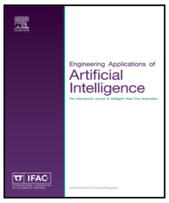
Engineering Applications of Artificial Intelligence 116 (2022) 105452

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



GCNET: Graph-based prediction of stock price movement using graph convolutional network

Alireza Jafari, Saman Haratizadeh [*]

Faculty of New Sciences and Technologies, University of Tehran, North Kargar Street, 1439957131 Tehran, Iran

ARTICLE INFO

Keywords:
Stock price prediction
Deep learning
Graph convolutional network
Semi-supervised learning
GCN
Graph-based stock forecasting

ABSTRACT

The importance of considering related stocks data for the prediction of stock price movement has been shown in many studies; however, advanced graphical techniques for modeling, embedding and analyzing the behavior of inter-related stocks have not been widely exploited for the prediction of stocks price movements yet. The main challenges in this domain are to find a way for modeling the existing relations among an arbitrary set of stocks and to exploit such a model for improving the prediction performance for those stocks. The most of existing methods in this domain rely on basic graph-analysis techniques, with limited prediction power, and suffer from a lack of generality and flexibility. In this paper, we introduce a novel framework, called GCNET that models the relations among an arbitrary set of stocks as a graph structure called influence network and uses a set of history-based prediction models to infer plausible initial labels for a subset of the stock nodes in the graph. Finally, GCNET uses the Graph Convolutional Network algorithm to analyze this partially labeled graph and predicts the next price direction of movement for each stock in the graph.

GCNET is a general prediction framework that can be applied for the prediction of the price fluctuations of interacting stocks based on their historical data. Our experiments and evaluations on a set of stocks from the NASDAQ index demonstrate that GCNET improves the performance of the state-of-the-art algorithms in terms of Accuracy and Matthew's Correlation Coefficient by at least 1.5% and 2%, respectively.

1. Introduction

Predicting the stock prices and fluctuations of stock prices has been of interest for decades since it can be of great value for investors who need to decide how to invest in the market (Rather et al., 2017; Soni, 2011). Traditional stock prediction approaches are categorized into technical analysis and fundamental analysis. While fundamental analysis focuses on the intrinsic value of stocks and is mainly used by human experts, in technical analysis, the history of stock prices and price indicators are analyzed to extract useful patterns for the prediction of the future behavior of the stocks prices.

In past decades, technical analysis has been widely used (Bustos and Pomares-Quimbaya, 2020). Researchers have applied machine learning techniques to develop a wide variety of prediction models, including regression and classification systems, for stock price forecasting. Especially during the past few years, along with the advancements in the deep learning domain, sophisticated deep models have been developed for market forecasting, outperforming their traditional predecessors (Hoseinzade and Haratizadeh, 2019; Li and Liao, 2017; Patel et al., 2015; Gunduz et al., 2017). However, in most of these studies, the prediction models use historical data from the past prices of a single stock to predict the future of that stock's price and ignore the relationships between stocks as a source of information.

Since markets and stocks are naturally connected entities having interactions with each other, one can expect that the behavior of one entity can affect the others. This means that analyzing the information from stocks that are somehow connected to a certain target stock, can probably help us to predict the future behavior of that stock more accurately. Following this idea, in this paper, we are motivated to introduce a framework for extracting the hidden relations among stocks and exploiting them for the prediction of stock price fluctuations. The suggested framework needs to be general enough to be applicable for any set of stocks, and its prediction mechanism must be able to exploit the extracted relations among stocks for efficiently aggregating relevant information in order to make accurate predictions.

Although some recent studies have used historical data from other possibly related stocks/markets as well to train prediction models for a target stock/market (Hoseinzade and Haratizadeh, 2019; Zhong and Enke, 2017; Gunduz et al., 2017) they still have some limitations: They usually rely on an expert's advice to select a preferably small set of related stocks/markets to the target entity. Also, most of the existing models in this category are not flexible enough to explicitly model and use the direct and indirect relationships among entities or to reflect the already known degrees of relevance between them. Another drawback

[*] Corresponding author.
E-mail addresses: alireza.jafari7@ut.ac.ir (A. Jafari), haratizadeh@ut.ac.ir (S. Haratizadeh).

https://doi.org/10.1016/j.engappai.2022.105452
Received 20 May 2022; Received in revised form 25 August 2022; Accepted 13 September 2022
Available online 7 October 2022

of this approach is that the architectures of these models are usually task-dependent and are designed for prediction over a fixed set of related entities and they may not be easily applicable for other, possibly larger, sets of entities.

A natural approach to model the inter-relations among a set of entities is to use graphical analysis that can discover the hidden relationship between stocks from the graph structure, unlike previous works, which use the manual selection of related stocks. Graph-based modeling and analysis of data is a standard technique in many domains such as social network analysis or recommendation systems; however, it has not been widely used for modeling relations among financial entities and its application in this domain has been limited to a few studies (Yin et al., 2022; Kia et al., 2020; Long et al., 2020; Kia et al., 2018; Kim and Sayama, 2017).

The common approach in these studies is to construct a (usually correlation-based) graph structure reflecting some kind of relation among entities and then propagate a set of already known labels to other nodes through the paths that exist among nodes in the graph. A graphical model in this approach explicitly represents the stock/index relations and it can be easily generated for different sets of such financial entities, so, this class of prediction models are usually more explainable and flexible than the traditional fixed-input neural networks. However they still suffer from some important flaws. Firstly, they rely on a set of known labels for some entities while that may not be available in all settings. Secondly, their prediction mechanism is solely based on the structure of the graph and usually ignores any local information about each entity's current state for prediction of its. Finally, these algorithms in their prediction step usually use neighbor-based approaches that cannot be trained or optimized based on the current state of a node and its neighbors or the structure of the current underlying graph. This lack of adaptability makes the performance of the prediction step extremely dependent on the initial structure of the graph, which limits the prediction performance especially when there are hidden interactions among entities that are not reflected in the graphical model.

Fortunately, in recent years there have been significant developments in the field of graph analysis using Graph Neural Networks. These techniques are able to analyze graph-structured data in order to generate new representations of each node while considering node features. In this paper, we introduce GCNET, GCN on NETwork of stocks, a general framework for the prediction of stock price fluctuations that formulates this classic problem as a label prediction task in a graph. It models the price historical data of a set of inter-related stocks (or markets) along with the observed degrees of mutual relations among them as a graph and creates beneficial aggregation for each node by considering its related neighbor stocks. GCNET uses a deep graphical semi-supervised learning algorithm to generate a model for predicting the next price (or index) fluctuation for each stock (or market). The main contributions of this paper can be summarized as follows:

- We introduce a novel framework for efficiently aggregating useful information from different stocks in order to make more accurate predictions about the future stock price movements.
- We introduce a novel approach for modeling the relations among stocks as a so-called influence network that improves its overall prediction performance.
- We introduce a novel approach to formulate the prediction of stock price fluctuations as a label prediction task over a graphical model of inter-related financial entities.
- We suggest a method that can exploit any set of history-based predictors to generate the initial labels required by its semi-supervised learning process and so it is applicable in settings where there is no already known label/class for any node/stock.
- A comprehensive set of experiments on two sets of hundred stocks shows that GCNET predicts the next price fluctuations for the stocks in the graph with a significantly better performance compared to the state-of-the-art baselines.

In the next section, we will review the related work. Our network construction method and prediction algorithm are explained in Section 3. Section 4 presents detailed information about the dataset used in our experiments, the experimental setup, the results, and a discussion about the observations. Finally, we conclude the paper in Section 5.

2. Related works

Many different machine learning techniques have been used for the prediction of stock markets in the past decades. Most of these techniques rely on analyzing the historical data of a single stock data to extract useful patterns for predicting its future behavior. Kara et al. (2011) used Artificial neural network (ANN) and Support vector machine (SVM) to predict the direction of movement of the Istanbul Stock Exchange National 100 Index and showed that ANN predicts the index significantly better than SVM. Zhong and Enke (2017) used ANN to predict S&P500 in their model and showed that using Principal component analysis (PCA) to construct new representations for the initial feature vectors can improve the prediction accuracy of the ANN-based model. Arévalo et al. (2016) examined artificial neural networks with different structures and observed that a deep neural network with five hidden layers achieves the highest accuracy in predicting the direction of movement for a single stock's price. Long Short-Term Memory (LSTM) is one of the most popular models used for time series prediction. Nelson et al. (2017) used LSTM and Multilayer perceptron (MLP) for forecasting the Brazilian stock market. The results showed the superiority of LSTM over MLP. In a more recent study, LSTM has been combined with an attention mechanism to predict market shares based on the price information from the past few days (Feng et al., 2019). Although only one stock data is used in this model to predict, in more recent studies, researchers have emphasized the importance of considering related stocks by trying to use other stocks in their model training using simple techniques. Shah et al. (2021) analyzed historical data of different stocks by Bi-directional Long Short-Term Memory and predict stock price trends. They also presented a framework for depth and time calculation learning faster than the one-directional approach. Gite et al. (2021) and Mehta et al. (2021) used LSTM to predict the Indian stock market and tried to consider sentiments derived by users that could affect the trading patterns of traders. They demonstrated that aggregating sentiment information along with technical analysis can improve the stock movement prediction accuracy.

Convolution neural network, CNN, is another class of deep learning algorithms used in stock market forecasting while its ability to extract high-level features from price data has been studied by several researchers (Gunduz et al., 2017; Hoseinzade and Haratizadeh, 2019; Di Persio and Honchar, 2016). In one study, Di Persio and Honchar (2016) different kinds of deep neural networks including ANN, LSTM, and CNN were used to predict the S&P500 index, and the results showed that CNN outperforms other deep models in prediction. Gunduz et al. (2017) used CNN to predict the market index of the Istanbul Stock Exchange. They used a handmade set of stocks to input their model to consider related stocks. Wu et al. (2021b) and Wu et al. (2021a) predicted some American and Taiwanese stocks' direction of movement using a deep CNN model and a combined model of LSTM and CNN layers, respectively. They constructed a sequence array of historical data and leading indicators containing options and futures to feed their model. They tried to use information from the futures as a useful source that can increase prediction accuracy, by considering some leading indicators. Hoseinzade and Haratizadeh (2019) predicted well-known stock market indices in the United States using CNN. They showed that 2D and 3D CNNs could be applied to combine information from a fixed set of related time series in order to successfully predict their future behavior. Although the above models have yielded acceptable results, most of them do not use stock relationships as a valuable source of information and the rest are limited to using the handful of stocks that the authors guess are relevant.

Table 1
Summary of the related work.

Author/Year	Network modeling	Information sources	Prediction Method
Kara et al. (2011)	No	Single market	ANN, SVM
Park and Shin (2013)	Yes	Multiple related stocks	Label propagation
Patel et al. (2015)	No	Single market	ANN, SVR, RF
Di Persio and Honchar (2016)	No	Single market	MLP, RNN, CNN
Zhong and Enke (2017)	No	Single market	PCA+ANN
Nelson et al. (2017)	No	Single stock	LSTM, MLP
Gunduz et al. (2017)	No	Multiple related stocks	CNN
Kim and Sayama (2017)	Yes	Multiple related stocks	ARIMA
Qin et al. (2017)	No	Single stock	LSTM
Kia et al. (2018)	Yes	Multiple related markets	SVM
Feng et al. (2019)	No	Single stock	LSTM
Hoseinzade and Haratizadeh (2019)	No	Multiple related entities	CNN
Kim et al. (2019)	Yes	Multiple related stocks	LSTM+GAT
Long et al. (2020)	Yes	Multiple related stocks	node2vec+LSTM
Kia et al. (2020)	Yes	Multiple related markets	PageRank
Shah et al. (2021)	No	Multiple related stocks	LSTM
Wu et al. (2021a)	No	Single stock	LSTM+CNN
Wu et al. (2021b)	No	Single stock	CNN
Mehta et al. (2021)	No	Single stock	LSTM, SVM, MLP
Yin et al. (2022)	Yes	Multiple related stocks	LSTM
Wu et al. (2022)	No	Single stock	Struc2vec, RNN

Another class of algorithms for financial time series prediction is the Graph-based methods (Yin et al., 2022), Kia et al. (2020), Long et al. (2020), Kia et al. (2018), Kim and Sayama (2017), and Shin et al. (2013). The majority of these methods model the relations of the entities as correlation graphs and use models to predict that cannot be easily generalized to other sets of stocks or markets. Park and Shin (2013) created a network of nodes representing the stocks in the Korean Stock Exchange and some commodity price time series based on the similarity of their feature vectors. They assigned the known labels based on the amount of variation in the 5-day moving average of stock prices and predicted the direction of movement for the Korean Stock Exchange using a label propagation algorithm. (Kim and Sayama, 2017) used network analysis and showed that graph structures can reveal important information about the future of stock prices. Taking into account the parameters of the network, they developed an AutoRegressive Integrated Moving Average (ARIMA) model and predicted the value of the S&p500 index. In Kia et al. (2018) a graph structure is introduced in which the nodes represent the markets, and the edges represent strong correlations between the market index time series. The nodes whose corresponding indices are already known, are labeled while the nodes whose corresponding indices are still unknown, due to the time differences between different time zones, are predicted using a label spreading algorithm. Long et al. (2020) create a knowledge graph representing different companies and apply the node2vec algorithm to select the relevant stocks of the target for constructing helpful embedding (Grover and Leskovec, 2016). They use the similarities between the resulting embeddings as a quantitative measure of the bilateral relations between stocks. They weight each stock using these similarities and train an LSTM model for prediction without exploiting the power of network analysis in the forecasting process.

In Kia et al. (2020) an association rule mining technique is used for defining the weights of edges in a graph of markets, and to predict the unknown labels in the graph, a variation of PageRank algorithm is used. As we mentioned before, the simple label prediction algorithms used in this class of methods, not only rely on the existence of already known labels but also they fail to model possibly complex relations among the internal states of the nodes, their inter-relations, and their labels. Also, despite the fact that the performance of these algorithms is highly dependent on the structure of the underlying graph, the graph construction process needs to be subject to further studies in this domain. Like most of the mentioned models, Yin et al. (2022) use the correlation graph to predict the stock price of the Chinese stock market. They create new representations by attention mechanism and then use these vectors to train and predict the LSTM model. This class of models relies entirely on underlying graph structures, that are mostly defined based on historical price correlations among stocks, while other possible graph structures have received less attention.

As a modern node embedding and label prediction algorithm, Graph Convolutional Network, GCN, has been recently introduced. GCN is an extension to CNN that is able to handle graphical data (Kipf and Welling, 2016). It is a Graph Neural Network (GNN) that takes the graph structure and node features as inputs and aggregates and transforms information from neighbors of each node to create a new representation for that node which is more informative for the prediction of a target variable like the label or class of the node. In this context, nodes can be stocks and edges indicate the mutual importance of stocks. GCN can construct representations of related stocks by analyzing the underlying graph while considering both stock relationships and the initial feature vector of each stock. GCN has recently received extensive attention because of its outstanding performance on node classification tasks in various fields, including recommender systems (Hekmatfar et al., 2021), traffic prediction (Zhao et al., 2019) and, cancer diagnosis (Zhou et al., 2019). In the field of stock market prediction, GNNs are limited to a few studies due to their recent emergence (Kim et al., 2019; Hu et al., 2018). Kim et al. (2019) used a graph attention network to forecast stocks. For creating their graph, they used a classic dataset of Wikidata in previous work (Vrandečić and Krötzsch, 2014), and their infrastructure network was very simple. GNN has been used in a few studies for representational learning from textual data available in social media, news articles, and blogs (Hu et al., 2018) but they have not been widely applied in the field of graph-based financial prediction to analyze financial graphical models yet.

Table 1 summarizes the explained papers in terms of the type of data source, initial variable set, and the prediction method they use. While there has been a tendency towards using deep learning models in more recent papers, in most of the research a sing stock's data is being processed to generate prediction models, while only a few papers have aggregated information from multiple related stocks/markets using a graph-based approach in their prediction process.

In the next section, we will introduce GCNET, as a novel framework for graph-based prediction of the direction of stocks price movements. GCNET resolves the shortcomings of the existing methods by introducing a novel approach to modeling the prediction of the price fluctuations for a set of inter-connected stocks in a market as a semi-supervised label prediction process using GCN. It also presents a new technique for constructing the graphical model of the data and introduces an effective method for labeling a subset of the graph nodes using reliable history-based predictions as well.

3. Model description

In this section, we are going to explain our suggested semi-supervised prediction algorithm called GCNET. As summarized in Algorithm 1, GCNET first models the stock market as a complex network structure of the stocks' historical data. Using the structure of the created graph, for each day of the stock market separately, we create a graph of nodes, which represent stocks and we have introduced a novel technique for connecting and weighting graph edges. Also, each node contains a feature vector of technical indicators from the specified day of corresponding stock. In the next step, we assign a set of initial labels to a portion of the nodes of test day using a set of reliable predictions made by a so-called PLD method. Each label in this step represents a first guess about the possible next price fluctuation for the corresponding stock. The algorithm then uses a GCN technique to process the resulting partially labeled graph, in order to refine the initial labels and also predict the labels for the unlabeled subset of the nodes. The final labels form the algorithm's predictions for the next-day fluctuations of the stocks. In the following subsections, we explain the details of the GCNET steps.

Algorithm 1 GCNET

```
Input :
    Dataset (Price history for m stocks)
Output :
    List of predicted labels

1: $G \leftarrow$ Generate_Influence_Graph(Dataset)
2: $G \leftarrow$ PLD(Dataset,G)                ▷ label a subset of the nodes
3: $G \leftarrow$ Add node feature vectors to G
4: $Predictor \leftarrow$Train GCN on G
5: $L \leftarrow Predictor(G)$                 ▷ predict labels using the trained GCN
6: return (L)
```

3.1. Network construction

In this section, we introduce a graph generation algorithm for constructing a novel so-called influence network, which is designed to serve the needs of the subsequent label prediction mechanism used in the GCNET framework. The steps of the graph construction procedure are summarized in Algorithm 2 and Fig. 1 This algorithm tries to generate a network containing paths through which sharing information among nodes may improve the performance of the target prediction task for each node. In the following sub-sections, we explain the details of the graph structure and the algorithm used for generating it.

As the algorithm shows, the nodes represent stocks (line 3 of the algorithm). If there are m stocks in the dataset: $Stocks = \{s_1, s_2, \dots, s_m\}$, we denote the graph as $G = (V, E)$, in which $V = \{v_{s_1}, v_{s_2}, \dots, v_{s_m}\}$ is the set of nodes and the set E is the set of edges. Each edge $e_{i,j}$ connects the pair of nodes v_{s_i} and v_{s_j}, where $i \neq j$. The edges are weighted and we denote the weight of the edge $e_{i,j}$ as $w_{i,j}$. To add edges to the graph, GCNET first calculates an influence score for each pair of nodes. To do this, it needs to split the data into train and validation parts (lines 4–5). This score estimates how useful is to combine the historical information from those stocks for the prediction of each one's future fluctuation. If this aggregation of information has a positive influence on the prediction performance, then an edge connecting that pair of nodes is added to the graph, with a weight reflecting the amount of that influence.

GCNET needs to build several simple prediction models to construct the influence graph with n nodes. For each target stock, one model is trained using that stock's historical data, and $n-1$ models are trained, each using the historical data from the target stock and one other stock in the graph. To achieve a reasonable computation time for training the required prediction models we use Quadratic Discriminant Analysis that is a fast and effective history based learning algorithm. Quadratic discriminant analysis (QDA) is a well-known statistical classification technique that is a variant of linear discriminant analysis that allows for non-linear separation of data. QDA classifier is attractive because it has closed-form solutions that can be easily computed, work well in practice, and have no hyper-parameters to tune. QDA finds an efficient decision rule when a linear discriminant procedure is not sufficient to separate the groups under study or when the covariance matrices are not equal among groups (James et al., 2013).

QDA assumes multivariate data, following a normal distribution, and uses the Bayes theorem to calculate the probability a sample x belongs to class k. However, a separate covariance matrix Σ_k, is assumed for each class, $k = 1, 2, \dots, k_n$ (n = number of classes), yielding the quadratic discriminant function as:

$$\delta_k(x) = -\frac{1}{2} log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) + log\pi_k \tag{1}$$

where δ_k is the score value of discriminant for class k, Σ_k is the covariance matrix and π_k is the prior probability of the kth population. Class Clf, which sample x belongs to is then predicted as:

$$Clf(x) = \arg\max_k \delta_k(x) \tag{2}$$

To train the models, for each stock s_i a feature vector v_i is extracted from its historical data based on the set of features summarized in Table 2. Also, for each pair of stock s_i and s_j an average feature vector is defined as $v_{ij} = v_{ji} = average(v_i, v_j)$. Then GCNET uses QDA to train four prediction models P_i, P_j, P_{ij} and P_{ji} for each pair s_i, s_j of stocks. P_i denotes the model predicting the next price movement for stock s_i using v_i as input, while P_{ij} uses v_{ij} as the input of the model to predict the next price movement for stock s_i (Similar definitions hold for P_j, P_{ji}).

Suppose that Acc_i, Acc_{ij}, denote the prediction accuracy of P_i, P_{ij} for stock s_i while Acc_j and Acc_{ji} represent the accuracy of P_j, P_{ji} in prediction of the next price movement for s_j all measured on the validation data (lines 8–13 of the algorithm). The $influence_{\{i,j\}}$ is then defined as follows (line 14) :

$$influence_{\{i,j\}} = \frac{(Acc_{ij} - Acc_i) + (Acc_{ji} - Acc_j)}{2} \tag{3}$$

Clearly, for a pair of stocks s_i and s_j, $influence_{\{i,j\}}$ reflects the average improvement in prediction performance achieved for each stock by using the other stock's information as well in the prediction model. Based on this measure then the weight of the edge connecting the corresponding nodes in the graph is calculated as follows (lines 15–18):

$$w_{i,j} = w_{j,i} = \max(influence_{\{i,j\}}, 0) \tag{4}$$

In other words, the weight of the edge between two stock nodes represents how much the aggregation of information between those two nodes we expect to boost the performance of the resulting predictions for each stock. The intuition of using this technique in GCNET comes from the behavior of the aggregation step in the GNN algorithms such as GCN. Since this graph is going to be ultimately used by GCN, the edges form the paths over which information from neighboring nodes are aggregated by GCN before being used for the final label prediction. The explained heuristic tries to directly estimate the usefulness of such aggregation for the target prediction task, for each pair of nodes. If the aggregation seems to be useful then an edge is added with an appropriate weight, while the edge is dropped otherwise.

In the final step, GCNET removes the edges with the lowest weight values one by one, and continues to do so until removing the next edge will make the graph disconnected (lines 20–24). This sparsification process keeps the strong connections in the graph and removes the unreliable low-weight edges that may introduce noise to the later

Algorithm 2 Generate Influence Graph

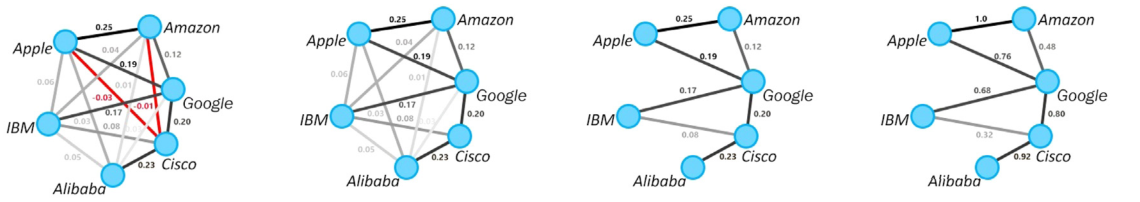
```
Input :
  Dataset (Price history for m stocks)
Output :
  Graph G(V,E)

1:  $V \leftarrow \{\}, E \leftarrow \{\}$
2:  for each stock $s_i$ do
3:      $V \leftarrow V \bigcup \{n_i\}$
4:      $X^T_{s_i}, X^V_{s_i} \leftarrow split\ data$(matrix of features of $s_i$)        ▷ Split the training and validation parts of the data
5:      $F^T_{s_i}, F^V_{s_i} \leftarrow split\ data$(vector of past price fluctuations of $s_i$)
6:  end for
7:  for each pair of stocks $(s_i, s_j)$ do
8:      $Acc_i \leftarrow QDA_i.fit(X^T_{s_i}, F^T_{s_i}).score(X^V_{s_i}, F^V_{s_i})$
9:      $Acc_j \leftarrow QDA_j.fit(X^T_{s_j}, F^T_{s_j}).score(X^V_{s_j}, F^V_{s_j})$
10:     $X_{Processed} \leftarrow 1/2(X_{s_i} + X_{s_j})$        ▷ Averag the two feature vectors to combine information
11:     $X^T_{Processed}, X^V_{Processed} \leftarrow split\ data(X_{Processed})$
12:     $Acc_{ij} \leftarrow QDA_{P_{ij}}.fit(X^T_{Processed}, F^T_{s_i}).score(X^V_{Processed}, F^V_{s_i})$
13:     $Acc_{ji} \leftarrow QDA_{P_{ji}}.fit(X^T_{Processed}, F^T_{s_j}).score(X^V_{Processed}, F^V_{s_j})$
14:     $Influence_{i,j} = 1/2((Acc_{ij} - Acc_i) + (Acc_{ji} - Acc_j))$        ▷ Average prediction improvement for the two stocks
15:     if $Influence_{i,j} > 0$ then
16:         $E \leftarrow E \bigcup \{e_{i,j}\}$
17:         $w_{i,j} \leftarrow Influence_{i,j}$        ▷ Set the weight of $e_{i,j}$
18:     end if
19: end for
20: while G is connected do        ▷ Remove the edges with small weights
21:     $e_{last} \leftarrow$ the edge with the smallest weight in $E$
22:     $E \leftarrow E - \{e_{i,j}\}$
23: end while
24: $E \leftarrow E \bigcup \{e_{last}\}$        ▷ Restore the last edge to make the graph connected again
25: return $(G(V,E))$
```

steps of the algorithm. In addition, using a sparser graph improves the efficiency of the GCNET in learning the final label prediction model.

The explained influence network structure can be constructed for each single day of the market, whether it is a day in the past or it is the target prediction day. GCNET models the prediction of the next stocks' price movements as a label prediction task on the target prediction day's graph, where a label of a node represents the corresponding stock's price fluctuation (rise/fall). Since for a day in the past, those fluctuations are already known, its corresponding graph can be easily labeled based on the observed price fluctuations. However for a test/prediction day, the true labels (price fluctuations of the stocks) are not known, yet GCNET tries to assign labels to some of the nodes of that graph too, using a procedure that we will explain in the next section.

3.2. Initial label assignment process

As mentioned earlier, GCNET uses a semi-supervised style of learning to generate a label prediction model, that uses a partially labeled graphical representation of data as input. In this setting, an existing label for a graph node is supposed to represent the already-known next price movement for its corresponding stock, while the label prediction's task is to predict the next price movement for stocks whose labels are unknown. However, for a test/prediction day t, the next price fluctuations are not known for any stock, so the graph cannot be partially labeled using the real price movements of any subset of the nodes. To handle this problem GCNET assigns initial labels to some of the nodes of such a test-day network, by trying to make reliable predictions about the future movements of their corresponding stocks



state(a): Define the weight of each possible edge to be the influence score of its corresponding pair of nodes

state(b): Remove all the edges with negative weights

state(c): Prune the edges with small weights

state(d): Normalize the edge weights

Fig. 1. Graph construction steps.

using its so-called PLD mechanism. Although those initial labels can be refined and changed by the algorithm in the next steps, assigning those initial labels injects some intuition about the possible future of the market into the graph to improve the performance of the subsequent embedding and prediction steps of GCNET.

3.2.1. *Plausible label discovery(PLD)*

Here we introduce the PLD algorithm that assigns some initial plausible predicted labels to a subset of the nodes in the test-day graph. The main steps of the PLD procedure are presented in Algorithm 3. PLD trains a set of basic history-based prediction algorithms (PA), as defined in line 1 of the algorithm, to predict the labels for the nodes of the graph on a daily basis.

Suppose that for the prediction of the rise/fall in stock prices for a target day t we have a set of labeled instances for each stock s_i for a period of T days before t in historical data. PLD splits the dataset into training and validation sets, such that the validation set VS_{s_i} contains samples belonging to a period of d days before t and the training set TS_{s_i} contains older samples (lines 3–4).

For each stock s_i, then a prediction model is generated by each of the basic algorithms using the training set TS_{s_i}. PLD then scores the predictors for each stock by evaluating their prediction performance over the validation set of that stock (lines 5–7). The reason for selecting the most recent labeled instances as the validation set is to make the scores assigned to the predictors more reliable by using the instances that are closer to the target day for evaluation. Also, to put more emphasis on predictions closer to the target day t when evaluating the predictors over the validation set, we use a weighted scoring method that gives higher weights to the accuracy of predictions made for the more recent days. The scoring method has been defined in Eq. (5):

$$score^{j}_{s_i} = \sum_{i=0}^{d-1} a_{t-1-i} \times (1-c)^{i}$$

$$where$$

$$a_l = \begin{cases} 1 & \text{if predictor j predicts the class of} \\ & \quad \text{instance } ins^{l}_{s_i} \text{ correctly} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$and$$

$$0 < c \ll 1$$

Once all the predictors are scored, we select the predictor with the highest score, $best_{s_i}$, for each stock s_i. GCNET assumes that $best_{s_i}$ will have the most reliable prediction, among predictors of stock s_i, for the price movement of s_i on the target day t. Here we call this score the predictability score of the stock node s_i (line 8) (6):

$$predictability_{s_i} = score^{best_{s_i}}_{s_i} \tag{6}$$

To select a node for initial labeling, in addition to its predictability score, GCNET also considers how dense the graph is in the neighborhood of that node (line 10). Intuitively, nodes that are located in denser neighborhoods are possibly better candidates for initial labeling as their label information can be shared through several paths with many other nodes in the later embedding steps of the algorithm.

To evaluate the neighborhood density of each node, we use the weighted local clustering coefficient (Saramäki et al., 2007) to define the density score for a node s_i as:

$$density_{s_i} = \frac{1}{deg(i)(deg(i)-1)} \sum_{j,k} (\hat{w}_{i,j}\hat{w}_{i,k}\hat{w}_{j,k})^{1/3} \tag{7}$$

Where the edge weights are normalized by the maximum weight in the network, $\hat{w}_{i,j} = w_{i,j}/max(w)$, and the contribution of each triangle depends on all of its edge weights. This measure assigns high scores to the nodes located at the center of dense neighborhoods and lower scores to the nodes in whose neighborhood strong edges are rare. The final score assigned to each node is defined by the product of its predictability and density score (line 11) as defined in (8):

$$privilege_{s_i} = predictability_{s_i} * density_{s_i} \tag{8}$$

In the final step, n% of the nodes with the highest node privilege are selected to be assigned initial labels (lines 13–16). The assigned label to each selected node is the prediction of its corresponding stock's best predictor, $best_{s_i}$ for the target day: if the stock node s_i is among the top n% nodes with the highest privilege values and its best predictor predicts a rise/fall in its price it will be labeled with $+1/-1$. However, if the node's privilege score is not among the top n% scores then it will be left unlabeled.

The set of the basic algorithms we have used in PLD includes Neural Network, Linear Discriminant Analysis, Decision Tree, Naive Bayes, Quadratic Discriminant Analysis, Random Forest, and Multilayer Perceptron. The input features used to represent the instances for training these models are the same features used in the network construction step, as summarized in Table 2.

Please note that the explained model training and evaluation process is repeated for each target day, so, the nodes that are labeled by PLD, are not necessarily the same for different target days.

3.3. *Price movement prediction*

In this phase, node features are added to the partially labeled graph of stocks generated in previous steps, and the resulting graph is used to train a Graph Convolutional Network model for the prediction of the stock price movements as final node labels of the graph. We first present a brief background about the Graph Convolutional Network model and then we will explain how a training instance for a prediction day is generated to be used by the GCN model for the prediction of the next price fluctuations, and also we present the details of the training and prediction procedure.

3.3.1. *GCN background*

Graph Convolutional Network (GCN) is a framework for representation learning in graphs and is derived from graph signal processing theory. GCN can be applied directly on graph structured data to extract informative representations for each node by aggregating information from its neighbors in depth d. GCN can be considered as a Laplacian smoothing operator for node features over graph structures. The architecture of GCN consists of a series of convolutional layers. Each layer of GCN integrates information from the neighbors of each node to update its representation according to the structure of all nodes and edges in the graph by the Laplacian matrix.

In spectral theory, the convolution operation is defined in the Fourier domain by computing the eigendecomposition of the graph Laplacian. The operation can be defined as the multiplication of a signal $x \in \mathbb{R}^N$ (a scalar for each node) with a filter $g_\theta = \text{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^N$:

$$g_\theta \star x = \mathbf{U} g_\theta(\Lambda)\mathbf{U}^T x \tag{9}$$

where $\mathbf{U}$ is the matrix of eigenvectors of the normalized graph Laplacian (L), with a diagonal matrix of its eigenvalues Λ. Eq. (9) describes a spectral convolution filter g_θ used for graph data x. This operation results in potentially intense computations. (Hammond et al., 2011) approximated spectral filters in terms of Chebyshev polynomials and Kipf and Welling (2016) limit the layer-wise convolution operation to K = 1 to alleviate the problem of overfitting on local neighborhood structures for graphs with very wide node degree distributions and Introduce the Graph Convolution Network. Also, GCN simply transform Eq. (9) as a fully connected layer with a built-in convolution filter, which is written as the following equation:

$$H^{l+1} = ReLU(\hat{A}H^{l}W^{l}) \tag{10}$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Here, $\tilde{A} = A + I_N$, in which $\tilde{A}$ is the adjacency matrix of the graph G, with added self-connections, I_N is the identity matrix, $\tilde{D} = \sum_j \tilde{A}_{ij}$ and W^l is a layer-specific train able weight matrix.

Algorithm 3 PLD procedure for the target day **t**

```
Input :
  Training Data (training instances for all stocks)
  Graph G(V,E)
Output :
  Partially labeled graph G(V,E)

1: $PA \leftarrow \{pa^1, pa^2, ..., pa^k\}$    ▷ Prediction algorithms
2: for each stock $s_i$ do
3:     $VS_{s_i} \leftarrow \{ins_{s_i}^{t-1}, ins_{s_i}^{t-2}, ..., ins_{s_i}^{t-d}\}$    ▷ d instances for validation
4:     $TS_{s_i} \leftarrow \{ins_{s_i}^{t-d-1}, ins_{s_i}^{t-d-2}, ..., ins_{s_i}^{t-T}\}$    ▷ The rest for train
5:     for each algorithm $pa^j$ in $PA$ do
6:         $score_{s_i}^j \leftarrow pa^j.train(TS_{s_i}).evaluate(VS_{s_i})$    ▷ Use $pa^j$ to train and evaluate a predictor for stock $s_i$
7:     end for
8:     $best_{s_i} \leftarrow$ The best trained predictor for $s_i$    ▷ The one with the highest score
9:     $predictability_{s_i} \leftarrow score_{s_i}^{best s_i}$
10:    $density_{s_i} \leftarrow weighted\_local\_clustering\_coefficient(G, node_{s_i})$
11:    $privilege_{s_i} \leftarrow predictability_{s_i} * density_{s_i}$
12: end for
13: $TP \leftarrow$ Top $n\%$ stocks $s_i$ with the highest $privilege_{s_i}$    ▷ The selected nodes to be assigned initial labels
14: for each stock $s_k$ in $TP$ do
15:     $lablel_{s_k} \leftarrow$ price fluctuation predicted by predictor $best_{s_k}$ for the target day t of stock $s_k$
16: end for
17: return (G(V,E))    ▷ Partially labeled g raph
```

Table 2
Technical indicators and price information used by basic prediction algorithms in PLD.

Indicator	Description	Indicator	Description
OP	Open price	RSI-S	RSI indicator signal
HP	High price	BB-S	Bollinger bands indicator signal
LP	Low price	MACD-S	MACD indicator signal
CP	Close price	SAR-S	SAR indicator signal
Volume	Trading volume	ADX-S	ADX indicator signal
CCI	Commodity Channel Index	S-S	Stochastic indicator signal
SAR	Stop and reverse index	MFI-S	MFI indicator signal
ADX	Average directional movement	CCI-S	CCI indicator signal
MFI	Money flow index	V-S	Sign(Volume -Avg(last 5 days))
RSI	Relative strength Index	CPOP-S	Sign(CP-OP)
SK	Slow stochastic %K	CPCPY-S	Sign(CP-Closing price yesterday)
SD	Slow stochastic %D		

3.3.2. *Generating the prediction day instances*

Although one can choose to train a label prediction model for a target prediction day only by using its corresponding partially labeled graph, such a policy may lead to training a sub-optimal prediction model. The reason is that the initial labels in the prediction day's graph, based on which the model would be trained, are scarce and also prone to noise as they represent the predictions made by PLD, not the real observed price fluctuations. To address this concern, GCNET, generates training instances that are composed of five graphs: the graph of the target day itself, which is partially labeled by PLD, along with four graphs representing the four days preceding the target prediction day, that are fully labeled based on the true price fluctuations observed on those days. The structures of all these networks are the same as the influence graph, while each stock has a node in each of those graphs with possibly different vectors representing the state of that stock on the corresponding day (Fig. 2). Node feature vectors are composed of a set of technical indicator signals and the stock's last day price information. The complete set of features used for representing each node has been summarized in Table 3. Such an instance is then used by a Graph Convolutional Network to train a label prediction model and predict the final labels of the prediction day's graph nodes.

3.3.3. *Training and predicting procedure*

To train a model to predict the node labels of a prediction day's graph, we use a GCN with three convolution layers. The first hidden layer H^0 receives the original node features (feature matrix X). Our model runs 3 iterations of updates according to Eq. (10) to generate the final output node embeddings and the overall process of generating the output can be defined as:

$$Y = softmax(\hat{A}ReLU(\hat{A}ReLU(\hat{A}XW^0)W^1)W^2) \tag{11}$$

where $W^0 \in \mathbb{R}^{|X| \times C_0}$ is an input-to-hidden weight matrix for a hidden layer with C_0 feature maps and $W^1 \in \mathbb{R}^{C_0 \times C_1}$ is a hidden-to-hidden weight matrix with C_1 feature maps and $W^2 \in \mathbb{R}^{C_1 \times C_2}$ is a hidden-to-output weight matrix. The value of C_2 for prediction of the two classes (rise and fall) is 2. We train the network using cross-entropy loss over predictions for all stocks modeled in the network:

$$Loss = -\sum_{i \in Z_L} \sum_{f \in F} Z_{if} lnY_{if} \tag{12}$$

where Z_L is the set of labeled nodes and F is the set of price movement classes.

Considering the number of features and training examples, we used 4 channels for the first and second layers and 2 channels to predict

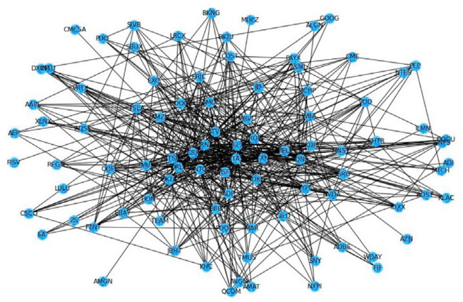
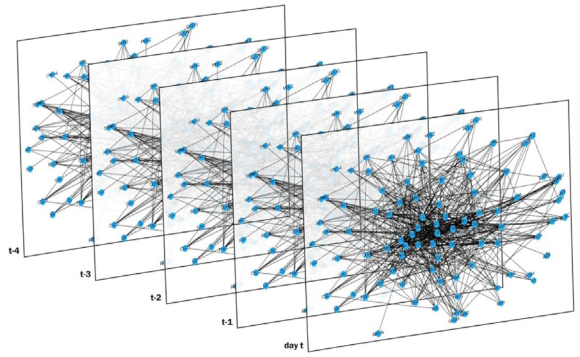



Fig. 2. A sample influence graph generated from the dataset used in this paper to represent the stock relations (left). GCNET combines the influence graphs of a prediction day with the graphs of its four preceding days to generate an instance of data used to train a model and predict the final labels of the prediction day's graph nodes.

Table 3
Features used to represent nodes in the graph.

Indicator	Description	Indicator	Description
RSI-S	RSI indicator signal	MFI-S	MFI indicator signal
BB-S	Bollinger bands indicator signal	CCI-S	CCI indicator signal
MACD-S	MACD indicator signal	V-S	Sign(Volume -Avg(last 5 days))
SAR-S	SAR indicator signal	CPOP-S	Sign(CP-OP)
ADX-S	ADX indicator signal	CPCPY-S	Sign(CP-Closing price yesterday)
S-S	Stochastic indicator signal		

2 classes in the third layer. The activation function for the first and second layers is ReLU and for the third layer is Sigmoid.

As we explained before, five graphs representing days $t-4$ to t are used for training the prediction model for the target day t. These graphs are used in chronological order to train the model for n epochs: GCNET first trains the GCN model for n epochs using the graph of the day $t-4$, then it continues the training process for another n epochs using the graph of the day $t-3$ and so on (the value of n is set using the validation data). By first using the fully-labeled graphs of the previous days whose node labels are reliable, GCNET tries to train an initial model that is consistent with the behavior of the market in the past few days. That initial model is then adjusted in the last n training epochs using the partially labeled graph of the target day t that represents the most recent state of the market as well as some initial predictions about its future. The result of this process is a model that can predict the final node labels for the graph of the day t (i.e. the next price fluctuations for all stocks in day t). GCNET repeats these model training and label prediction steps for each prediction day in the data. Fig. 3 summarizes the component architecture and the steps of GCNET.

3.4. Complexity analysis

GCNET algorithm is comprised of three phases: creating the network, running PLD, and training GCN. In the following, we use m and n to denote the number of stocks/nodes and the number of training data samples for each stock.

To create the influence network, first, the influence value between each pair of stocks is calculated by training the required QDA prediction models as we explained in Section 3.1. QDA is a variation of LDA algorithms with the time complexity of $O(knf + k^3)$ where f is the number of features, and $k = \min(f, n)$. When the number of training samples and the number of features is small, as they are in this work, this algorithm runs fast. In our implementations, we use a small set of features and $f < n$, so the time complexity is $O(nf^2+f^3)$. The number of models that must be trained to construct the entire graph is $O(m^2)$ and the total computation complexity for generating the complete influence network is $O(m^2(nf^2 + f^3))$. The computational cost of sparsifying each graph is $O(m^2)$.

The computational time complexity of the PLD phase depends on the set of basic algorithms we select to use. We have applied simple algorithms among which random forest has the highest time complexity that dominates the running time of this module. The Time complexity for building a complete decision tree is $O(f\ n \log n)$, where f is the number of features used by the model. So, the computational complexity of the random forest algorithm is $O(k\ f\ n \log n)$, where k is the number of trees generated by the model (Hassine et al., 2019). Since the PLD algorithm trains its models for each stock separately, its model training time complexity for the configuration used in this study would be $O(k\ f\ m\, n \log n)$. One can decide to train the basic algorithms for every prediction day, or instead, use the trained models for a few days before retraining or tuning them. In this study, we use a set of simple algorithms to keep the computational time low, and retrain them for each prediction day.

The time complexity of GCN is $O(L\,\|\hat{A}\|_0\, f + L\, N f^2)$, where $\|\hat{A}\|_0$ is the number of nonzero values of the adjacency matrix and f, L and n denote the size of feature vectors used by GCN, the number of GCN layers and the number of the nodes(i.e. stocks) respectively (Wu et al., 2020). The GCN training step is repeated for each prediction day while the trained model is used to predict the labels for all stock nodes in the graph.

Regarding the fact that the underlying graph used by GCNET is rather small and sparse, the running time for its periodical construction and daily analysis is low. So in practice, the training time of GCNET is almost linearly proportional to the running time of the most time-consuming algorithm used in PLD as well as the number of the PLD models. If, as we did in this study, few simple and efficient algorithms are used in PLD, the GCNET algorithm will be more time-efficient than most of the state-of-the-art deep models used by single stock prediction systems, like CNN, whose time complexity is $O(\Sigma_{l=1}^{d}\, n_{l-1}\, s_l^2\, n_l\, m_l^2)$, where d is the number of convolutional layers, n_l is the number of filters in the lth layer, n_{l-1} is the number of input channels of the lth layer, s_l is the size of the filter, and m_l is the size of the output (Tsironi et al., 2017), and also, Graph attention network (GAT), the time complexity of which is K times of GCN's, where K is the number of heads (Wang et al., 2021).

4. Experiments

In this section, we first explain the data and the pre-processing step, and then we present the evaluation process and the results of the GCNET model.

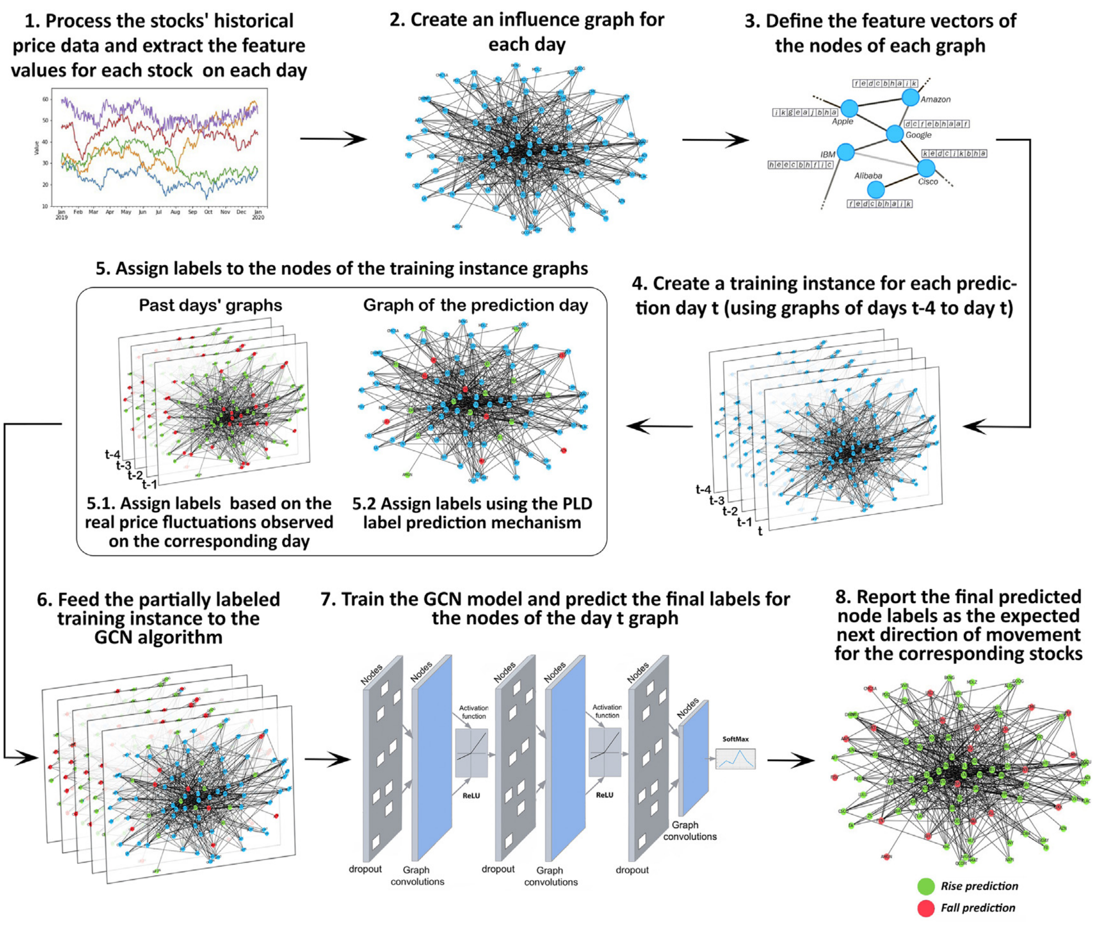

1. Process the stocks' historical price data and extract the feature values for each stock on each day
2. Create an influence graph for each day
3. Define the feature vectors of the nodes of each graph
Amazon
Apple
Google
IBM
Alibaba
Cisco
4. Create a training instance for each prediction day t (using graphs of days t-4 to day t)
5. Assign labels to the nodes of the training instance graphs
Past days' graphs
Graph of the prediction day
5.1. Assign labels based on the real price fluctuations observed on the corresponding day
5.2 Assign labels using the PLD label prediction mechanism
6. Feed the partially labeled training instance to the GCN algorithm
7. Train the GCN model and predict the final labels for the nodes of the day t graph
8. Report the final predicted node labels as the expected next direction of movement for the corresponding stocks
Nodes
Activation function
ReLU
SoftMax
dropout
Graph convolutions
Rise prediction
Fall prediction

Fig. 3. Steps of the GCNET Algorithm.

4.1. Data description

To evaluate the performance of GCNET and the baseline algorithms we used a dataset of 100 stocks with the largest market capitalization in Nasdaq, obtained from Yahoo Finance Appendix. We excluded 7 stocks whose data was not enough to support the training and evaluation process of all the baselines. For the rest of the stocks, the data set contains daily records from 01/01/2011 to 01/01/2021 with open, low, high, close, and adjusted close prices as well as the volume values. The historical data before each test day is used to train the model. The PLD models and the semi-supervised label prediction mechanisms are retrained for every prediction day. The data in the 09/15/2020 to 09/29/2020 interval is used for validation and to set the hyper-parameters. The test period is from 09/30/2020 to 12/30/2020 which includes 5952 records. Our dataset is available at the following address.[1]

4.2. Implementation and parameters of GCNET

To keep the graph structure up to date, the graph can be rebuilt in t-day intervals. If the training and validation dataset using which the edge weights are calculated are large enough, the resulting graph is expected to perform well for a long period of time, however, in our experiments, we reproduce the graph every 30 days. Fig. 2 shows a sample graph that is generated for the dataset used in this paper. We apply Adam optimizer (KingaD, 2015) to train the GCN and to avoid over-fitting we applied L2 regularization and dropout techniques. The weights of the GCN have been initialized by Glorot uniform initializer (Glorot and Bengio, 2010). For GCN implementations we used Keras (Chollet et al., 2018) and Spektral (Grattarola and Alippi, 2021) libraries. The number of top %n nodes that are labeled by the basic models is set using the validation data. Other parameters like learning rate and dropout rate have been set to widely-used values in the literature.

4.3. Evaluation metrics

Following previous work in stock prediction domain Wu et al. (2022), Kia et al. (2018), we adopt the standard measure of accuracy and Matthews Correlation Coefficient (MCC) as evaluation metrics. Given the confusion matrix $\begin{pmatrix} tp & fn \\ fp & tn \end{pmatrix}$ containing the number of samples classified as true positive, false positive, true negative, and false negative, Accuracy and MCC are defined as:

$$ACC = \frac{tp + tn}{tp + tn + fp + fn} \tag{13}$$

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \tag{14}$$

4.4. Baselines

We compare our model with the following baselines, which include network-based models, deep learning models, artificial neural network models, and models that use feature selection:

- **ALSTM:** This is a deep model, which uses a dual-stage attention-based recurrent neural network. In the first stage, the attention

[1] https://github.com/ut-kdd/GCNET-Dataset

Table 4
Performance of GCNET and the baselines on Nasdaq dataset.

Model	Accuracy	MCC
ALSTM (Qin et al., 2017)	0.5263 ±0.0004	−0.002 ±0.001
STOCKNET (Xu and Cohen, 2018)	0.5474 ±0.0027	0.0375 ±0.009
HATS (Kim et al., 2019)	0.5512 ±0.0021	0.0618 ±0.007
CNN-pred (Hoseinzade and Haratizadeh, 2019)	0.5342 ±0.0062	0.0366 ±0.015
Adv-LSTM (Feng et al., 2019)	0.5466 ±0.0045	0.0412 ±0.011
exLift+ DiMexRank+(PLD) (Kia et al., 2020)	0.5403 ±0.0068	0.0744 ±0.013
SACLSTM (Wu et al., 2021a)	0.5354 ±0.0041	0.0401 ±0.009
Price graphs (Wu et al., 2022)	0.5521 ±0.0032	0.0821 ±0.008
GCNET (our model)	**0.5671** ±0.0049	**0.1013** ±0.011

mechanism is applied to input features and is used to extract the relevant representation at each time step by referring to the previous hidden state. In the second stage, a temporal attention mechanism is applied to select relevant encoder hidden states across all time steps. Finally, an LSTM neural network is used to predict the stock's next movement. Qin et al. (2017).

- **STOCKNET:** This is a deep model that uses A variational auto-encoder to combine price and text information. Price features are modeled sequentially and the text is encrypted using a hierarchical attention mechanism in a multi-day sequence. It consists of three main components: Encoder, Decoder, and a module called Attentive temporal auxiliary, which integrates temporal loss through an attention mechanism to predict stock price movements (Xu and Cohen, 2018).
- **HATS:** A hierarchical graph attention method that uses graph modeling to weigh different relationships between stocks. They use a relational modeling module with initialized node representations. Their model selectively aggregates information on different relation types and adds the information to the representations of each company. They use only historical price data. Kim et al. (2019).
- **CNN-pred:** A deep CNN model that uses a diverse set of financial variables, including technical indicators, stock indices, futures contracts, etc. Their model constructs three and two-dimensional input tensors. Then, the input tensors are fed into a specified CNN model to make predictions. They use a fixed combination of the target market's time series as well as a few related time series as input to the model to predict the future of the target time series (Hoseinzade and Haratizadeh, 2019). To adopt CNN-pred as a baseline algorithm in this paper, we used the three closest neighbors of the target stock in the correlation graph as the related time series whose data is fed to the CNNpred model.
- **Adv-LSTM:** In this work, the authors introduced a deep model using an Adversarial Attentive LSTM mechanism, which leverages adversarial training to simulate stochasticity during model training. They advised employing adversarial training to improve the generalization of a neural network prediction model because the input features to stock prediction are typically based on stock price, which is essentially a stochastic variable and continuously changed with time by nature. Adversarial learning trains a classification model to defend adversarial examples, which are intentionally generated to perturb the model. They added perturbations to their stock data and train the model to work well under small yet intentional perturbations (Feng et al., 2019). We have tuned the hyperparameters of Adv-LSTM in our evaluations in order to compare the performance of GCNET with the best possible performance of Adv-LSTM.
- **exLift+DiMexRank+(PLD):** exLift + DiMexRank is a network-based model for prediction of market indices that encrypts relationships between markets' indices using Association rule learning and predicts the direction of market index movement using PageRank algorithm. The initial labeling step of this algorithm is designed for a specific market prediction task that cannot be applied in the domain of this study (Kia et al., 2020). Therefore, to use this algorithm as a baseline model, we applied our suggested PLD method for the initial labeling step in exLift+DimexRank, and kept the rest of the model intact.
- **SACLSTM:** This is a deep model, which combines convolution neural network and long short term memory neural network. It creates a sequence vector of 30-day historical data and options, that passes through four CNN layers and one LSTM layer. Finally a dense layer generates the predicted stock price movements (Wu et al., 2021a).
- **Price graphs:** This is a graph-based model that transforms price time series into visibility graphs. Then, structural information and associations among temporal points are extracted by a graph embedding method. Price graphs use the attention-based layers and a fully connected layer for stock trend prediction (Wu et al., 2022).

4.5. *Results and discussion*

In this section, we compare our model to the baseline models mentioned in the previous section. Our evaluations were performed on a large set of stocks in the Nasdaq market, and Table 4 summarizes the average performance of GCNET and baseline algorithms on 10 independent experiments over 93 stocks in the data set.

As can be seen, GCNET outperforms all baselines, while HATS, Price graphs, and exLift+ DimexRank+(PLD) achieve the highest prediction performances among baselines. The significant superiority of GCNET over baseline algorithms most of which use different sources of information and diverse sets of features as well as sophisticated prediction techniques demonstrates the power of the proposed graph-based prediction algorithm. As another important observation, graph-based models, especially GCNET, achieve significantly higher values of MCC compared to other algorithms. MCC, or Matthew's Correlation Coefficient, is designed to summarize the confusion matrix. A high value in the MCC measure reflects the power of a model in successfully predicting both ascent and descent classes. One reason for this superiority can be the fact that graph-based techniques exploit the graph structure to infer a consistent set of predicted final labels. In such a setting, if an initial prediction is mistakenly biased towards one of the classes for a certain stock, it can be corrected by the subsequent network-based information-aggregation steps.

The satisfactory performance of exLi-ft+DimexRank+(PLD), as a simple graph-based algorithm, affirms the usefulness of the graphical modeling in this domain, even if a simple label prediction mechanism is applied in that algorithm. Also, the superiority of our models over exLift+DiMexRank+(PLD) (that uses random walk analysis to assign labels to the nodes), shows the positive effect of formulating the price movement prediction problem as a model-based label prediction task. As we mentioned before, discovering and combining latent information from different stocks, whose relations are modeled as a graph structure, is the core ability of GCN that we exploited in our suggested algorithm, and as the experimental observations show, it can lead to significant improvement of the prediction performance.

Also, Price graphs, which is a powerful and recently introduced graph-based deep model, has achieved an accuracy of 0.5521, which is

Table 5
The number of times a model has outperformed others in prediction of a stock from Nasdaq Top-20 stocks.

Model	GCNET	Price graphs	HATS	STOCKNET	SACLSTM
Outperformed	15	2	1	1	1

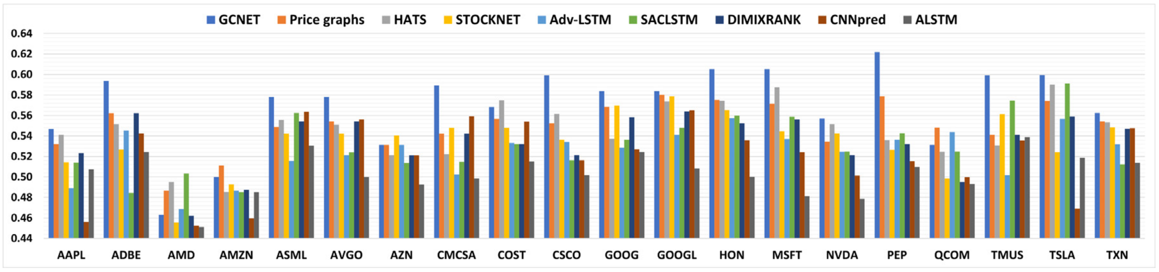


Fig. 4. Performance of GCNET and the baselines on Nasdaq Top-20 stocks.

Table 6
Effect of the underlying network structure on GCNET's performance.

Method	Accuracy
GCNET with Correlation Network	0.5579
GCNET with Influence Network	0.5671

Table 7
Performance of supervised and semi-supervised learning approaches.

Method	Accuracy
Supervised GCN	0.5508
Supervised GCN+LSTM	0.5546
GCNET (Semi-Supervised) (our model)	0.5671

significantly lower than the performance of GCNET. The Price graphs, unlike GCNET, uses the visibility graphs to model the relations between samples of a single stock over time, and the superiority of GCNET shows that our introduced approach for graph-based analysis of different stocks relations seems to be a more reliable and effective technique in the prediction of stock price fluctuations.

The previous experiment showed the superiority of the average GCNET's performance in the prediction of 93 stocks in Nasdaq. To observe the performance of the models on single stocks as well, in Fig. 4, we have presented the prediction accuracy of the models on top-20 stocks with the largest market capitalization in Nasdaq. It is clear that most of the times GCNET performs significantly better than other models. Table 5 illustrates the number of times a model has outperformed others for the prediction of each stock in this experiment. As it can be seen, GCNET has the best prediction in 15 out of 20 stocks, which makes its superiority in this experiment evident.

4.5.1. Model components analysis

In this subsection, we are going to discuss the importance of model components separately and provide more experiments for each component. First, we investigate the role of the influence network and then we discuss the node selection mechanism and GCN.

Network. The influence network discovers about 2300 edges during the weighting process, and after pruning the edges with low weight, the graph retains the effect of about 800 edges, which achieved an average of 3.67% increase in forecast accuracy in network validation data.

To see how GCNET performs when using another approach for modeling the stock relations, we replaced the influence network with a correlation network, which, as we explained in the literature review section, is the most commonly used network structure for predicting stocks. We created a network of stocks in which the edges represent the correlations among stocks past prices. We then used the same pruning technique applied for the influence network so that the edges connecting the strongly correlated stock nodes remain in the network and the pruned network is still connected. We kept the rest of the GCNET algorithm intact and evaluated its performance in the prediction of the stock price movements. The results are summarized in Table 6.

As the results show, the original version of GCNET outperforms the one using the well known and widely used correlation graph with a significant improvement. Another interesting observation is that even the correlation based version of GCNET is still superior to other baselines as reported in Table 4, that demonstrates the effectiveness of the introduced graph-based framework compared to other state of the art approaches for the prediction of stock price movements.

Semi-supervised training. As the main part of our model, GCN predicts the next day by combining neighbors' information and considering labeled nodes for stock movement. To demonstrate the importance of using the GCN algorithm in a semi-supervised approach, we compared our model with two different supervised versions in both of which the model is trained only with the graphs from previous days whose nodes have known labels (This means that the last day's graph whose labels are unknown and in the original setting, was partially labeled using the predictions of the PLD mechanism is no longer available to the GCN model). In the first supervised model, GCN model trained on the fully-labeled networks of the past days is directly used to predict the target day's price movements while in the second model, we fed the embeddings created by the GCN to an LSTM model for the prediction of the target day's price movements.

The results presented in Table 7 show that the use of GCN in a supervised framework reduces the performance of the model, and confirms that our suggested semi-supervised approach for extracting embeddings leads to superior label-prediction performance. The reason is that in the suggested semi-supervised approach for embedding, the most recent information about the state of each stock is considered during the GCN weight training and learning process over the target-day's partially labeled graph (Kipf and Welling, 2016).

Label assignment. As we explained before, the algorithm uses a pool of basic agile predictors and by network analysis techniques, selects the most effective nodes for labeling as well as the best available predictions to be used as their label. To see the effect of using the pool of predictors mechanism for assigning accurate initial labels, we evaluated the performance of the model when using each single member of the pool, instead of the original ensemble technique, for label assignment. The results are summarized in Table 8 and it can be seen that using the original ensemble labeling technique has led to a significant improvement in the GCNET's overall performance. It can also be seen that among the basic algorithms used by the PLD, Random Forest has the highest prediction performance, ranked second after PLD

Table 8
Performance of different algorithms used for assigning initial labels.

Algorithm	Overall accuracy of **GCNET**
Fisher Discriminant Analysis	0.5362
Decision Tree	0.5389
Naive Bayes classifier	0.5396
Multilayer Perceptron	0.5403
Linear Discriminant Analysis	0.5494
Random Forest	0.5511
Plausible Label Discovery (PLD)	0.5671

Table 9
Performance of GCNET in improving the accuracy of initial predictions.

Method	Accuracy
PLD on initially labeled nodes	0.5247
GCNET on initially labeled nodes	0.5598
GCNET on all nodes	0.5671

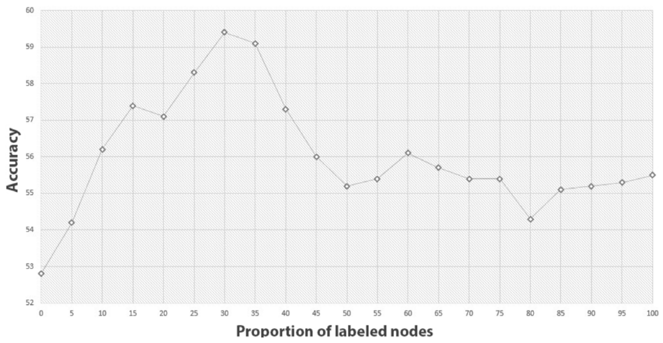


Fig. 5. Performance of GCNET when different proportions of nodes are initially labeled.

itself, that is an interesting observation, as PLD and Random Forest are both ensemble methods.

As we know, the GCNET assigns initial labels to a subset of the nodes in the graph of the target day (as an initial guess about the future price movements of some stocks) and leaves the rest of the nodes unlabeled. After processing this partially labeled graph, GCNET generates its final predictions for every stock, including both initially labeled and unlabeled ones. Table 9 shows how the GCNET prediction mechanism improves the initial labels. As can be seen, on average, only 0.5247 of the initial labels have been correct, while GCNET improves the accuracy of those labels in its final prediction by about 3.5%. It also achieves even a higher overall prediction accuracy, considering both initially labeled and unlabeled nodes. This is an important observation that shows the performance of GCNET exceeds the prediction performance of PLD, even for the nodes that have initially been labeled by PLD. The reason is that GCNET combines initial predictions with the information reflecting the graph structure and the local node states to refine the initial guesses and make consistent and reliable final predictions. This observation confirms the usefulness of graph construction and analysis techniques used by GCNET.

Also, GCNET uses a procedure to select some nodes for labeling by considering the density of each part of the graph. To see the effect of the node selection procedure, we evaluated the performance of GCNET version in which a random subset of nodes are selected and then the label produced by their best predictors are assigned to them as initial labels. The results are represented in Table 10. It can be seen that our original node selection procedure has a critical role in the overall GCNET performance improvement.

To report the effect of n, the portion of nodes that are initially labeled by PLD, on the performance of the algorithm, we presented the prediction's performance for different values of this parameter in Fig. 5. One advantage of GCN is its outstanding performance when training the

Table 10
The effect of the node selection procedure on the GCNET's performance.

Method	Accuracy
Random node selection	0.5498
GCNET's node selection preocedure	0.5671

model with a rather small set of labeled nodes (Kipf and Welling, 2016). However, as shown in this figure, using a very small value for n, reduces the accuracy of the prediction possibly because in such a situation, the information provided by the labeled nodes would not be enough for extracting reliable patterns for prediction. On the other hand, labeling too many nodes by the basic predictors reduces the performance of the algorithm. These observations confirm the positive role of network analysis in boosting the performance of the prediction: Injecting too much information from the history based models into the graph, not only can introduce some noise to the model but also limits the effect of GCN in the label prediction process, that apparently reduces the performance of the algorithm.

5. Conclusion

In this paper, we presented a novel GCN-based framework for the prediction of stock price movements. Our suggested framework models the relations among stock prices as a novel graph structure called influence network. The algorithm uses a pool of supervised history-based prediction models to label a subset of nodes using the plausible predictions of those models. The partially labeled network is then analyzed by a GCN to extract new representations for the nodes and predict the next direction of price movement for all the stocks. Our experiments showed that the suggested graph-based semi-supervised prediction model significantly outperforms state-of-the-art baseline single-stock and graph-based prediction algorithms. They also show that the introduced algorithm for constructing the influence network as well as the initial labeling assignment technique and the final label prediction mechanism, each has a contribution to the high performance achieved by GCNET framework.

Although GCNET was designed to be applicable when the only available source of information is the price history of stock, one direction for future research is to investigate how other sources of information, such as textual reports available on the web or derived sentiments from social networks data can be used to enhance the quality of the underlying network model generated by GCNET. Another research direction is to study the process of assigning initial labels to the nodes by considering the graphical features of the nodes and the amount of initial information that is injected into different parts of the graph.

CRediT authorship contribution statement

Alireza Jafari: Conceptualization, Design, Implementation, Formal analysis, Visualization, Writing. **Saman Haratizadeh:** Conceptualization, Design, Validation, Resources, Supervision, Writing, Review & Editing.

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.

Appendix. Dataset

Table A.11 shows the 100 greatest stocks in Nasdaq index in terms of market capitalization, meanwhile, our dataset has been collected from the Yahoo Finance website at the beginning of 2021.

Table A.11
The list of the stocks in the dataset used in this paper.

Stock	Name	Stock	Name
AAPL	Apple Inc.	JBHT	J B Hunt Transport Services Inc.
ABNB	Airbnb, Inc.	JD	JD.com, Inc.
ADBE	Adobe Inc.	KDP	Keurig Dr Pepper Inc.
ADI	Analog Devices, Inc.	KHC	The Kraft Heinz Company
ADP	Automatic Data Processing, Inc.	KLAC	KLA Corporation
ADSK	Autodesk, Inc.	LCID	Lucid Group, Inc.
AEP	American Electric Power Company, Inc.	LRCX	Lam Research Corporation
AFRM	Affirm Holdings Inc.	LULU	lululemon athletica inc.
ALGN	Align Technology, Inc.	MAR	Marriott International
AMAT	Applied Materials, Inc.	MCHP	Microchip Technology Incorporated
AMD	Advanced Micro Devices, Inc.	MDLZ	Mondelez International, Inc.
AMGN	Amgen Inc.	MELI	MercadoLibre, Inc.
AMZN	Amazon.com, Inc.	MNST	Monster Beverage Corporation
ASML	ASML Holding N.V.	MRNA	Moderna, Inc.
ATVI	Activision Blizzard, Inc	MRVL	Marvell Technology, Inc.
AVGO	Broadcom Inc.	MSFT	Microsoft Corporation
AZN	Astrazeneca PLC	MTCH	Match Group, Inc.
BIDU	Baidu, Inc.	MU	Micron Technology, Inc.
BIIB	Biogen Inc.	NFLX	Netflix, Inc.
BKNG	Booking Holdings Inc.	NTES	NetEase, Inc.
BNTX	BioNTech SE - ADR	NVDA	NVIDIA Corporation
CDNS	Cadence Design Systems, Inc.	NXPI	NXP Semiconductors N.V.
CHTR	Charter Communications, Inc.	ODFL	Old Dominion Freight Line, Inc.
CMCSA	Comcast Corporation	OKTA	Okta, Inc.
CME	CME Group	ORLY	O'Reilly Automotive, Inc.
COST	Costco Wholesale Corporation	PAYX	Paychex, Inc.
CRWD	CrowdStrike Holdings, Inc.	PDD	Pinduoduo Inc.
CSCO	Cisco Systems, Inc.	PEP	Pepsico, Inc.
CSX	CSX Corporation	POOL	Pool Corporation
CTAS	Cintas Corporation	PYPL	PayPal Holdings, Inc.
CTSH	Cognizant Technology Solutions Corporation	QCOM	QUALCOMM Incorporated
DDOG	Datadog, Inc.	REGN	Regeneron Pharmaceuticals, Inc.
DOCU	DocuSign, Inc.	ROKA	Roku, Inc.
DXCM	DexCom, Inc.	ROST	Ross Stores, Inc.
EA	Electronic Arts Inc.	SBUX	Starbucks Corporation
EBAY	eBay Inc.	SIVB	SVB Financial Group
ERIC	Education Resources Information Center	SNPS	Synopsys, Inc.
EXC	Exelon Corporation	SNY	Sanofi SA
FB	Meta Technology company	TEAM	Atlassian Corporation Plc
FISV	Fiserv, Inc.	TMUS	T-Mobile US, Inc.
FTNT	Fortinet, Inc.	TROW	T Rowe Price Group Inc.
GILD	Gilead Sciences, Inc.	TSLA	Tesla, Inc.
GOOG	Alphabet Inc.	TXN	Texas Instruments Incorporated
GOOGL	Alphabet Inc.	VRTX	Vertex Pharmaceuticals Incorporated
HON	Honeywell International Inc.	WBA	Walgreens Boots Alliance, Inc.
IDXX	IDEXX Laboratories, Inc.	WDAY	Workday, Inc.
ILMN	Illumina, Inc.	XEL	Xcel Energy Inc.
INTC	Intel Corporation	XLNX	Xilinx, Inc.
INTU	Intuit Inc.	ZM	Zoom Video Communications, Inc.
ISRG	Intuitive Surgical, Inc.	ZS	Zscaler, Inc.

References

Arévalo, A., Niño, J., Hernández, G., Sandoval, J., 2016. High-frequency trading strategy based on deep neural networks. In: International Conference on Intelligent Computing. Springer, pp. 424–436.

Bustos, O., Pomares-Quimbaya, A., 2020. Stock market movement forecast: A systematic review. Expert Syst. Appl. 156, 113464.

Chollet, F., et al., 2018. Keras: The python deep learning library. Astrophys. Source Code Library ascl–1806.

Di Persio, L., Honchar, O., 2016. Artificial neural networks architectures for stock price prediction: Comparisons and applications. Int. J. Circuits Syst. Signal Process. 10 (2016), 403–413.

Feng, F., Chen, H., He, X., Ding, J., Sun, M., Chua, T.-S., 2019. Enhancing stock movement prediction with adversarial training. arXiv preprint arXiv:1810.09936.

Gite, S., Khatavkar, H., Kotecha, K., Srivastava, S., Maheshwari, P., Pandey, N., 2021. Explainable stock prices prediction from financial news articles using sentiment analysis. PeerJ Comput. Sci. 7, e340.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, pp. 249–256.

Grattarola, D., Alippi, C., 2021. Graph neural networks in TensorFlow and keras with spektral [application notes]. IEEE Comput. Intell. Mag. 16 (1), 99–106.

Grover, A., Leskovec, J., 2016. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 855–864.

Gunduz, H., Yaslan, Y., Cataltepe, Z., 2017. Intraday prediction of Borsa Istanbul using convolutional neural networks and feature correlations. Knowl.-Based Syst. 137, 138–148.

Hammond, D.K., Vandergheynst, P., Gribonval, R., 2011. Wavelets on graphs via spectral graph theory. Appl. Comput. Harmon. Anal. 30 (2), 129–150.

Hassine, K., Erbad, A., Hamila, R., 2019. Important complexity reduction of random forest in multi-classification problem. In: 2019 15th International Wireless Communications & Mobile Computing Conference. IWCMC, IEEE, pp. 226–231.

Hekmatfar, T., Haratizadeh, S., Goliaei, S., 2021. Embedding ranking-oriented recommender system graphs. Expert Syst. Appl. 181, 115108.

Hoseinzade, E., Haratizadeh, S., 2019. CNNpred: CNN-based stock market prediction using a diverse set of variables. Expert Syst. Appl. 129, 273–285.

Hu, Z., Liu, W., Bian, J., Liu, X., Liu, T.-Y., 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. pp. 261–269.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An Introduction to Statistical Learning, Vol. 112. Springer.

Kara, Y., Boyacioglu, M.A., Baykan, Ö.K., 2011. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul stock exchange. Expert Syst. Appl. 38 (5), 5311–5319.

Kia, A.N., Haratizadeh, S., Shouraki, S.B., 2018. A hybrid supervised semi-supervised graph-based model to predict one-day ahead movement of global stock markets and commodity prices. Expert Syst. Appl. 105, 159–173.

Kia, A.N., Haratizadeh, S., Shouraki, S.B., 2020. Network-based direction of movement prediction in financial markets. Eng. Appl. Artif. Intell. 88, 103340.

Kim, M., Sayama, H., 2017. Predicting stock market movements using network science: An information theoretic approach. Appl. Netw. Sci. 2 (1), 1–14.

Kim, R., So, C.H., Jeong, M., Lee, S., Kim, J., Kang, J., 2019. Hats: A hierarchical graph attention network for stock movement prediction. arXiv preprint arXiv:1908.07999.

KingaD, A., 2015. A method for stochastic optimization. In: Anon. International Conference on Learning Representations. ICLR, San Dego.

Kipf, T.N., Welling, M., 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Li, W., Liao, J., 2017. A comparative study on trend forecasting approach for stock price time series. In: 2017 11th IEEE International Conference on Anti-Counterfeiting, Security, and Identification. ASID, IEEE, pp. 74–78.

Long, J., Chen, Z., He, W., Wu, T., Ren, J., 2020. An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market. Appl. Soft Comput. 91, 106205.

Mehta, P., Pandya, S., Kotecha, K., 2021. Harvesting social media sentiment analysis to enhance stock market prediction using deep learning. PeerJ Comput. Sci. 7, e476.

Nelson, D.M., Pereira, A.C., de Oliveira, R.A., 2017. Stock market's price movement prediction with LSTM neural networks. In: 2017 International Joint Conference on Neural Networks. IJCNN, IEEE, pp. 1419–1426.

Park, K., Shin, H., 2013. Stock price prediction based on a complex interrelation network of economic factors. Eng. Appl. Artif. Intell. 26 (5–6), 1550–1561.

Patel, J., Shah, S., Thakkar, P., Kotecha, K., 2015. Predicting stock market index using fusion of machine learning techniques. Expert Syst. Appl. 42 (4), 2162–2172.

Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., Cottrell, G.W., 2017. A dual-stage attention-based recurrent neural network for time series prediction. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 2627–2633.

Rather, A.M., Sastry, V., Agarwal, A., 2017. Stock market prediction and portfolio selection models: a survey. Opsearch 54 (3), 558–579.

Saramäki, J., Kivelä, M., Onnela, J.-P., Kaski, K., Kertesz, J., 2007. Generalizations of the clustering coefficient to weighted complex networks. Phys. Rev. E 75 (2), 027105.

Shah, J., Jain, R., Jolly, V., Godbole, A., 2021. Stock market prediction using bi-directional LSTM. In: 2021 International Conference on Communication Information and Computing Technology. ICCICT, IEEE, pp. 1–5.

Shin, H., Hou, T., Park, K., Park, C.-K., Choi, S., 2013. Prediction of movement direction in crude oil prices based on semi-supervised learning. Decis. Support Syst. 55 (1), 348–358.

Soni, S., 2011. Applications of ANNs in stock market prediction: a survey. Int. J. Comput. Sci. Eng. Technol. 2 (3), 71–83.

Tsironi, E., Barros, P., Weber, C., Wermter, S., 2017. An analysis of convolutional long short-term memory recurrent neural networks for gesture recognition. Neurocomputing 268, 76–86.

Vrandečić, D., Krötzsch, M., 2014. Wikidata: a free collaborative knowledgebase. Commun. ACM 57 (10), 78–85.

Wang, Z., Wang, Y., Yuan, C., Gu, R., Huang, Y., 2021. Empirical analysis of performance bottlenecks in graph neural network training and inference with GPUs. Neurocomputing 446, 165–191.

Wu, J.M.-T., Li, Z., Herencsar, N., Vo, B., Lin, J.C.-W., 2021a. A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. Multimedia Syst. 1–20.

Wu, J.M.-T., Li, Z., Srivastava, G., Tasi, M.-H., Lin, J.C.-W., 2021b. A graph-based convolutional neural network stock price prediction with leading indicators. Softw. - Pract. Exp. 51 (3), 628–644.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y., 2020. A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. 32 (1), 4–24.

Wu, J., Xu, K., Chen, X., Li, S., Zhao, J., 2022. Price graphs: Utilizing the structural information of financial time series for stock prediction. Inform. Sci. 588, 405–424.

Xu, Y., Cohen, S.B., 2018. Stock movement prediction from tweets and historical prices. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1970–1979.

Yin, T., Liu, C., Ding, F., Feng, Z., Yuan, B., Zhang, N., 2022. Graph-based stock correlation and prediction for high-frequency trading systems. Pattern Recognit. 122, 108209.

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., Li, H., 2019. T-gcn: A temporal graph convolutional network for traffic prediction. IEEE Trans. Intell. Transp. Syst. 21 (9), 3848–3858.

Zhong, X., Enke, D., 2017. Forecasting daily stock market return using dimensionality reduction. Expert Syst. Appl. 67, 126–139.

Zhou, Y., Graham, S., Alemi Koohbanani, N., Shaban, M., Heng, P.-A., Rajpoot, N., 2019. Cgc-net: Cell graph convolutional network for grading of colorectal cancer histology images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops.