectional LSTM module to predict stock returns for superior portfolio construction. Peng and Jiang [19] predicted the future movement of financial market prices by using a DNN and correlation graph that indicates the relationships between companies. A graph neural network (GNN) is suitable for modeling the relationships between different features [20] because it considers the relationship between different objects. The proposed model is based on a variant of the GNN structure. The following section introduces GNNs and their variants.

2.2. Graph neural network

GNNs are deep-learning-based networks that operate in the graph domain. A graph is a type of data structure that describes a set of objects (nodes) and the relations (edge) between these objects. Thus, the structural information of data can be modeled into a graph, and this information can then be used to do more than what traditional deep learning neural network modules can do. Because a graph is a locally connected structure, shared-weight parameters can reduce the computational cost, and a multilayer network can hierarchically reflect the pattern of a graph's structure. However, the application of a traditional deep learning module often causes loss of the structural information of a graph. Consequently, graph neural networks have been developed [20]. Over the past 20 years, researchers have proposed learning-based networks [21,22] for solving time-series-based tasks by using graphs. Chiang et al. [23] proposed Cluster-GCN, which is a GCN algorithm suitable for SGD-based training and is based on exploitation of graph clustering structure. Kapoor et al. [24] adopted a GNN and mobility data for COVID-19 case prediction. In this GNN, the nodes represent human mobility, the spatial edges represent the human-mobility-based inter-region connectivity, and the temporal edges represent node features through time. Chen et al. [25] proposed a MV-GAN model that enriches user and product semantics by conducting metapath-guided neighbor aggregation and multiview fusion on heterogeneous travel product recommendation graphs. Liu et al. [26] proposed a method that combines a recurrent neural network (RNN) and GNN, which is called recurrent GNN, to predict the next-period prescription. By using deep learning to model the complex dependency inside a graph, one can aggregate the information within the graphical data and then obtain a nonlinear representation. The following sections describe the variants of GNNs.

Convolution on a graph structure is an extended version of the traditional convolution operation. A typical convolution layer can only operate on data in the Euclidean space; therefore, many spectral and spatial approaches have been developed for aggregating neighborhood information. GraphSAGE [27] is a major advancement in GNNs. It first selects an aggregation function, such as LSTM or a pooling aggregator, to collect embedding information on the neighborhood by using the neural network module. A graph convolution network (GCN) has been applied in many applications. For example, Nie et al. [28] proposed a model called M-GCN to solve the 2D-image-based three-dimensional model retrieval problem. Wang et al. [29] proposed the MSF-GCN, which can balance information from a graph's structure and node attribute features during the aggregation step. Moreover, Cui et al. [30] used a deep learning framework called TGC-LSTM to learn a traffic network. Liang et al. [31] proposed a graph convolutional network based on SenticNet, namely Sentic GCN, to leverage the affective dependencies of a sentence in accordance with a specific aspect.

A gated GNN is a GNN variant in which a modified graph updater is used to learn a convergence result of node embedding by updating the information collected from the aggregator. A GGSNN [32] uses a GRU as the node state updater of GNNs, unrolls the recurrence for a fixed number of steps (T), and uses the back propagation through time (BPTT) algorithm to compute gradients.

The attention mechanism was developed for application in the CV domain and was subsequently extended to NLP tasks, such as machine translation. This mechanism can highlight the relationship between context and the target word in a sentence. The attention mechanism can also be applied to the task-relevant parts of a graph. Velikovi et al. [33] used a multi-head self-attention mechanism to calculate the alignment score of pairwise nodes. A heterogeneous graph, such as a knowledge graph, is a multi-relational graph with multiple edge types. Shang et al. [34] focused on graph-level embedding and proposed a graph-focused attention mechanism for conducting graph regression. The aforementioned studies indicate that the attention mechanism can be appropriately used in graph-mining tasks.

2.3. Relation networks

Relation networks are a series of networks released by Google DeepMind for solving relationship modeling problems, and these networks can be considered GNN variants. The relation network proposed by Santoro et al. [35] is a DNN module that contains a structure used for spatial relational reasoning. Relational reasoning is a process in which an attempt is made to identify whether a meaningful pattern exists inside an information flow or a data transformation. Battaglia et al. [36] proposed an interaction network that can reason how objects in complex systems interact with each other. A temporal relation network [37] is a relation network [35] extended to activity recognition. Gao et al. [38] use an architecture called relational triplet network (RTN) to capture semantic patterns in heterogeneous networks. Zheng et al. [39] proposed a relation network that takes advantage of meta-learning to solve the problem of few-shot caricature face recognition. Deng et al. [40] invent a geometry-attentive relational reasoning approach to address the problem of facial landmark detection. As for the prediction of financial markets, Cheong et al. [41] use a model called STCNN-RN to find the abnormal situations with some outlier points. By applying an interpretability model, they can help investors check what causes these anomalies.

The aforementioned studies indicate that GNNs can formulate relation information with the help of a graph structure. In this study, in addition to using a GNN, the internal information of a graph was obtained for different situations in the financial market by adopting the attention mechanism.

3. Attentive GGSNN

In this section, we first talk about the problem statement in our proposed model. After, we introduce the concept of gated graph neural networks (GGNNs). We then provide an overview of our proposed method, which is an improved GGNN. We finally introduce all the ideas and variants in our model. Moreover, we use relation networks as feature embedding modules to create embeddings. We consider these networks as benchmark models for a GGSNN. The final section introduces the concept of relation networks and describes how the modified models can be applied to financial time-series data.

3.1. Problem statement

Our goal is to learn a function $f_\theta(I_{C*T}, Q, A_E) = \hat{A}$ that has the objective of solving multiple categorical classification problems by using the inputs I_{C*T}, Q, and A_E. The parameter θ is the training parameter of the deep learning model. The matrix I_{C*T} consists of the return data of all C companies during time T. The question Q can be a multi-hot array of two targeted companies that must be answered. Furthermore, A_E represents the adjacency matrix of all E edge types, and the concatenation of adjacency matrix for each edge type. The answer $\hat{A}$ represents the predicted value, which is a multiple categorical classification, where $\hat{A} \in R^n$, $n \in [3, 4]$. For instance, if the question Q 'Volatility' denotes the difference in fluctuation between two query companies, and the answer set $\hat{A}$ is ['Same', 'Big', 'small'].

Fig. 1 presents the overall framework of our proposed system. First, the time-series feature is changed to return data, which are used as the return data of the proposed system. Second, the input matrix I_{C*T} is fed into a graph representation and feature extraction module to obtain a high-level feature representation. Third, the embedded representation is input into a relation network, which provides information on different companies. Fourth, the question embedding is concatenated with the output of the relation network, and an attentive GGSNN is used to obtain the final representation. Finally, multiclass classification is performed for each question.

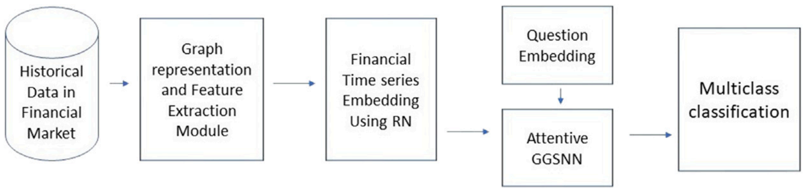

Historical Data in Financial Market
Graph representation and Feature Extraction Module
Financial Time series Embedding Using RN
Question Embedding
Attentive GGSNN
Multiclass classification

Fig. 1. A brief description of our proposed framework.

3.2. Improved GGSNN for analyzing financial time-series data

Researchers aim to develop mathematical models or formulas for accurately evaluating or simulating the conditions of financial markets. However, numerous factors can affect financial markets. Due to the powerful representation capability of graphs, the difficulty in forecasting future market conditions can be reduced by constructing a relational network by mapping the correlation between commodities into a graph structure. The nodes denote companies, and the edges denote the relations between two companies. Multiple-type edges can provide graphs with relation information on various aspects of companies' status.

The aim of this study was to develop a graph-focused model based on the GNN architecture that can learn the correlation between companies and function as a VQA-like system that can send a query about the real market situation. As displayed in Fig. 2, the learning process of the proposed model can be split into two phases. In the first phase, the input matrix of the proposed model comprises the return data of all companies during a specific period. The input matrix is fed into the embedding module and converted into a high-level representation. The features of companies can be converted into an advanced representation or embedding by using a 1D Convolution layer or GRU before constructing a graph. The embedded features are considered to be the node features of the graph structure. After preparing the adjacency matrix, we initialize the hidden state of the GNN with node features and learn the correlation between the nodes in the graph by using an attention-based GGSNN. The second phase is the operation of the output model of the proposed system. The final state of the graph and the embedding matrix of the question are taken as the input of a multilayer feed-forward neural network. The final embedding of our model is then fed into different output layers, and multiclass classification is performed for each question.

The following section describes the proposed method in detail, explains how the proposed method can be improved, and details the application of the proposed model to tasks in the finance domain.

3.3. Propagation model

As presented in the graph propagation model of the GGSNN, the learning framework can be split into two parts. The first part of the propagation model aggregates information from the neighbors of each node. The second part of the model then updates the state of each node with a GRU-like module, which takes the previous state of the node and the aggregation message obtained in the first part as the input. The detailed description of the model processes is provided in the following text.

In this study, we selected seven variables to model the different relationships between companies and constructed a comprehensive and complex graph representation for the downstream task. The seven selected variables were the "location of a company", "the industry attribution", "similarity of products", "scale of the company", "assets", "EPS", and "dividend". These variables were also called as data fusion, and all the edges represent pairwise comparisons between two nodes. We choose these variables because it can help users achieve their goals of risk management and maximize the profit. Considering that these variables need to be compared between the two, we chose the relation network architecture. The aforementioned variables are described in the following text:

- Location of a company's headquarters: A firm's location can have crucial implications for its operations. For example, a firm's location can affect its acquisition decisions, financial structure, returns, wage expenses, and access to essential business services [42–44]. Moreover, the clustering effect may lead to a strong industry chain. Consequently, the location of a company's headquarters is selected as one of the variables in the model.
- The industry attribution: Attribution analysis is an evaluation tool used to analyze and explain a portfolio's performance. The adjacency matrix of this variable is a binary assumption. If the companies in each pair belong to the same industry, the value of this variable is 1; otherwise, the value of this variable is 0.
- Similarity of products: Many companies belonging to the Dow Jones Industrial Average (DJIA) 30 use a strategy of diversification in their business, that is, conglomeration. Therefore, we used an edge type to represent the similarity between the products serviced by these companies.
- Scale of the company: The scale of a company is one of the factors in the Fama–French three-factor model, which aims to describe stock returns [45]. Therefore, we select this factor to model the relationships between companies and the constructed graph representation. The number of employees is used to form a matrix. The higher the number of employees hired by a company, the higher is the cost incurred by it. Differences in scale considerably affect the speed with which information, such as commands, decisions, or strategies, are propagated. The higher the value, the closer the scale of two companies.
- Assets: In the Fama–French five-factor model [46], assets are a crucial variable for explaining stock prices. Therefore, we select assets as one of our model variables. The assets represent the minimum guaranteed value of an enterprise. In our graph, a matrix is constructed using the value computed from the balance sheet of a company.
- EPS: EPS refers to earnings per share, which is defined as the monetary value of earnings per outstanding share of common stock for a company. For listed companies, the EPS and the company's stock price have a certain relation; EPS not only helps measure a company's current financial standing but also helps track its historical performance. Thus, the EPS is a key factor for the company's existing shareholders and potential investors to measure the company's profit.
- Dividend: A firm's dividend policy has a crucial effect on the current price of its shares [47]. Most companies listed on the US stock market issue cash dividends to investors. Investors often believe that the issuance of dividends implicitly indicates that the development of a company is mutual and steady and that the company earns substantial profits.

Figs. 3 and 4 display the graph obtained in the experiment. The nodes denote companies, and the edges denote the relations between two companies. Fig. 3 shows the stacked graph of all the edge types. To simplify the setting of graph input for our improved GGSNN, the adjacency matrix of the EPS and dividend is a weighted average result over the quarter to which the training months belong. The seven independent adjacency matrices are concatenated together, and seven learnable weight matrices are used to conduct feature mapping separately for the

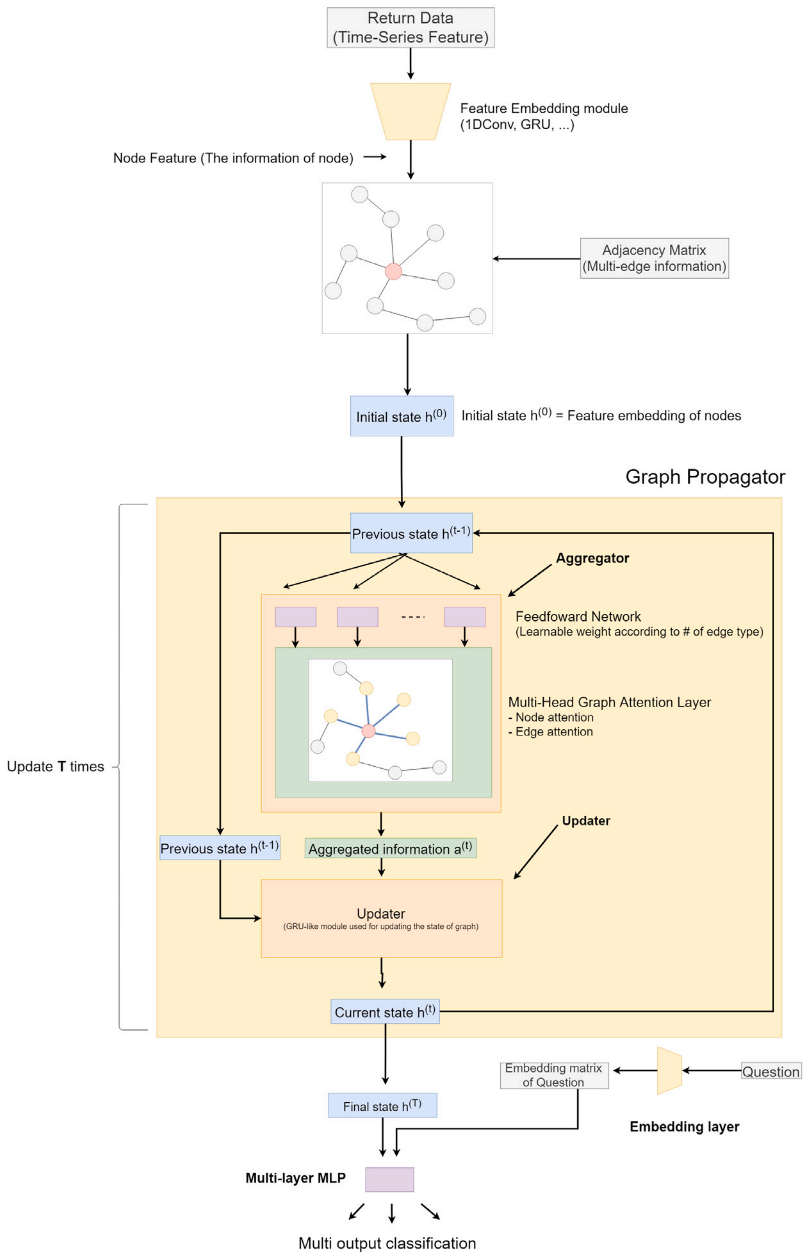

Return Data (Time-Series Feature)
Feature Embedding module (1DConv, GRU, ...)
Node Feature (The information of node)
Adjacency Matrix (Multi-edge information)
Initial state h(0)
Initial state h(0) = Feature embedding of nodes
Graph Propagator
Previous state h(t-1)
Aggregator
Feedfoward Network (Learnable weight according to # of edge type)
Multi-Head Graph Attention Layer
- Node attention
- Edge attention
Update T times
Previous state h(t-1)
Aggregated information a(t)
Updater
Updater (GRU-like module used for updating the state of graph)
Current state h(t)
Embedding matrix of Question
Question
Embedding layer
Final state h(T)
Multi-layer MLP
Multi output classification

Fig. 2. Architecture of our proposed model: Attention-based GGSNN.

graph representation. The graph representation is obtained through an independent matrix multiplication or concatenation of each edge type and input feature of each adjacency matrix.

It can be observed that model fusion enhances the learning ability of related models; while individual models may not be able to learn different market perspectives, fusion techniques can be used to accomplish this task.

Additional company factors that do not change frequently can be added to the graph representation. The more precise the indicator or factor added, the closer is the graph to the market condition. The factors that change frequently are suggested to be set as features rather than edge types in the graph.

3.4. Aggregation function and attention mechanism

In this study, we split the propagator model into two modules: the aggregator and updater. The aggregator is used to collect the information of neighboring nodes, and the updater is used to update the node state. In previous studies on GNNs with a gate mechanism, improvement measures focused on the gate module used in the updater or the application of different types of graphs.

In 2017, the attention mechanism [33] was used to address the drawback of the model based on graph convolution. On the basis of the aforementioned study, we believe that using the attention mechanism in the aggregation stage to determine the weighted sum of the information obtained from neighbors is crucial. Because the application domain of this study is financial time series, the importance of each node or edge type differs in different periods; therefore, the proportion of information from each neighbor to be propagated to the target node should be different. Eq. (1) presents the aggregate function of the GGSNN. The message a is simply a matrix multiplication of the hidden state of all nodes and the adjacency matrix A. Thus, the neighbor nodes have equal importance for the target node.

$$\begin{aligned} h_v^{(0)} &= [x_v,\ Zero - Padding] \\ a_v^{(t)} &= A[h_1^{(t-1)}, \dots, h_N^{(t-1)}]^T + b \end{aligned} \tag{1}$$

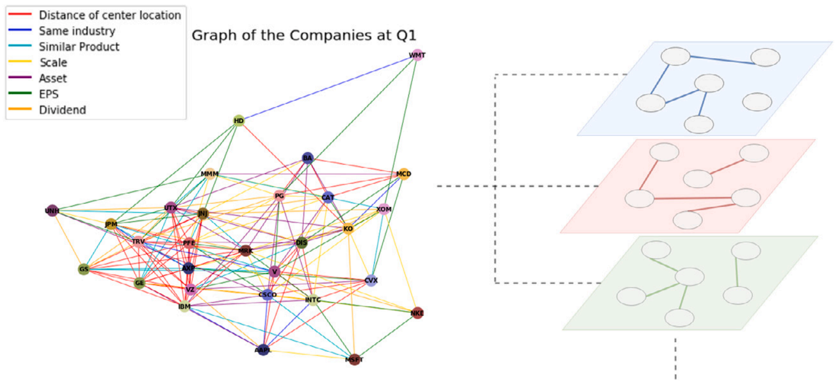
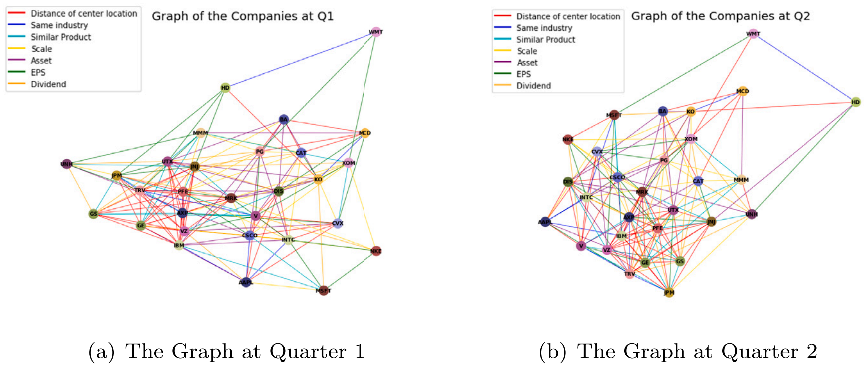
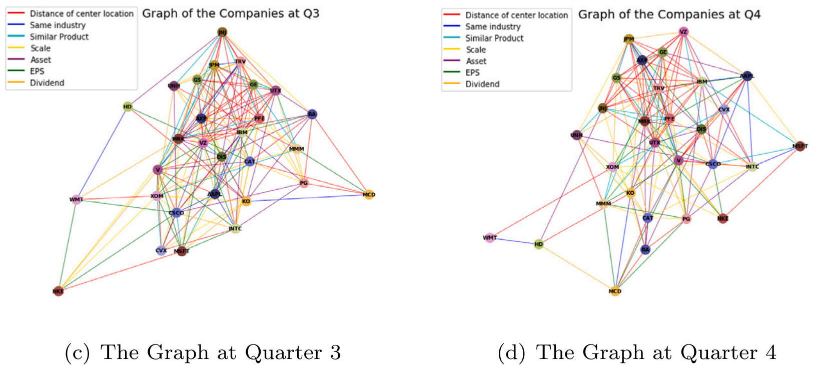


Fig. 3. The stacked multi edge-type graph of our work.



(a) The Graph at Quarter 1 (b) The Graph at Quarter 2



(c) The Graph at Quarter 3 (d) The Graph at Quarter 4

Fig. 4. An illustration of our multi edge-type graph.

The attention mechanism is used to pay attention to different regions of an image or correlate words in a sentence. And so does the region information of nodes in a graph. To implement the node attention mechanism, a learnable alignment model should be developed to calculate the importance score s and attention weight α for each node. The method used to attend the neighbor nodes to target nodes is illustrated in Fig. 5.

In this study, we used the graph to attend itself by computing the importance score for each pair of nodes related by the adjacency matrix A. We also calculated the importance scores for each pair of target node and its neighbor node (the score of the target node was also included). Thus, the computation of α only focused on the local information of each target node. The importance score can be expressed as follows:

$$s_{ij} = a\left([Q\vec{h}_i, Q\vec{h}_j]\right) \tag{2}$$

It is used to represent the importance of features from nodej to nodei. In Eq. (2), $h = h_1, h_2, \dots, h_N$ and N is the number of nodes. The computation of $scores_{ij}$ only focuses on the neighbors of each targeted $node_i$. The parameter Q is a learnable weight matrix used for the linear

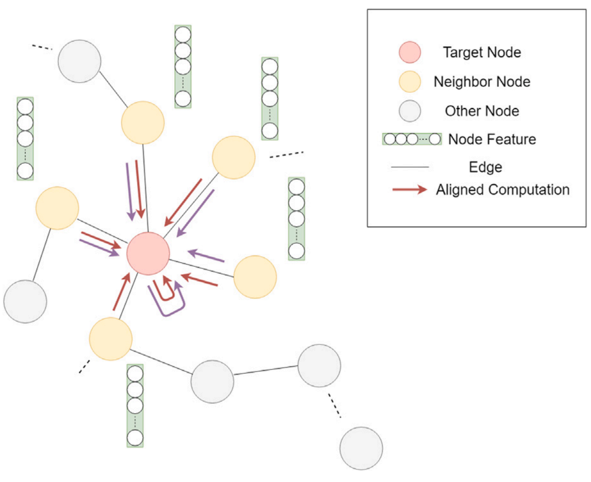


Fig. 5. An illustration of node attention.

transformation of node features, and a is a shared mapping function of the attention mechanism. Both Q and a are implemented using a feed-forward neural network. To calculate the attention matrix α, the aligned score must be normalized across all choices of $node_j$ with the softmax function. The attention matrix α is calculated as follows:

$$\alpha_{ij} = \text{Softmax}(s_{i,j}) = \frac{\exp\left(\tanh\left(a^T\left[W\vec{h}_i, W\vec{h}_j\right]\right)\right)}{\sum_{k\in N(i)\cup\{i\}} \exp\left(\tanh\left(a^T\left[W\vec{h}_i, W\vec{h}_k\right]\right)\right)} \quad (3)$$

The attention matrix α can be used to weigh node features and implement edge attention. When attention-based information is used, the formula of the aggregator can be expressed as follows:

$$a_v^t = \left(\sum_{u\in N(v)\cup\{v\}} \alpha_{v,u} W h_u^{t-1}\right) \quad (4)$$

Attention can also be used to assign importance to each type of neighbor of targeted nodes in the case of multi-type edge information. The importance of an edge may vary, especially in the finance domain. In the developed model, different types of edges are constructed according to the status or characteristics of companies, and each edge (relation) type has different degrees of influence on the market. By adopting edge attention, the weighted information across different relations can be combined into an aggregated message and used to update the state of the graph. The score is expressed as follows:

$$s_e = a_{edge}^t \quad (5)$$

This simplifies the alignment to only depend on the aggregation information of the target edge type after the node attention mechanism is applied. The attention matrix β is computed as follows:

$$\beta_{edge} = \text{Softmax}\left(\tanh\left(W_E \cdot s_{edge}\right)\right) \quad (6)$$

The attention matrix β can be used to weigh the aggregated information of an edge. In this case, the aggregator formula can be expressed as follows:

$$a_v^t = \left(\sum_{edge\in E} \sum_{u\in N(v)\cup\{v\}} \beta_{edge}\alpha_{v,u} W_{edge}^t h_u^{t-1}\right) \quad (7)$$

Finally, to stabilize the learning process of the attention mechanism, we employed multi-head attention. We constructed M independent attention mechanisms to execute the transformation presented in Eq. (7) and then conducted an averaging operation on the attention results. The aggregator formula can be rewritten as follows:

$$a_v^t = \left(\frac{1}{M}\sum_{m=1}^{M} \sum_{edge\in E} \sum_{u\in N(v)\cup\{v\}} \beta_{edge}\alpha_{v,u} W_{edge}^{t(m)} h_u^{t-1}\right) \quad (8)$$

After mapping the attention-based aggregator, one can obtain an aggregated message a_v, which comprises the information collected from the neighboring nodes of the target nodes. This information is a comprehensive representation of features across different edge types from multiple independent aggregators. The updater of our attention-based model, which is used to update the state of the graph, can be expressed as follows:

$$\begin{aligned}
h^{(0)}{}_v &= [x_v,\ Zero - Padding] \\
a^{(t)}{}_v &= \left(\frac{1}{M}\sum_{m=1}^{M} \sum_{edge\in E} \sum_{u\in N(v)\cup\{v\}} \beta_{edge}\alpha_{v,u} W_{edge}^{t(m)} h_u^{t-1}\right) \\
z_v{}^{(t)} &= \sigma(W^z a_v{}^{(t)} + U^z h_v{}^{(t-1)}) \\
r_v{}^{(t)} &= \sigma(W^r a_v{}^{(t)} + U^r h_v{}^{(t-1)}) \\
\widetilde{h_v}^{(t)} &= tanh(W a_v{}^{(t)} + U(r_v{}^{(t)} \odot h_v{}^{(t-1)})) \\
h_v{}^{(t)} &= (1 - z_v^{(t)}) \odot h_v^{(t-1)} + z_v^{(t)} \odot \widetilde{h_v}^{(t)}
\end{aligned} \quad (9)$$

The overall propagation function is expressed as follows:

$$h_v^t = \text{GRU}\left(h_v^{t-1}, \sum_{edge\in E} \sum_{u\in N(v)\cup\{v\}} \beta_{edge}\alpha_{v,u} W_{edge}^t h_u^{t-1}\right) \quad (10)$$

To construct a question answering system, first, we used an embedding layer for each company and obtained the embedding of target company pairs. Second, we concatenated the distributed representation of our question with the final state of the graph. Third, the concatenated result was fed into a multi-layer feed-forward neural network.

$$o = O_\theta(h^T, Q) \quad (11)$$

Finally, the prediction of different output layers was obtained. In Eq. (11), C denotes the number of classes and Q denotes the number of question types.

$$loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{q=1}^{Q}\sum_{c=1}^{C} \mathbf{1}_{y_i\in C_c} \log p_{\text{model}}\left[y_i \in C_c\right] \quad (12)$$

The learning framework of our model is presented in Algorithm 1. The following section introduces a variant of the proposed model.

Algorithm 1 Framework of Improved Gated Graph Sequence Neural Network for Financial Time-Series

```
Require:
    $I_{C\times T}$ : The input matrix, which is consist of the return data of all C companies during a period of time T
    $Q$ : The question of two targeted companies to be answer
    $A_E$ : The adjacency matrix of all E edge types, the concatenation of adjacency matrix for each edge type
Ensure:
    $A$ : The answer of the question we ask
 1: Feeding the input matrix $I_{C\times T}$ into feature embedding module and get high-level representation of feature
 2: Build the graph
    Take the embedding of feature of each company as node features $n_F$.
    Prepare the adjacency matrix $A_E$.
 3: Initialize the hidden state $h_0$ of graph neural network with the node features $n_F$.
 4: Go into the Graph Propagator:
 5: for each $t \in [1, T]$ do
 6:     Go into the Aggregator in Propagator:
 7:     Do linear transformation with feed-forward neural network for each edge type.
 8:     Node attention for all target nodes and their neighbor nodes.
 9:     Edge attention for node attention result of all edge types.
10:     Go into the Updater in Propagator:
11:     Use the aggregated information from the Aggregator to update the hidden state $h_{t-1}$ of graph
12:     Get the new state of graph $h_t$
13: end for
14: Output final state of graph $h_T$
15: if The distributed representation of $Q$ is needed then
16:     Get the embedding of two companies $E_{c1}$, $E_{c2}$ and the type of question $E_q$ by feeding into two different Embedding layer separately
17:     $Q$ = Concatenate($E_{c1}$, $E_{c2}$, $E_q$)
18: end if
19: Concatenate the final state of graph $h_T$ with $Q$ and feed it into a Multi-layer feed-forward neural network
20: Map the final embedding result to predict answer $A$
21: return $A$
```

3.5. Variant of the proposed attention-based model

An RNN module is appropriate for time-series forecasting. Moreover, modern RNN modules, such as LSTM and GRUs, avoid the gradient vanishing or exploding problem due to the memory cell; thus, these modules can preserve long-term information. However, RNN modules are difficult to implement in parallel and lead to an increase in the computational complexity. Therefore, in this study, we used a GRU cell instead of a GRU to learn the embedding of features. Furthermore, we used two approaches to update the state of the graph.

Features are randomly selected from N time steps for all C companies and sorted by time. These features are then used as the input of GRUCell Version 1. Subsequently, all the feature vectors of each company in each time step are tagged with their corresponding company and time annotations.

The aforementioned operation is conducted to train a robust GRU module that can represent a continuous change of features through time. Then, the concatenation of random and sorted features as well as the annotations of company and time are used as the input of GRUCell Version 1 to obtain the embedding of features. The hidden state of the graph propagator is subsequently initialized with the average embedding as follows:

$$h^{(0)}{}_v = \frac{1}{E}\sum_{edge}^{E} \text{GRUCell}_{\text{edge}}([x_v[RandomAndSortIdx], Anno_c, Anno_t]) \tag{13}$$

Next, the state of the graph is updated using the proposed gated graph model according to the initial state $h^{(0)}$. Moreover, to increase the stability of the updating of the hidden state in the graph and the convergence of the learning process, a skip connection is added in the output state (d denotes the skip time step of the graph updater).

$$h_v{}^{(t)} = h_v{}^{(t)} + h_v{}^{(t-d)} \tag{14}$$

GRUCell Version 2 is similar to GRUCell Version 1 in terms of the embedding of features; however, the updating method for the graph state differs between the two models. In GRUCell Version 2, first, features from N time steps are randomly selected for all C companies and sorted by time. These features are then used as the input of the aforementioned model. Second, all the feature vectors of each company in each time step are tagged with their corresponding company and time annotations. Third, the node feature in each time step is subjected to linear transformation, and the hidden state of the GNN is initialized with the raw feature of the first time step x. Next, E independent GRUCell modules are used to obtain the next hidden state of GRUCell Version 2 ($h_{GRUCellt+1}$) for all GRUCell edges. The states of the nodes in the graph are then updated as h t + 1. Moreover, a skip connection is added in the output state [e.g., the connection presented in Eq. (14)]. Finally, the new state of the graph (h_{t+1}) is taken as the hidden state of GRUCell Version 2. This state is used to perform feature embedding and then obtain the next state of GRUCell Version 2 ($h_{GRUCellt+2}$) with the feature of the next time step x_1. Thus, the fixed number of time steps (T) used to call the graph updater is decided by the selected value of N.

A high-level representation of financial time-series data can be obtained using a 1D convolution layer. A shared parameter filter is passed through the entire time series and used to extract the short-term information and spatial location between each channel, which are independent factors. Moreover, the 1D convolution operation can be performed in parallel and is faster than the RNN module in obtaining the embedding of features. Some studies [48,49] have indicated that convolution-based models can outperform or match the performance of RNN models in various tasks. A high-level representation of features can be obtained using a multi-layer convolution network with a time-series input in which the embedding is taken as the initial hidden state and the node features of the graph are used to perform the follow-up operations. The later section will introduce our relation network architecture and then introduce how to apply financial data on 1D and 2D Convolution-based Relation Network respectively.

A temporal convolution block involves a combination of causal and dilated convolution. Causal convolution is used to ensure that the prediction of the previous time step is not based on future information because the output $\hat{y}$ of time step t is only derived from the convolution operation at time $t-1$ and the previous time steps. Moreover, dilated convolution is used to enlarge the receptive field by setting different dilation rates in hidden layers. The purpose of dilated convolution is to capture long-term information and avoid information loss due to the use of pooling. Therefore, in this study, we used a multi-layer dilated 1D convolution layer to obtain the embedding of our feature. Then, we used soft alignment to implement graph attention with the node features and follow-up layers.

3.6. Embedding financial time-series data by using a relation network

This section introduces the framework of the modified relation network, which is used to learn the relationship between different financial time-series signals through the deep learning approach. Portfolio managers traditionally use non-correlated stocks to lower risk, but we analyzed financial time-series data on volatility and other financial indicators for model prediction accuracy. Thus, we can invest in stocks with low volatility regardless of whether their correlations are high or low. The system overview is displayed in Fig. 6.

Inspired by the relation network proposed by DeepMind [35], the relationship between different objects can be learned using a relation network because this network solves problems on the basis of relational reasoning. For a relation network, we argue that when the task involves relational reasoning, obtaining a good result by training the model with pure CNN layers and a multilayer feed-forward neural network architecture is difficult. This is because these typical network architectures cannot execute relational modeling if the networks are simply stacked together. To address this problem, we can consider two aspects of learning approaches: modeling the network architecture by using a graph-based method or retaining the input as unstructured data and then designing a module that can learn the relationships among data.

In this study, we converted the collected financial time-series data in a similar manner to how an RGB image is converted into a matrix of pixel values. First, we created a time-series sequence comprising time-series signals of factors such as the price or return value of a company for a certain period. Multivariate time-series data are represented as multiple directed graphs, each of which contains T nodes. Given n exogenous series, the exogenous series at time t, namely $X_t = (X1_t, X2_t, \dots, Xn_t) \in R^n$, as the features of node t in the graph structure, $X = (X_t, X_t, \dots, X_{t-1})$ represents the neighborhood node features of node t. Then, we constructed a matrix containing several time-series sequences of companies for a certain period, which denoted the historical information of these companies. The aforementioned matrix has the appearance of image V in the VQA system. The price or return value of a company has the same relationship with a time-series sequence as a pixel value does with an image. When handling input data, we convert the closing price into the form of return data, which is a profit on an investment, for normalization. Return data are defined as follows:

$$r(t) = p_{t+1} - p_t \tag{15}$$

where $r(t)$ is the return data at time t, p is the closing price data. For an intensive study, the relation-focused model and question encoding state can be learned simultaneously by sequentially feeding the sentence into word embedding and LSTM.

The proposed model forecasts the future market condition A by learning historical information, such as the return data r, from the input matrix I used according to the corresponding question Q. First, the input matrix I_{C*T} is taken as the convolution layer's input. Second, each question is set as a multi-hot encoding vector or embedding vector according to whether a distributed representation of the question sentence is required. Third, a relation network module must be constructed. The input of this network is a set of objects expressed as $O = (o_1, o_2, \dots, o_n)$, and the number of objects isset in accordance with the sequence length of final CNN feature maps. If the number of final CNN feature maps is N and the length of each map is M, the

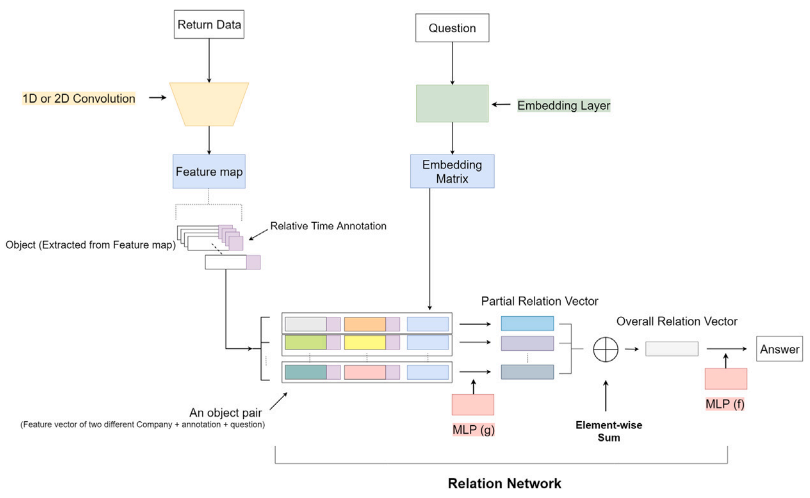


Fig. 6. The overview of relation network architecture in financial time-series data.

"object" is a vector with M objects that has a length of N, that is, each object represents a high-level feature at each relative time $t \in [0, M]$, after convolution. Once the object set has been obtained, a module is constructed to determine the relationships between all objects. From the viewpoint of graph theory, the module input can be represented using a complete and undirected graph whose nodes are objects and whose edges represent the relations between each object pair. In implementation, each object is tagged with coordinates to denote its time information, and two MLPs are then used to infer object relations. First, an MLP is used to compute each object–object pair relation, which is a partial relation. Second, the element-wise summation results of the output of the first MLP are fed to the remaining MLPs to compute the overall relation. Finally, the relation network module learns the relations between each object pair, conducts question embedding (seen as a condition), and integrates these components to answer a question.

According to [35], the simplest function of the relation network module is that presented in, where o_i, o_j means each object–object pair, each object o is the vector mentioned above. g_θ means the first MLP which uses the same parameter θ and it will be used to compute partial relations. f_ϕ means the second MLP which uses the same parameter ϕ and it will be used to compute overall relation.

$$RN(O) = f_\phi\left(\sum_{i,j} g_\theta(o_i, o_j)\right) \tag{16}$$

When inferring object relations by this module, each object o will be tagged with time coordinate c and each object–object pair will concatenate the question embedding vector. So we can rewrite the formula below.

$$RN(O) = f_\phi\left(\sum_{i,j} g_\theta((o_i, c_i, o_j, c_j), q)\right) \tag{17}$$

Although 2D convolution is not possible due to the characteristic of the input data that the company order is irrelevant, we modified the training strategy of our network so that it could retain the structure and information of each sequence and perform the 2D convolution operation on our time-series data. The following text introduces the architecture of our relation network and indicates how to perform 1D and 2D convolution on financial data.

In our experiment, we use a 1D-convolution-based relation network and 2D-convolution-based relation network as our benchmarks for comparison with the graph neural network model proposed in this section. Therefore, this section describes how the developed 1D convolution-based model is applied to financial time-series data. First, the data of all companies in the training dataset for a short period of time T are taken as the input I_{C*T} of multistacked 1D convolution layers, and a feature map F is obtained after convolution. Second, the question Q of two target companies is converted into an embedding encoding vector or a one-hot encoding vector. Third, the object pairs extracted from the final CNN feature maps are concatenated with the corresponding time coordinate and question embedding vector. Fourth, the concatenated vector is fed into the relation network module for computing relations. Finally, the output corresponding answer A is obtained. The relation network module considers relations across all pairs of objects that are conditioned on the question embedding and tagged with the time and company coordinates. It integrates all these components to answer the question.

Then we are going to describe how our network is modified to contain 2D convolution layers. The keywords in the questions are processed with an embedding layer to produce a question embedding vector or simply converted to multihot vectors. The input matrix is processed through multilayer 2D convolution by using filters on the time axis of the input data to produce a set of objects for the relation network module; thus, the convolution result of each company is preserved in the input matrix. Objects are constructed using the feature map after conducting the convolution operation on the return data. Then, the relation network module considers relations across all pairs of objects conditioned on the question embedding and tagged with time and company coordinates. It integrates all these components to answer the question. This operation is beneficial because it enables company information to be retained due to the fact that convolution is conducted separately for each company.

4. Experiments

The proposed model was evaluated by conducting three experiments. This section describes the dataset used in the experiments and defines the evaluation metrics used to compare the proposed model with other approaches. The experimental settings are then detailed. Subsequently, the experimental results are presented and discussed.

4.1. Dataset and experimental setting

The training data used for modeling were obtained by preprocessing the raw trade data stored in the WRDS New York Stock Exchange (NYSE) Trade and Quote database. The trade data contained all the trading information for 2017 of all the companies listed on the NYSE. We unified the trading data to be second data. The data obtained after preprocessing contained information on the opening, highest, lowest, and closing prices as well as the trading amount per second for each company. We selected the closing price data of the companies belonging to the DJIA 30 for modeling all the proposed networks. The

Table 1
The HyperParameters of our work.

Experimental parameters in our work	
Time length of each image(frame)	300 s
Shift length for each image	150 s
Predict time length of future	150 s
# of head used in graph attention	2
# of time T used for updating graph	4
Threshold δ of attention mask	0.0
Optimizer with its initial learning rate	Adams(3e−5)

DJIA is one of the oldest and most credible market indices in the United States. The members of the DJIA 30 are the largest and most well-known 30 listed companies in the United States. For normalization, we converted the closing price into return data and used these data as our input data. The return data are defined as follows:

$$r(t) = p_t - p_{t-1} \tag{18}$$

where p_t denotes the closing price at time t. We discarded the price data of two companies, namely Dow Chemical and Du Pont, because these companies merged on August 31, 2017, which led to different principles of comparison for the same company. The merged company had different price ranges in different periods of 2017. Therefore, the number of companies in our training dataset was 29.

Rolling analysis is an approach in which time-series data are used to assess a model's stability over time. The rolling window of our experiment was defined as 6 months of data for training and 1 month of data for testing. Then, shift 1 month to get next 6 months and one month for training and testing until the end of the time. After rolling the entire dataset, six results were obtained for each model in our experiments.

According to the timing of the stock market in the United States (from 9:30 a.m. to 4:00 p.m.), closing price data for 23 400 s on each trading day were considered. To construct a VQA-like system, one requires to create a training image, a frame containing the return data of all companies for a certain period T, information on the future condition of the training image, and a frame containing the return data of all companies for a certain period T_f. In our experiment, T was 300 and T_f was 150. Thus, we used the first 300-s data to predict the following 150-s data. We shifted the frame for each $T_s = 150$; thus, the time overlap for each continuously generated frame was Ts. The experimental process is described in the following text.

To simplify the complexity of our VQA-like system, the question type was predefined. Three types of questions were learned with our model: questions on "pairwise price change", "strength", and "volatility". The "pairwise price change" indicates the difference in the price trend between two companies, and the answer set for this parameter is ["NoChange", "BothRise", "BothFall", "Reverse"]. "Strength" indicates the difference in the strength as well as the rate of uptrend or downtrend between two companies. The answer set for this parameter is ["Same", "Strong", "Weak"]. "Volatility" denotes the difference in the fluctuation between two companies, and the answer set for this parameter is ["Same", "Big", "Small"]. These question set and answer set were also be seen as one of data fusion. To train the proposed model, some parameters must be known. The parameter setting for the proposed model is presented in Table 1.

4.2. Evaluation metrics

In this study, we predicted the future market condition by using a deep learning model with multiple outputs as well as multiclass classification. To evaluate whether the proposed model outperforms a benchmark model, we determined the precision, recall, and F1 score of the models in the inference phase.

The maximum drawdown (MDD) is used to evaluate the return stability of an investment over a specified period. It indicates the maximum loss from a peak to a trough of accumulated return, which depends on the largest price drop. The MDD is expressed as follows:

$$MDD = \frac{V_{peak} - V_{trough}}{V_{peak}} \tag{19}$$

where V_peak is the peak value before the largest drop and V_trough is the trough value before a new peak is achieved. The lower the MDD, the lower the risk.

In the experiment described in the following text, we evaluated the overall accuracy, F1 score, and MDD for each model.

4.3. Benchmark model for financial time series: DRN

In one of our experiments, we constructed a model for benchmarking the relational learning model of financial time series. We used this relational model for learning in a reason task in the finance domain by adding a temporal annotation to each object. Moreover, we performed a 2D convolution operation with a special filter size on each company in the input data to preserve the information of each company.

The rolling test results obtained in the experiments are presented in Tables 2 and 3. Table 2 presents a comparison of the accuracy of the two proposed models (i.e., the 1D-convolution-based and 2D-convolution-based relation networks), baseline model (i.e., the MLP model), and random choice model for the three question types.

The random choice model had lower accuracy for question types 1 and 2 than did the other four models. The reason for the inaccuracy of the random choice model for question type 3 is that our problem setting involved a pairwise case. A between-target volatility of 0 is rare—sometimes present in less than 10% of cases. If no case has constant volatility between the targets, the random choice is 0.5. Otherwise, it approximates but is less than 0.5. Otherwise, it will be close to 0.5 but less than 0.5. For deep learning, at the baseline, the performance of LSTM was much better than that of DNN, indicating that LSTM is more suitable for solving the problems of stock price prediction and portfolio construction. Among the five models compared, the 2D-convolution-based relation network had the highest accuracy for question type 1. Thus, the 2D-convolution-based relation network, which was expected to preserve the information of each company, achieved the best result when examining the relative trends of pairwise companies. The aforementioned model may be applicable to research on relative correlation. The proposed models, namely the MLP and LSTM models, had 2%–3% higher accuracy than the baseline model. Moreover, the overall accuracy of the 1D-convolution-based relation network was marginally higher than that of the 2D-convolution-based relation network for most months in the rolling test. The results in Table 3 indicate that the 2D-convolution-based relation network outperformed the 1D-convolution-based relation network as well as the LSTM and MLP models. The F1 score of the 2D-convolution-based relation network was approximately 2/3 times higher than that of the 1D-convolution-based relation network and the LSTM and MLP models for each question type in the rolling test. The two variants of the improved relation network model were the benchmark models for the developed attention-based GGSNN. The following section describes the experiment conducted on the developed GGSNN model.

This section describes the setting of the experiment conducted on the variants of the GGSNN.

4.4. Comparison of the original and attention-based GGSNN models

To build a graph with multi-edge information, we need to select some useful status or To evaluate the performance of the proposed model, we conducted an experiment on the original and attention-based GGSNN models. We trained four variants of the GGSNN model, namely the original GGSNN with a complete graph, the original GGSNN with our multiedge graph, the GRUCell version of the attention-based

Table 2
The overall accuracy of our benchmark model — Relation network in 2017.

Overall test accuracy for relation networks in 2017						
		Random choice	Multi-layer DNN	LSTM	1D-Conv. based RN	2D-Conv. based RN
July	Overall accuracy	–	0.3613	0.3692	0.3817	**0.4085**
	Q1 accuracy	0.2500	0.2538	0.2468	0.2609	0.3320
	Q2 accuracy	0.3333	0.3491	0.3700	0.3521	0.3770
	Q3 accuracy	0.4∼0.5	0.4809	0.5741	0.5321	0.5164
August	Overall accuracy	–	0.3770	0.3877	**0.3993**	0.3895
	Q1 accuracy	0.2500	0.2657	0.2562	0.2945	0.3163
	Q2 accuracy	0.3333	0.3458	0.3533	0.3706	0.3696
	Q3 accuracy	0.4∼0.5	0.5195	0.5537	0.5329	0.4825
September	Overall accuracy	–	0.3667	0.3755	0.3821	**0.3871**
	Q1 accuracy	0.2500	0.2563	0.2538	0.2549	0.3019
	Q2 accuracy	0.3333	0.3593	0.3762	0.3567	0.3575
	Q3 accuracy	0.4∼0.5	0.4845	0.4967	0.5346	0.5020
October	Overall accuracy	–	0.3537	0.3763	**0.3816**	0.3713
	Q1 accuracy	0.2500	0.2466	0.2343	0.2633	0.2780
	Q2 accuracy	0.3333	0.3399	0.3564	0.3614	0.3546
	Q3 accuracy	0.4∼0.5	0.4745	0.5381	0.5201	0.4812
November	Overall accuracy	–	0.3681	0.3800	**0.3838**	0.3719
	Q1 accuracy	0.2500	0.2658	0.2424	0.2928	0.2771
	Q2 accuracy	0.3333	0.3673	0.3664	0.3696	0.3543
	Q3 accuracy	0.4∼0.5	0.4713	0.5312	0.4891	0.4844
December	Overall accuracy	–	0.3883	0.3762	**0.3930**	0.3875
	Q1 accuracy	0.2500	0.2780	0.2495	0.3100	0.3009
	Q2 accuracy	0.3333	0.3806	0.3772	0.3706	0.3762
	Q3 accuracy	0.4∼0.5	0.5063	0.5019	0.4985	0.4855
Average	Overall accuracy	–	0.3692	0.3774	**0.3869**	0.3860

Table 3
The classification report of relation network models during inference phase.

Classification report of relation networks for rolling test in 2017																		
	July									August								
	Q1			Q2			Q3			Q1			Q2			Q3		
	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
Multi-layer DNN	0.15	0.23	0.26	0.16	0.11	0.33	0.06	0.53	0.50	0.06	0.18	0.25	0.24	0.24	0.34	0.03	0.50	0.50
LSTM	0.19	0.25	0.22	0.11	0.33	0.16	0.03	0.50	0.05	0.01	0.11	0.02	0.12	0.35	0.18	0.00	0.02	0.00
1D-Conv. based RN	**0.22**	0.23	0.23	0.29	0.32	0.33	0.36	0.50	0.50	0.20	0.27	0.25	**0.27**	0.25	0.34	**0.49**	0.50	0.49
2D-Conv. based RN	**0.22**	0.26	0.30	**0.31**	0.26	0.38	**0.45**	0.49	0.48	**0.28**	0.29	0.32	**0.27**	0.24	0.33	0.48	0.50	0.48
	September									October								
	Q1			Q2			Q3			Q1			Q2			Q3		
	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
Multi-layer DNN	0.05	0.09	0.25	0.26	0.23	0.33	0.06	0.50	0.50	0.06	0.03	0.26	0.16	0.11	0.33	0.08	0.04	0.50
LSTM	0.01	0.09	0.02	0.12	0.35	0.18	0.47	0.50	0.49	0.01	0.08	0.01	0.11	0.33	0.16	0.01	0.08	0.01
1D-Conv. based RN	0.04	0.21	0.25	0.23	0.24	0.33	**0.51**	0.52	0.51	0.10	0.25	0.25	**0.24**	0.21	0.33	**0.50**	0.51	0.50
2D-Conv. based RN	**0.25**	0.26	0.31	**0.28**	0.32	0.33	0.24	0.50	0.50	**0.15**	0.27	0.28	0.21	0.38	0.33	0.12	0.51	0.52
	November									December								
	Q1			Q2			Q3			Q1			Q2			Q3		
	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
Multi-layer DNN	0.04	0.02	0.25	0.18	0.28	0.33	0.24	0.49	0.49	0.18	0.24	0.27	0.18	0.22	0.33	0.28	0.50	0.49
LSTM	0.01	0.07	0.01	0.13	0.35	0.19	0.00	0.06	0.01	0.00	0.07	0.01	0.13	0.36	0.19	0.46	0.50	0.48
1D-Conv. based RN	**0.19**	0.29	0.28	0.18	0.24	0.33	**0.50**	0.50	0.50	**0.25**	0.27	0.28	**0.28**	0.24	0.34	0.49	0.49	0.49
2D-Conv. based RN	0.17	0.27	0.27	**0.26**	0.24	0.33	0.41	0.50	0.50	0.10	0.07	0.27	**0.28**	0.24	0.34	**0.50**	0.52	0.51

GGSNN with our multiedge graph, and the temporal convolution version of the attention-based GGSNN with our multiedge graph. The number of hidden dimensions in the graph state was 64. The number of time steps (T) used in the updater of the graph was 4. The Adam optimizer was used, and its learning rate was $3 * 10^{-5}$. In the implementation of GGSNN models, superior results are obtained when the learning rate is small and weight initialization or batch normalization is conducted for each hidden layer.

The experiment conducted on the GGSNN models comprised three parts. In the first part, we compared the two variants of the original GGSNN with the frames of features generated from a VQA pair. We then input a complete graph, in which the nodes were connected by edges in the original GGSNN, or multiedge graph. This part of the experiment ensured the addition of edge information formed by different edge types to the GGSNNs, which improved their performance. In the second part of the experiment, we compared the original and attention-based GGSNN models. This part of the experiment ensured the addition of our overall graph attention mechanism to the GGSNNs, which improved their performance. In the third part of the experiment, we compared two variants of the attention-based GGSNN, namely GRUCell and the 1D convolution version. This part of the experiment was conducted to compare the performance of the aforementioned models when using different feature extraction modules to embed the raw feature. The updating mechanism of GRUCell is described.

Table 4 presents a comparison of the overall accuracy of the proposed models. Moreover, Table 5 presents the classification results of the aforementioned models. As presented in Table 4, the prediction results of all the proposed models are superior to those of the random choice model. Moreover, the proposed models provided excellent results for question type 3. In most cases, the testing accuracy of the attention-based GGSNN with a TCN block was higher than that of

Table 4
The overall accuracy of attention-based GGSNN in 2017.

Overall test accuracy of GGSNN for rolling test in 2017						
		Random choice	GGSNN_CompleteGraph	GGSNN	AttGGSNN-GRUCell	AttGGSNN-TCN
July	Overall accuracy	–	0.4222	0.4249	0.4144	**0.4267**
	Q1 accuracy	0.2500	0.2503	0.2549	**0.2570**	0.2521
	Q2 accuracy	0.3333	0.3621	0.3627	0.3594	**0.3629**
	Q3 accuracy	0.4~0.5	0.6529	0.6570	0.5491	**0.6650**
August	Overall accuracy	–	0.4372	0.4457	0.4122	**0.4485**
	Q1 accuracy	0.2500	0.2802	0.2723	0.2493	**0.2798**
	Q2 accuracy	0.3333	0.3607	0.3623	0.3466	**0.3635**
	Q3 accuracy	0.4~0.5	0.6646	**0.7023**	0.6408	0.7022
September	Overall accuracy	–	0.4367	**0.4497**	0.3963	0.4477
	Q1 accuracy	0.2500	0.2725	0.2818	0.2539	**0.2821**
	Q2 accuracy	0.3333	0.3575	**0.3681**	0.3621	0.3665
	Q3 accuracy	0.4~0.5	0.6801	0.6993	0.5727	**0.6994**
October	Overall accuracy	–	0.4288	0.4458	0.3896	**0.4478**
	Q1 accuracy	0.2500	0.2645	0.2848	0.2534	**0.2878**
	Q2 accuracy	0.3333	0.3602	0.3630	0.3583	**0.3631**
	Q3 accuracy	0.4~0.5	0.6615	0.6896	0.5573	**0.6924**
November	Overall accuracy	–	0.4356	**0.4427**	0.3902	0.4303
	Q1 accuracy	0.2500	0.2689	0.2757	0.2425	**0.2780**
	Q2 accuracy	0.3333	0.3638	**0.3699**	0.3600	**0.3699**
	Q3 accuracy	0.4~0.5	0.6742	**0.6825**	0.5563	0.6431
December	Overall accuracy	–	0.4454	0.4478	0.4121	**0.4526**
	Q1 accuracy	0.2500	0.2749	**0.2862**	0.2861	0.2836
	Q2 accuracy	0.3333	0.3661	0.3753	**0.4106**	0.3774
	Q3 accuracy	0.4~0.5	0.6951	0.6819	0.6019	**0.6968**
Average	Overall accuracy	–	0.4343	0.4415	0.4025	**0.4423**

Table 5
The classification report of graph-based models during inference phase.

Classification report of graph-based model for rolling test in 2017																		
	July									August								
	Q1			Q2			Q3			Q1			Q2			Q3		
	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
GGSNN_CompleteGraph	0.04	0.02	0.25	0.17	0.11	0.33	0.49	0.47	0.50	0.23	0.25	0.26	0.20	0.33	0.34	**0.50**	0.49	0.50
GGSNN	0.02	0.01	0.25	0.17	0.11	0.33	0.49	0.47	0.50	**0.26**	0.33	0.30	0.30	0.39	0.39	**0.50**	0.49	0.50
AttGGSNN-GRUCell	**0.22**	0.25	0.23	**0.26**	0.32	0.34	**0.50**	0.49	0.50	0.10	0.24	0.24	0.19	0.29	0.32	**0.50**	0.49	0.50
AttGGSNN-TCN	0.04	0.02	0.25	0.17	0.11	0.33	0.49	0.47	0.50	**0.26**	0.27	0.27	**0.31**	0.37	0.38	**0.50**	0.49	0.50
	September									October								
	Q1			Q2			Q3			Q1			Q2			Q3		
	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
GGSNN_CompleteGraph	0.05	0.14	0.25	0.17	0.11	0.33	**0.49**	0.47	0.50	0.06	0.03	**0.24**	0.17	0.27	0.34	**0.48**	0.46	0.50
GGSNN	**0.26**	0.29	0.31	**0.34**	0.38	0.39	**0.49**	0.47	0.50	0.24	0.26	0.28	0.25	0.33	0.34	**0.48**	0.46	0.50
AttGGSNN-GRUCell	0.25	0.26	0.27	0.27	0.24	0.34	**0.49**	0.49	0.49	**0.25**	0.26	0.33	0.25	0.27	0.34	0.10	0.53	0.51
AttGGSNN-TCN	**0.26**	0.29	0.31	0.31	0.37	0.39	**0.49**	0.47	0.50	**0.25**	0.29	0.29	**0.37**	0.37	0.38	**0.48**	0.46	0.50
	November									December								
	Q1			Q2			Q3			Q1			Q2			Q3		
	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
GGSNN_CompleteGraph	0.18	0.28	0.27	0.21	0.38	0.34	**0.48**	0.47	0.50	**0.21**	0.29	0.28	0.20	0.30	0.33	**0.48**	0.46	0.50
GGSNN	0.22	0.24	0.25	0.18	0.45	0.33	**0.48**	0.47	0.50	0.18	0.26	0.28	0.18	0.19	0.33	**0.48**	0.46	0.50
AttGGSNN-GRUCell	**0.26**	0.27	0.28	0.25	0.30	0.34	**0.48**	0.48	0.48	0.07	0.22	0.24	**0.26**	0.42	0.34	**0.48**	0.47	0.49
AttGGSNN-TCN	0.25	0.29	0.31	**0.28**	0.39	0.36	**0.48**	0.47	0.50	**0.21**	0.30	0.31	0.22	0.37	0.34	**0.48**	0.46	0.50

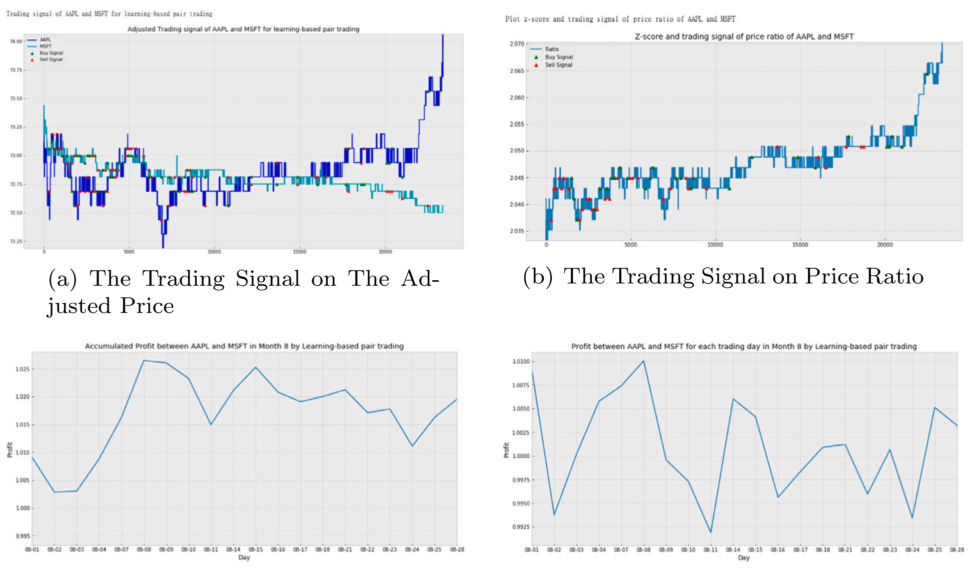
the original GGSNN during the training phase. Although the attention-based GGSNN with GRUCell was outperformed by the other models, it exhibited the capability of updating the node state in different time steps and the good representation of our proposed models while we only pick features of some time steps to learn the graph. As presented in Table 5, the attention-based GGSNN with a TCN block outperformed the other models. The testing accuracy results for each question type indicate the suitability of determining the overall relation by using graph models, which can uniformly learn all the question types in the training phase. The following section describes the application of the proposed models to pairs trading.

4.5. Financial application of our proposed models: pairs trading

We also used the proposed models to conduct pairs trading in a real market. Pairs trading is a market-neutral trading strategy that involves taking long and short positions for two stocks with a positive correlation. When the price difference (spread) between paired stocks deviated from the historical average, we simultaneously adopted a short position for the stock with a higher price and a long position for the stock with a lower price. We then waited for the stock pair to return to the long-term equilibrium relationship and thus earned profits from the price convergence of two stocks.

In the experiment, we executed pairs trading for each trading day by using intraday return data. We then calculated the accumulated return and MDD to examine the performance of the proposed models. We followed the following trading principles:

1. We adopted cointegration-based pairs trading as the rule-based strategy. Cointegration indicates whether the spread between two instruments is stationary, that is, whether the spread of



(a) The Trading Signal on The Adjusted Price

(b) The Trading Signal on Price Ratio

(c) The Accumulated Profit

(d) The Daily Profit

Fig. 7. A sample of overall neural network-based pairs trading.

two time series is around long-term average values. First, we conducted a hypothesis test on an entire day's price data for each stock pair constructed from the combination of the 29 companies in our data. Second, we selected the stock pair (A, B), whose p value in the cointegration test was smaller than 0.05. Third, we calculated the ratio for the price of the two companies constituting the stock pair. Fourth, we calculated the z score of the rolling result between the short- and long-term moving averages by using the following equation:

$$zscore = \frac{MA_{short} - MA_{long}}{standard_{long}} \tag{20}$$

Fifth, we implemented a long–short strategy according to the z score. A stock was bought (long 1 A and short r B) if the z score was smaller than -1, and a stock was sold (short 1 A and long r B) if the z score was larger than 1. Finally, we accumulated all the profits calculated for each frame.

2. Our NN-based strategy was based on a comprehensive prediction of the developed VQA-like system. The answers to all the three question types for each frame were considered when selecting an operation for different market situations. First, we created all the frames of a target trading day. Because a trading day comprised 23 400 s in this study, we could create 154 frame pairs of historical and future returns ($Concat([V_{[t-T:t]}, V_{[t:t+T_f]}])$, where t is some time around the opening time). Moreover, we only used all the frames of historical return ($\sum_{s=0}^{154} V_{t+s*T_s-T:t+(s+1)*T_s}$ where T_s is the shift time and $t = 300$, which indicates that trading begins in the 300th second after the opening of the market on each trading day). Second, we forecasted the future condition of all the question types. Finally, we accumulated all the calculated profits for each frame.

The following text describes our pairs trading experiment and reports the MDD of each model. To simplify the model comparison, we only selected two representative companies, namely Apple and Microsoft, to form a common standard for the evaluation. In future studies, we can automatically create a portfolio and recommend it to users.

Fig. 7 illustrates the neural-network-based pairs trading, which involves using the answer predicted by the model for each question type to determine whether to buy or sell a stock.

Finally, we examined the accumulated profits obtained in the rolling test for the last 6 months in 2017 when using the proposed models and baseline model. Fig. 8 presents the long-term profit in 2017. The results in this figure indicate that our attention-based GGSNN with a TCN block can be applied to finance markets. The good performance of our 1D-convolution-based and 2D-convolution-based relation networks may have been caused by our adopted strategy of determining the buy and sell signal mainly on the basis of the prediction for question type 1. The aforementioned models were mainly focused on learning the implicit pattern of question type 1 from the results in Table 2. However, our ultimate goal was to model the overall relationship, which provides a suitable and generalized representation of each type of relation setting for a question. Therefore, we had to examine the overall performance in terms of the comprehensive result for each evaluation metric.

4.6. Overall comparison of the relational learning models

This section presents an overall comparison of all the models used in our experiments. We examined the overall testing accuracy during the training phase by using a resampling approach. Then, we determined the MDD of all the models in the pairs trading experiment. As presented in Table 6, most of the graph-based models had a higher testing accuracy than did the random choice model. This is not available to reach in our proposed relational network used for benchmarking. The overall accuracy of the graph-based models was approximately 3%–6% higher than that of the variants of the relation network model. The 1D- and 2D-convolution-based RNs exhibited the best performance of all the baseline models, which means that they have some advantages over LSTM in analyzing stock price patterns. The proposed attention-based GGSNN with a TCN block had the highest F1 score and accuracy in most cases. According to Table 7 and all the figures of accumulated profit, this model had the second-lowest risk due to its small MDD; thus, the aforementioned model can earn profit. Although the GRUCell version of the attention-based GGSNN with new propagation did not outperform the attention-based GGSNN with a TCN block, the GRUCell version learned some patterns in the time series and outperformed the benchmark model. We believe that this model can exhibit superior performance if the information in each time step is increased. Additional input features can be added for the nodes to design other architectures based on the SOTA in the sequential learning domain.

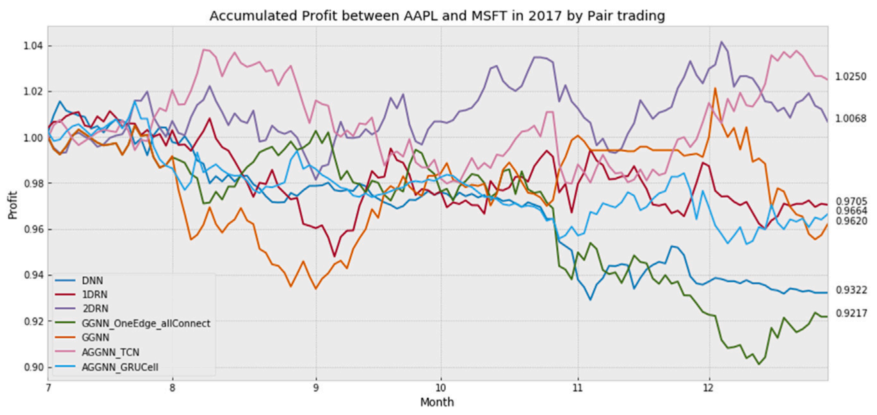


Fig. 8. The overall accumulated profit for all models.

Table 6
The accuracy and F1 score of all models for the rolling test in 2017.

Overall test accuracy and F1 score of all models for rolling test in 2017									
		Multi-layer DNN	LSTM	1D-Conv. based RN	2D-Conv. based RN	GGSNN_CompleteGraph	GGSNN	AttGGSNN-GRUCell	AttGGSNN-TCN
July	Overall accuracy	0.3613	0.3692	0.3817	0.4085	0.4222	0.4249	0.4144	**0.4267**
	F1 score	0.13	0.11	0.29	0.32	0.23	0.22	**0.33**	0.24
August	Overall accuracy	0.3770	0.3877	0.3993	0.3895	0.4372	0.4457	0.4122	**0.4485**
	F1 score	0.11	0.04	0.32	0.34	0.31	0.35	0.26	**0.37**
September	Overall accuracy	0.3667	0.3755	0.3821	0.3871	0.4367	**0.4497**	0.3963	0.4477
	F1 score	0.13	0.20	0.26	0.25	0.23	0.35	0.35	**0.36**
October	Overall accuracy	0.3537	0.3763	0.3816	0.3713	0.4288	0.4458	0.3896	**0.4478**
	F1 score	0.10	0.04	0.28	0.16	0.24	0.32	0.20	**0.37**
November	Overall accuracy	0.3681	0.3800	0.3838	0.3719	0.4356	**0.4427**	0.3902	0.4303
	F1 score	0.15	0.05	0.29	0.28	0.29	0.29	0.33	**0.34**
December	Overall accuracy	0.3883	0.3762	0.3930	0.3875	0.4454	0.4478	0.4121	**0.4526**
	F1 score	0.21	0.20	0.29	0.29	0.29	0.28	0.27	**0.30**
Average	Overall accuracy	0.3692	0.3774	0.3860	0.4343	0.4415	0.4025	**0.4423**	

Table 7
The Maximum Draw Down of all models for the rolling test in 2017.

Maximum Draw Down (MDD) of each month for rolling test in 2017							
	July	August	September	October	November	December	Average
Rule-based	0.2238	0.2473	0.2331	0.2461	0.1958	0.2220	0.2280
Multi-layer DNN	0.0231	0.0673	0.0190	0.0259	0.0342	0.0173	0.0852
LSTM	0.0182	0.0198	0.0230	0.0329	0.0161	0.0155	0.0183
1D-Conv. based RN	0.0149	0.0464	0.0203	0.0271	0.0293	0.0278	0.0627
2D-Conv. based RN	0.0193	0.0336	0.0215	**0.0186**	0.0192	0.0333	**0.0403**
GGSNN_CompleteGraph	0.0182	0.0203	0.0303	0.0486	0.0314	0.0233	0.1031
GGSNN	0.0182	0.0573	**0.0086**	0.0191	**0.0093**	0.0645	0.0707
AttGGSNN-GRUCell	0.0275	**0.0177**	0.0124	0.0283	0.0204	0.0165	0.0612
AttGGSNN-TCN	**0.0130**	0.0307	0.0286	0.0308	0.0120	**0.0120**	0.0561

We also observe that the MLP and LSTM models occasionally produce less profit than the rule-based method does. This indicates that DNN or LSTM has limitations and cannot be applied individually to all portfolio construction situations.

5. Conclusion and future work

We conducted a cross-correlation research on financial time-series data by using two proposed deep-learning-based models and used the results to develop a VQA-like system that can indicate the future conditions of financial markets and be a prototype robo-advisor to recommend portfolios for users.

The main proposed model, namely the attention-based GGSNN with a TCN block, learns the overall relationships among commodities from a graph that contains multiedge information. We then developed two improved relation networks that learn the relatively spatial and temporal relation between commodities through a 1D or 2D convolution. The models based on 1D and 2D convolution involve end-to-end training and multioutput classification tasks. The results indicate that the use of multiedge information and the attention mechanism in the aggregation phase of messages enhances the performance of a GGSNN. Moreover, the resampling and uniform sampling of our question–answer pairs solved the class imbalance problem and made the classifier more robust.

Also, Fusion can take place at diverse components within a prediction process. We elaborate the necessity of applying fusion in stock market, and we mainly focus on data fusion and model fusion in our research.

Certain aspects of the present study can be improved in future studies. First, the proposed attention-based GGSNN was unable to form a novel propagation mechanism for financial time-series data in each time step. This feature may be realized by adopting a dynamic graph. Second, in future studies, the VQA-like system can be extended to concatenate a language model for question embedding, and the overall model can be used for multilabel classification to obtain generalized answers. Third, the speed of the proposed model can be increased using sequential learning architecture, such as dilated RNNs, or a convolution-based approach in the graph updating phase. Last but not least, some research [50,51] started to use Reinforcement Learning for trading and portfolio management. We can find if there any possibility to combine their concepts with our model and make it better.

CRediT authorship contribution statement

Wei-Chia Huang: Conceptualization, Methodology, Software, Investigation, Writing – Original Draft. **Chiao-Ting Chen:** Software, Investigation, Writing – review & editing. **Chi Lee:** Writing – review & editing. **Fan-Hsuan Kuo:** Writing – review & editing. **Szu-Hao Huang:** Conceptualization, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contract MOST 111-2221-E-A49-125-MY3 and Contract MOST 111-2622-8-A49-013-TM1; and in part by the Financial Technology (FinTech) Innovation Research Center, National Yang Ming Chiao Tung University.

References

[1] Aytac Altan, Seçkin Karasu, The effect of kernel values in support vector machine to forecasting performance of financial time series and cognitive decision making. 4 (2019) 17-21.

[2] A. Tucker, S. Swift, X. Liu, Variable grouping in multivariate time series via correlation, IEEE Trans. Syst. Man Cybern. B 31 (2) (2001) 235–245.

[3] K.J. Kim, Financial time series forecasting using support vector machines, Neurocomputing 55 (1–2) (2003) 307–319.

[4] R. Choudhry, K. Garg, A hybrid machine learning system for stock market forecasting, World Acad. Sci. Eng. Technol. 39 (3) (2008) 315–318.

[5] P.-Y. Wang, C.-T. Chen, J.-W. Su, T.-Y. Wang, S.-H. Huang, Deep learning model for house price prediction using heterogeneous data analysis along with joint self-attention mechanism, IEEE Access 9 (2021) 55244–55259, http://dx.doi.org/10.1109/ACCESS.2021.3071306.

[6] Huang Szu-Hao, Yu-Hsiang Miao, Yi-Ting Hsiao, Novel deep reinforcement algorithm with adaptive sampling strategy for continuous portfolio optimization, IEEE Access 9 (2021) 77371–77385, http://dx.doi.org/10.1109/ACCESS.2021.3082186.

[7] Ji Hyun Jang, Jisang Yoon, Jungeun Kim, Jinmo Gu, Ha Young Kim, DeepOption: A novel option pricing framework based on deep learning with fused distilled data from multiple parametric methods, Inf. Fusion 70 (2021) 43–59, http://dx.doi.org/10.1016/j.inffus.2020.12.010.

[8] Ben Moews, J. Michael Herrmann, Gbenga Ibikunle, Lagged correlation-based deep learning for directional trend change prediction in financial time series, Expert Syst. Appl. 120 (2019) 197–206.

[9] Chen Yu-Fu, Szu-Hao Huang, Sentiment-influenced trading system based on multimodal deep reinforcement learning, Appl. Soft Comput., 112, 107788 http://dx.doi.org/10.1016/j.asoc.2021.107788.

[10] Li Lin, Feng Zhu, Hui Sun, Yiyi Hu, Yunyun Yang, Dawei Jin, Multi-source information fusion and deep-learning-based characteristics measurement for exploring the effects of peer engagement on stock price synchronicity, Inf. Fusion 69 (2021) 1–21, http://dx.doi.org/10.1016/j.inffus.2020.11.006.

[11] Shihua Luo, Cong Tian, Financial high-frequency time series forecasting based on sub-step grid search long short-term memory network, IEEE Access (8) (2020) 203183-203189, http://dx.doi.org/10.1109/ACCESS.2020.3037102.

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

[13] Alaa Sagheer, Mostafa Kotb, Time series forecasting of petroleum production using deep LSTM recurrent networks, Neurocomputing (ISSN: 0925-2312) 323 (2019) 203–213, http://dx.doi.org/10.1016/j.neucom.2018.09.082.

[14] Thomas Fischer, Christopher Krauss, Deep learning with long short-term memory networks for financial market predictions, European J. Oper. Res. (ISSN: 0377-2217) 270 (2) (2018) 654–669, http://dx.doi.org/10.1016/j.ejor.2017.11.054.

[15] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, A. Iosifidis, Using deep learning to detect price change indications in financial markets, in: 2017 25th European Signal Processing Conference, EUSIPCO, Kos, Greece, 2017, pp. 2511–2515, http://dx.doi.org/10.23919/EUSIPCO.2017.8081663.

[16] H. Mo, J. Wang, Return scaling cross-correlation forecasting by stochastic time strength neural network in financial market dynamics, Soft Comput. 22 (2018) (2018) 3097.

[17] Hyungbin Yun, Minhyeok Lee, Yeong Seon Kang, Junhee Seok, Portfolio management via two-stage deep learning with a joint cost, Expert Syst. Appl. (ISSN: 0957-4174) 143 (2020) 113041, http://dx.doi.org/10.1016/j.eswa.2019.113041.

[18] Nhi N.Y. Vo, Xuezhong He, Shaowu Liu, Guandong Xu, Deep learning for decision making and the optimization of socially responsible investments and portfolio, Decis. Support Syst. (ISSN: 0167-9236) 124 (2019) 113097, http://dx.doi.org/10.1016/j.dss.2019.113097.

[19] N. Peng, H. Poon, C. Quirk, K. Toutanova, W.T. Yih, Cross-sentence n-ary relation extraction with graph lstms, Trans. Assoc. Comput. Linguist. 5 (2017) 101–115.

[20] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.

[21] R. Dahlhaus, Graphical interaction models for multivariate time series, Metrika 51 (2) (2000) 157–172.

[22] M. Talih, N. Hengartner, Structural learning with time-varying components: tracking the cross-section of financial time series, J. R. Stat. Soc. Ser. B Stat. Methodol. 67 (3) (2005) 321–341.

[23] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, Cho-Jui Hsieh, Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 257–266, http://dx.doi.org/10.1145/3292500.3330925.

[24] A. Kapoor, X. Ben, L. Liu, B. Perozzi, M. Barnes, M.J. Blais, S. O'Banion, Examining COVID-19 forecasting using spatio-temporal graph neural networks, 2020, ArXiv, abs/2007.03113.

[25] Chen Lei, Jie Cao, Youquan Wang, Weichao Liang, Guixiang Zhu, Multi-view graph attention network for travel recommendation, Expert Syst. Appl. 191 (2022) 116234, http://dx.doi.org/10.1016/j.eswa.2021.116234.

[26] S. Liu, T. Li, H. Ding, et al., A hybrid method of recurrent neural network and graph neural network for next-period prescription prediction, Int. J. Mach. Learn. Cybern. 11 (2020) 2849–2856.

[27] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.

[28] W. Nie, M. Ren, A. Liu, Z. Mao, J. Nie, M-GCN: Multi-branch graph convolution network for 2D image-based on 3D model retrieval, IEEE Trans. Multimedia http://dx.doi.org/10.1109/TMM.2020.3006371.

[29] Wang Youquan, Jie Cao, Haicheng Tao, Graph convolutional network with multi-similarity attribute matrices fusion for node classification, Neural Comput. Appl. (2021) 1–11, http://dx.doi.org/10.1007/s00521-021-06429-1.

[30] Z. Cui, K. Henrickson, R. Ke, Y. Wang, Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, IEEE Trans. Intell. Transp. Syst. 21 (11) (2020) 4883–4894, http://dx.doi.org/10.1109/TITS.2019.2950416.

[31] Liang Bin, Hang Su, Lin Gui, Erik Cambria, Ruifeng Xu, Aspect-based sentiment analysis via affective knowledge enhanced graph convolutional networks, Knowl.-Based Syst. 235 (2022) 107643, http://dx.doi.org/10.1016/j.knosys.2021.107643.

[32] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, 2015, arXiv preprint arXiv:1511.05493.

[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.

[34] C. Shang, Q. Liu, K.S. Chen, J. Sun, J. Lu, J. Yi, J. Bi, Edge attention-based multi-relational graph convolutional networks, 2018, arXiv preprint arXiv:1802.04944.

[35] A. Santoro, D. Raposo, D.G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, T. Lillicrap, A simple neural network module for relational reasoning, in: Advances in Neural Information Processing Systems, 2017, pp. 4967–4976.

[36] P. Battaglia, R. Pascanu, M. Lai, D.J. Rezende, Interaction networks for learning about objects, relations and physics, in: Advances in Neural Information Processing Systems, 2016, pp. 4502–4510.

[37] B. Zhou, A. Andonian, A. Torralba, Temporal relational reasoning in videos, 2017, arXiv preprint arXiv:1711.08496.

[38] Xiyue Gao, Jun Chen, Zexing Zhan, Shuai Yang, Learning heterogeneous information network embeddings via relational triplet network, Neurocomputing (ISSN: 0925-2312) 412 (2020) 31–41, http://dx.doi.org/10.1016/j.neucom.2020.06.043.

[39] Wenbo Zheng, Chao Gou, Lan Yan, Shaocong Mo, Learning to classify: A flow-based relation network for encrypted traffic classification, in: Proceedings of the Web Conference 2020, Association for Computing Machinery, New York, NY, USA, 2020, pp. 13–22, http://dx.doi.org/10.1145/3366423.3380090.

[40] Zongyong Deng, Hao Liu, Geometry-attentive relational reasoning for robust facial landmark detection, Neurocomputing (ISSN: 0925-2312) (2020) http://dx.doi.org/10.1016/j.neucom.2020.06.126.

[41] M.-S. Cheong, M.-C. Wu, S.-H. Huang, Interpretable stock anomaly detection based on spatio-temporal relation networks with genetic algorithm, IEEE Access 9 (2021) 68302–68319, http://dx.doi.org/10.1109/ACCESS.2021.3077067.

[42] Christopher W. Anderson, Beracha Eli, Robustness of the headquarters-city effect on stock returns, J. Financ. Res. 31 (2008) 271–300, http://dx.doi.org/10.2139/ssrn.949648.

[43] J. Vernon Henderson, Yukako Ono, Where do manufacturing firms locate their headquarters? J. Urban Econ. 63 (2008) 431–450, http://dx.doi.org/10.1016/j.jue.2007.02.006.

[44] Lindsay Baran, Rachel Wilson, Whom you connect with matters: director networks and firm location, J. Financ. Res. 41 (2018) 113–147, http://dx.doi.org/10.1111/jfir.12141.

[45] F. Fama Eugene, Kenneth R. French, The cross-section of expected stock returns, J. Finance 47 (1992) 427–465, http://dx.doi.org/10.1111/j.1540-6261.1992.tb04398.x.

[46] F. Fama Eugene, Kenneth R. French, A five-factor asset pricing model, J. Financ. Econ. 116 (2015) 1–22, http://dx.doi.org/10.1016/j.jfineco.2014.10.010.

[47] Merton H. Miller, Franco Modigliani, Dividend policy, growth, and the valuation of shares, J. Bus. 34 (1961) 411–433, http://dx.doi.org/10.1086/294442.

[48] A.V.D. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, K. . . . . Kavukcuoglu, Wavenet: A generative model for raw audio, 2016, arXiv preprint arXiv:1609.03499.

[49] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018, arXiv preprint arXiv:1803.01271.

[50] Y.-Y. Chen, C.-T. Chen, C.-Y. Sang, Y.-C. Yang, S.-H. Huang, Adversarial attacks against reinforcement learning-based portfolio management strategy, IEEE Access 9 (2021) 50667–50685, http://dx.doi.org/10.1109/ACCESS.2021.3068768.

[51] C.-H. Kuo, C.-T. Chen, S.-J. Lin, S.-H. Huang, Improving generalization in reinforcement learning–based trading by using a generative adversarial market model, IEEE Access 9 (2021) 50738–50754, http://dx.doi.org/10.1109/ACCESS.2021.3068269.