Stock Forecasting using Neural Network with Graphs

Shuyi Peng

Master of Science

University of York

Computer Science

May 2021

Abstract

Due to the complex characteristic in the stock market, it is always a challenge and interesting topic to predict stock price. With the development of neural network models, deep learning has become a popular way to solve the stock prediction problem. Many of the current studies focus on how the stock own historical information which will affect the stock price in the future. Although the individual historical features are essential, the stock price is also affected by the other stocks.

To capture such internal relations and influence, we propose to join stock graphs with the neural network model. The reason we choose to use graphs is that the connected graph structure can compress such relation between stocks. We investigate different graph construction methods so that we can describe the stock relation in a comprehensive way. Although graph convolutional network(GCN) has already been proved effective in the prediction of stock movement, it only considers one single graph. Here, we build a combination model based on the GCN that the model can deal with multiple graph features. Apart from GCN, we also applied the transformer-based model to learn the correlation between the stocks. Transformer is a popular model for natural language processing and the implementation in stock prediction is focus on dealing with the public mood. In our research, we applied the stock graph as a mask to attention layer so that the transformer can have prior knowledge.

Our experiment applies the stock data from the New York stock exchange. We propose our model using graphs outperforms the recurrent neural network or other methods which do not take the graph structure into account. In the experiment, we investigate how various type of graphs influence the prediction result. The results show that the combination of multiple graphs effectively improves accuracy. But it does not outperform the general GCN model due to the quality of our constructed graphs. Furthermore, we introduced three graph construction methods and examined their impacts on stock prediction problem. The result indicates that the correlation graph is the optimal choice among them. Both multi-graph GCN and transformer with graph mask outperform the LSTM model. Besides, pure transformer+LSTM also produces a better result than the LSTM model. The result reveals our assumption that the internal relation provides sufficient improvements for the stock prediction problem.

Acknowledgements

I would like to express gratitude to my supervisors Dr James Cussens and Dr Suresh Manandhar. This thesis would not have been possible without the valuable suggestion and supports that I received from them. I also want to thanks the college that provides me with a comfortable learning atmosphere and resources. Besides, I also want to thanks my friend Marcelo Sardelich and Zongyu Yin who gave me lots of support and inspiration during the research.

Declarations

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

It is always a challenging problem to forecast the stock prices. The reason for it is so difficult is because of inherently unstable factors and a complicated market outlook. The stock market reflects many influences, rendering particular circumstances challenging to evaluate. For example, while a newly published policy will impact specific sectors, precisely measuring the extent to which this policy will influence the sectors remains elusive. Any policy-driven influence on a primary industry will also reflect in the stock prices of secondary and tertiary sectors. Previous literature discussed that the stock price movement is a random process and stock performance is unpredictable[1, 2]. According to the Efficient Market Hypothesis(EMH) proposed in 1970, investors cannot get excess profits above the market average, and it is impossible to predict the direction of the market in the coming days or weeks[3]. The market is efficient means that the behaviour of investors are rational, and investors can respond reasonably to all market information quickly. Though the EMH is widely accepted, by the start of the twenty-first century, more people believe that the stock prices are at least partially predictable based on their past performance[4, 5].

White is the first person who implemented the neural network model to the stock prediction problem[6]. He used feedforward to decode nonlinear regularities in price movement. The results showed that a simple feedforward network is unable to refute the EMH hypothesis. With the improvement, the follwoing papers proved that the neural networks and other machining learning methods outperform statistical and traditional regression methods[7, 8].

Apart from the EMH hypothesis, behavioural finance is also widely discussed, and many of the current deep learning methods are based on this concept. The behavioural finance proposes that stock prices are not only determined by the value of the enterprise but also influenced by investors behaviours. The investors' behaviours are correlated with the public mood. Several papers collected information from social media, such as Twitter, and feed this mood information as the training features[9, 10, 11][12]. The model uses natural language process methods such as transformer to encode the public mood information and combine theses with stock features, such as close prices. The joint features feed the combined features to the neural network for prediction. This neural network varies, such as self-organizing fuzzy neural network(SOFNN), capsule network, convolution neural network(CNN), etc.

Although public mood is widely used in stock prediction problem, many studies still focus on the past performance of stocks. Since the features of stocks are time-sequential, recurrent neural network(RNN) is a widely used NN method for stock prediction[13][14]. One of the most popular RNN models is LSTM, and research shows that the performance of LSTM is better than the multiple layer perceptron[15]. Nevetheless, RNN is not the only choice to the stock prediction problem using historical information and it has many drawbacks[16]. Many other methods are invented to replace RNN in stock prediction problem such as genetic fuzzy neural networks(GFNN)[17], wavelet neural network and etc[18].

However, the stock market is very complicated. All current methods only focus on the information about the individual stock, which neglects the correlative information between different stocks. Actually, the stocks in the market will interact with each other. Such an interaction is hard to be caught by looking at their own history. So our research is aimed to predict the stock prices including the internal correlative information.

1.2 Research Objective

The information in the stock prices not only relates to themselves but also are influenced, either positively or negatively, by the performance of other stocks in the market. Our research aimed to identify this kind of relation between stocks and use this relationship to develop models in stock price prediction.

The historical performance of stocks usually refers to past stock prices and trading volume. In our research, we want to develop the graph structure to represent the correlation between stocks, and apply the price-performance as the training feature. Previous study shows that graph convolutional neural networks(GCN) has been used for stock prediction, and the corresponding result can improve the prediction accuracy[19]. In our research, we want to use the transformer to resolve graphs information and combined graphs and historical stock features

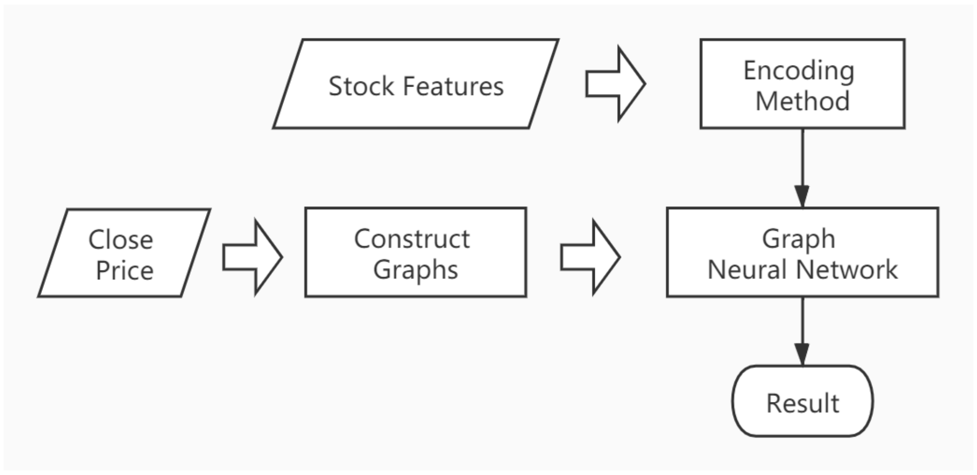


Figure 1.1: Architecture overview

to do some predictions. Presently, the transformer model is popular in solving natural language processing problem. It is more used for analysing public mood instead of deal with historical features in the stock prediction problem[10].

The focus of this project is to construct the graphs in the stock forecasting problem. It is a vital factor to find a way to represent such relations between stocks in this investigation. The first step is to build graphs where the edges represent the relationships and the nodes indicate the stocks. Because stocks can exhibit various relations with other stocks, it is essential to discuss the different ways of generating the graphs and evaluate how different graphs influence prediction. As mentioned, relationships will influence performance, but the historical performance pertaining to stock itself is equally important or possibly more critical to prediction. Consequently, it was our goal to create a useful tool based on this relationship along with information that pertains to stock prices to assist in the prediction of stock performance.

As shown in Figure 1.1, we need to build a graph neural network model that can combine stock graphs and their historical features to accomplish our aim. Besides, we design to encode historical features to improve model accuracy. GNN networks facilitated combining the model with the generated graphs in the stock. In addition, we introduce transformers to train the stock graphs. We expect that the various generated graphs could increase the accuracy of prediction.

1.3 Contributions

In our paper, we investigate how graphs help the prediction of the stock price. We used different methods to build stock graphs and to evaluate the influence of graph in the prediction of stock prices. Our contribution is as follows:

- We used multiple graphs convolution neural network to predict the stock price. Current GCN used a single graph for stock prediction, while we proposed a model to deal with multiple graphs to improve accuracy. Drawing on the fact that graphs produced by different methods contain varying information, we want the training process to include as much useful information as possible, so that we can analyse the stock market in a comprehensive way.
- We investigate how various stock graphs can influence the prediction. Drawing on the fact that various graphs provide different information, we investigate how the stock graph structure and the relation between stocks influence and prediction accuracy.
- Instead of using GCN, we used transformer architecture to deal with the stock graphs. Transformer is widely used for process public mood or event information, but not used for exploring the relation between stocks. In our research, we discussed how the attention module in the transformer works on the explore the stocks' internal relation and how to apply graph as masks to the transformer model.

1.4 Problem Analysis

The aim of this project is to use graphs to assist in prediction of stock performance. We conclude three main problems that we will face how we will solve in this research:

1. The datasets we are using do not include stock graphs, so we need to use the accessible information to generate stock graphs. The correlation-based stock graph is the most widely used stock presentation. However, we want to involve multiple stock graphs in the training process, so we need various graphs construction method besides correlation. For all graphs we construct, we want them to be reasonable for the prediction problem. In the literature review part, we have introduced three graph construction methods.

For each method, we will explain what the relation the graph represents is.

2. Not all stock features are useful in prediction. Especially in the case of the neural network, too much information may mislead the final prediction. In one example of using the past stock price as the input feature, year-old stock price information is not helpful for predicting the current stock price. Besides, the stock price is influenced by various factors; the feature we can obtain from the datasets are limited. In terms of stock features, we will limit the time range that each stock feature contains. For example, for each stock feature, we will only input the stock price for the most recent three months. Therefore, the prediction will be affected by recent stock performance. In the methodology part, we will test different input feature combination, and select the optimal setting for the later experiment.

3. Defining the output is also a problem. The most straightforward approach is to compare the next day's price with the current price. However, stocks have different reaction times in terms of the market. Due to the characteristic of the stock market, the daily stock performance is almost random. Such a prediction may suffer from low accuracy. Because this project aims to predict stocks' performance, the next day's price is not the only indicator. As long as the defined target reasonably shows stock performance, we can try different settings and find the most predictable setting through experimentation.

1.5 Report Structure

The report is divided into six parts: literature review, problem analysis, theory, design and implementation, evaluation and a conclusion.

The literature review provides a brief introduction to the method and theory that this project employed. Three aspects of existing concepts include an introduction to the stock market, methods to build stock graphs and a description of the neural network used in the project. We introduced three graph construction method in this section. Besides, we mentioned how the graph neural process the graph information and we also introduced the transformer model structure, and how this can process graph information.

The methodology section contains an introduction to datasets and description to aid in facilitating reproducing the existing methods; it also analyses how features selection and preprocessing impacted the prediction. Since there exists very few GCN methods on the stock predicton problem, we want to set these methoda as benchmark so that we could compare our methods with the current way of combining stock graphs in the stock prediction.

The implementation part discusses how we combined the existing method with new ideas and focuses on a way to align multiple stock graphs with the neural network algorithm. Moreover, we introduce the transformer method and use graphs as masks to accomplish the prediction in this section.

The result and evaluation part provides an analysis of how different model and settings affected the results of the experiment. This section presents a comparison of the differences between methods and discussed possible reasons for the given outcomes.

In the conclusion, an overall summary describes the performance of the various methods on the real data and evaluates whether the new idea offers improved prediction capability. In addition, some of the experimental findings may come into consideration in later work. We also summarized potential investigations that may positively impact the prediction model in the future.

2 Literature Review

2.1 Introduction to the stock market

The performance of a stock reflects a reaction on the part of investors. In other words, prices largely depend on investors' expectations for one or more stocks. Such expectations are influenced not only by the actual changes happening in the sectors but also by information that investors glean from the news, social media, etc. However, these information sources could be unreliable and cause difficulties in stock forecasting[20]. The existing research has already revealed main factors that cause stock price changes. Here, we briefly introduce how the market works and the main features to focus on in making a prediction.

People commonly consider that news reports and large trading volumes have a significant impact on stock prices. Theoretically, the price moves when new information becomes available to market participants who in turn respond to this informations[21]. According to this theory, the price should jump upon release of a piece of new news, meaning that news should be the primary determinant of price volatility. However, evidence shows that the volatility process is random.[22]. Only a small amount of stocks in the market will react to political and world events. The other evidence shows that the large transaction volumes are not responsible for the large jump in the stock price. In fact, the volume available in the market is very small compared to stock capitalisation. Only a small number of stocks in the market tend to react to political and world events. Other evidence reveals that large transaction volumes are not responsible for a large jump in stock price. In fact, the volume available in the market is small compared to stock capitalisation. The market is 'liquid'; the inference is that the price is established when liquidity dries out.

However, macroeconomic news such as interest rates, taxes and new policies can still influence the market, and these large events will lead to price jumps. The authors of 'Trading volume and serial correlation in stock returns'[23] note the influence of daily trading volumes on stock performance, showing that a stock price is more likely to decline on high-volume days when buyers' expectations for the stock price increase. In the short term, then, trading volume can be valuable in stock forecasting.

This project focuses on short-term prediction. Therefore, for the price of each stock along with trading volume as the input feature, news will not be considered in the model.

2.2 Graph Construction

Graphs can be generated in different ways. To define a stock market graph, we should define what the vertices and edges represent. In our case, the vertices (or nodes) are the selected stocks, and the edges are the intended area. Since we aim to use the graphs to assist in prediction, the definition of the edges may have different effects on the prediction. The binary sector- or industry-based graph is the most straightforward graph, in which the edges between the stocks represent whether the stocks belong to the same sectors or industry. Moreover, we attempted to include more complex graphs containing more meaningful information.

2.2.1 Sector graph

The concept of a sector graph is to collect the stock in the same sector. The data obtained from Yahoo Finance include the types of sector and industry related to the stock. The sector is the parent class of the industry type, i.e. two stocks can both be classified as healthcare sector, however, one belongs to the medical devices industry and the other one is related to the drug manufacturing industry. The sector graph $G_S = (V, E_S)$ is defined as follow:

$$S = \{s_1, s_2, \ldots, s_n\} \tag{2.1}$$

$$e_{ij} = \begin{cases} 1 & if\, v_i \in s_n, v_j \in s_n \\ 0 & if\, v_i \in s_n, v_j \notin s_n \end{cases} \tag{2.2}$$

Where S denote the set of sectors and the vertices(nodes) v_i and v_j are stocks. The $e_{ij}$is the value of edge between node i and j, 1 if the two vertices belong to the same sector, and 0 if they belongs to a different sector. If stocks we chose are all from the same type of sector, the edge will depends on the type of industry.

The sector graph is strong since it contains the information not been considered by the historical feature.

2.2.2 Correlation graph

The concept of correlations between the closing prices of stocks, commonly utilized to construct networks (graphs) for the stock market, is introduced in the paper[24]. As mentioned, the nodes are stocks, and the edges connecting each node are calculated by cross-correlations of the variations in stock prices.

The cross-correlation is based on the return price of the stocks. Assume $p_i(t)$ is the closing price of a stock i on day t. The function of the return price is defined as:

$$r_i(t) := \ln\left[\frac{p_i(t)}{p_i(t-1)}\right] \tag{2.3}$$

Let $x_i(t)$ and $x_j(t)$ be the return price of stock i and stock j respectively on day t, where $1 \leq t \leq T$ and T is the number of days we used for evaluation(i.e the size of the sequence). The method comparing two time series without any relative time shift. The correlation based stock graph is denoted by $G_C = (V, E_C)$ and the value of edge between nodes i and j is equal to cross-correlation value. The correlation c_{ij} between sequence x_i and x_j is defined as:

$$c_{ij} := \frac{\sum_t \left[(x_i(t) - \bar{x}_i)(x_j(t) - \bar{x}_j)\right]}{\sqrt{\sum_t (x_i(t) - \bar{x}_i)^2}\sqrt{\sum_t (x_j(t) - \bar{x}_j)^2}} \tag{2.4}$$

Where$\bar{x}_i$and $\bar{x}_j$ are the means of the return price sequence x_i and $x_j$respectively, over the period $t = 0$ to $t = T$.

According to the definition of cross-correlation, the result of correlation should be between 1 and -1, where 0 indicates two variables do not correlate, negative correlation means one variable increases as the other one decreases and positive correlation means two variables increase simultaneously. The correlation for the stock network according to the defination is scaled to 0 to 1 as this measures only the positive impact between two stocks.

A positive fractional number $\rho < 1$ is chosen as the threshold. The stocks i and j are only connected if $c_{ij} > \rho$. Therefore, the constructed graph will be an unweighted graph where the edge has no value assigned. The lower the threshold, the more connection exists between stocks. The experiment result shows that the network will be randomly connected if the ρ value is small. Therefore, we would like to choose a relatively high threshold value, i.e. $\rho = 0.9$ according to the sample in the paper, to make the connection reasonable.

The correlation-based network formed scale-free graphs(unweighted graph). The degree distribution of this graph can reflect the fluctuation in the market[25]. Thus, this graph contains information useful for our stock price prediction. The disadvantage of correlation graphs is that a scale between 0 and 1 loses information when the two stocks are correlated negatively. This information is also useful for predicting the stock performance since a reduction in the price of one stock will yield a signal of increase in the other stock.

2.2.3 Dynamic time warping

Dynamic time warping(DTW) is an algorithm utilized to discover an optimal alignment between two time-dependent sequences[26]. This technique was originally used for speech recognition to compare the difference between two speech patterns with different lengths. Using dynamic time warping is advantageous since it allows the algorithm to capture the difference between two sequences from a more macro perspective.

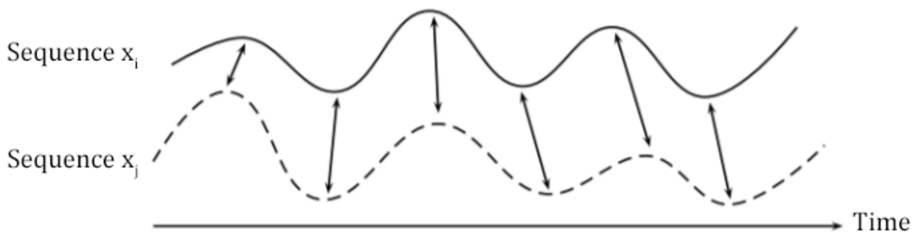


Figure 2.1: Two time sequences x_i and x_j

In Fig 2.1 displays how this technique measures the distance from peak to peak instead of measuring the distance along with a timeline. The input sequences do not need to be identical. Assume two stock feature sequence $x_i = (a_1, a_2, a_3, ..., a_n)$, $x_j = (b_1, b_2, b_3, ..., b_m)$, where n and m are the sizes of the corresponding sequence and they are not necessarily equal. We build a feature space base on x_i, x_j denoted by F, and $a_n, b_m \in F$. Based on this feature space we can have a cost matrix $C \in R_{n \times m}$ to measure the local cost between features $a, b \in F$, where each element in the cost matrix is defined by $C(n, m) = c(x_n, y_m)$. The c is a function as follow:

$$c : F \times F \rightarrow \mathbb{R}_{\geq 0} \tag{2.5}$$

The local cost measure normally uses absolute value of the difference. If the two features x, y are similar the cost c(x, y) is low; in contrast, the cost is high if they are different from each other. With this cost matrix, we can find the alignment between x_i and x_j that minimizes the costs[27].

Assume the an (n, m)warping path $p = (p_1, p_2, \ldots, pL)$, pl can be consider as the coordinate where $p_l = (n_l, m_l)$ for $l \in [1 : L]$. There's three rules restricting alignment of the optimal path:

1. The path must starts at the beginning points of two sequences and stops at the ends of the sequences, i.e $p_1 = (1, 1), pL = (n, m)$.
2. The path can not go backwards. If $p_l = (5, 5), p_{l+1} = (4, 6)$ is invalid since p_{l+1} must be greater or equal than n_l.
3. The step size is one for each search movement. For example, when $p_l = (5, 5), p_{l+1} = (7, 6)$ is invalid as it moves two steps. The only possible value for p_{l+1} in this example is $(6, 6), (6, 5), (5, 6)$.

These three rules must be satisfied simultaneously. The overall cost of a warping path between two sequence x_i and x_j is defined as :

$$C_{sum}(x_i, x_j) = \sum_{l=1}^{L} (c(a_{nl}, b_{ml})) \tag{2.6}$$

Where $c(x_{nl}, y_{ml})$ is the cost of the two corresponding features. Therefore, the optimal alignment is to minimize the warping costs overall.

Bring into the stocks data we are using, the size of feature sequences are equal. We use the DTW method to calculate the costs between the feature sequences, and the return of cost algorithm represents the similarity between the stock sequences. We define the graph of using DTW method as $G_D = (V, E_D)$, and the E_D is an edge matrix where $E_D \in R_{N \times N}$ and N is the number of stocks. Elements e_{ij} for $i, j \in [1 : N]$ in E_D is equal to the return of the warping cost function, and the graph G_D can be considered as the similarity stock graph. However, since we want to use graph to shows the relation between stock, the higher the weight on the edges means that the performances of two stocks are more similar[28]. In contrast, if the edge equals to the warping cost, the higher cost means that two stocks sequences have less similarity between them, which is in contradiction to what we expect. Therefore the edge value e_{ij} should be defined as follow:

$$e_{ij} = \frac{1}{min\left(C_{sum}(x_i, x_j)\right)} \tag{2.7}$$

So that the higher cost results in less weights on edges, which means that stocks sequences with higher warping costs have less connection between each other.

Compared to the correlation method, the time complexity is higher. Nevertheless, since each stock has a different reaction time to a change in the market, this method could more accurately test the similarity in two stocks' reaction to the change in the market. Besides, since the cost of warping path is always positive, we can avoid information loss when building the graphs.

2.3 Graph Neural Network

The graph neural network(GNN) model is first introduced in 2008[29]. The wide use of graph representation motivates the research of GNN. The GNN model is an extended version of current neural network methods that allows the model to deal with the data in graph domain. Currently, the GNN has been further extended into more specific models such as graph convolutional neural networks, graph attention networks, etc. The choice of graph type (directed graph, weighted graph, etc.) should drive the choice of the GNN model.

2.3.1 General graph neural network

Traditional machine learning uses a preprocessing algorithm that maps the graph structure into a simple representation to deal with graph-structured data. This preprocessing will omit critical information, such as topological dependency, which the final goal of the model may depend on. A GNN, in comparison, is an extended version of existing neural network methods and is designed for processing graph-structured data. In a GNN, the goal of learning can be represented as function$\tau(G, n) \in R_m$ whereτ is the function map graph G and n is one of its nodes into a vector of real numbers. The application of a GNN can be classified into two areas: graph focused and node focused[30].

1. Graph focused application: The function τ is independent from the node and classification (or regression) only depends on the graph structure.

2. Node Focused application: The function τ depends on the information of nodes and the aim of the classification (or regression) depends on the nodes.

In the GNN model a state $x_n \in R$ is attached to each node, where n is the attached node and the state contains the information based on the node's neighbour. The state x_n is used to produce an output on, and the output function determines the meaning of this output. The detailed definition is as follow:

$$x_n = f_w(l_n, l_{co[n]}, x_{ne[n]}, l_{ne[n]}) \tag{2.8}$$

$$o_n = g_w(x_n, l_n) \tag{2.9}$$

Where f_w represents the local transition function that summarizes the information of the node's neighbour, while g_w is the local output function that defines the output. The l represents the label, meaning l_n is the label of the current nodes and $o[n]$ is the set of edges that connect to node n, thus $l_{co[n]}$ is the label of the edges. The set of neighbour nodes connected with n is denoted by $ne[n]$, so that the $x_{ne[n]}$ and $l_{ne[n]}$ represents the states of the node's neighbours and labels of neighbours respectively. The equation can be re-written into the following format by stacking all the parameters together:

$$x = F_w(x, l) \tag{2.10}$$

$$o = G_w(x, l_N) \tag{2.11}$$

Where F_w is called the global transition function and G_w is the global output function. N is the numbers of the stacked elements. The model now takes a graph as an input and produces an output for each node.

An iterative scheme is applied to solve the above non–linear equation. For each state x_n, an iteration state t is attached to it.

$$x(t+1) = F_w(x(t), l) \tag{2.12}$$

$X(t)$ is the state under t^{th} iteration. Now the state $x(t)$ is considered as the state updated by the transition function based on the previous state. Therefore the output is written as:

$$o_n(t) = g_w(x_n(t), l_n) \tag{2.13}$$

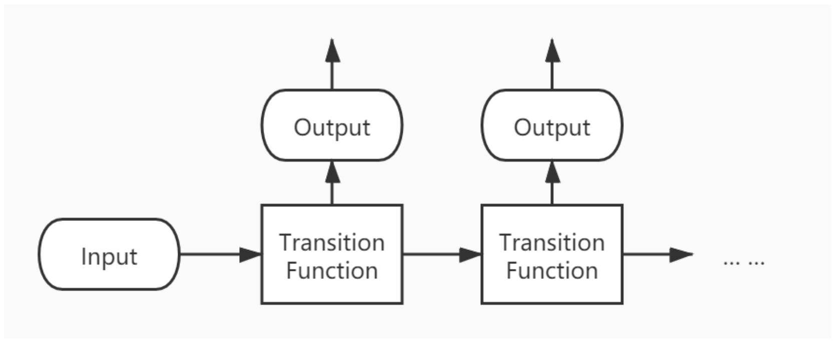


Figure 2.2: GNN model

The above two equations 2.12 and 2.13 can be used for the neural network unit. The model of GNN is shown in Figure2.2 which is similar to the recursive neural network model. Each unit stores the current state information, and the transition function active current unit. The output function is another unit that produces an output for each unit.

According to our problem, we aim to predict the performance of each. Therefore, the task is node-focused, and supervision is taken on every node. The learning algorithm for GNN is based on gradient descent, containing a forward and a backward function. To learn the parameters of f and g, we need a loss function, which is defined as follows:

$$loss = \sum_{i=1}^{p} (t_i - o_i) \tag{2.14}$$

Where p is the number of supervised nodes and t_i is the target information for a specific node. The states x_n are iteratively updated until they approach the fixed point where $x(T) \simeq x$ at time T. The gradient will learn from the loss function, and the weight will be updated according to the gradient.

In the GNN model, the transition function f_w is critical. The model has abundant power to handle most types of graphs. However, since the type of graph that we are using is fixed, this model is a bit surplus for our problem. Besides the neighbours' information, we want the model to focus on the features that individual nodes (stocks) contain. Moreover, we want to choose a simpler model to deal with the graph information.

2.3.2 Graph convolutional neural network

The graph convolutional neural network(GCN) is a graph neural network that is based on an efficient variant convolutional neural networks and it has a very good performance in chemistry problem[31] and paper classification problem[32]. The GCN model is inspired by first-order approximation of spectral graph convolutions[33]. The hidden layers learnt in the model encode the graph structure and the attributes of each node. The filter parameter is shared over all locations in the graph.

For the GCN model, the aim is to learn a function that takes the features on the graph as input and produces an output combining both nodes and graph information[32]. Therefore two necessary inputs are required:

1. A feature matrix X: this matrix has size $N \times D$ where N is the number of nodes and D is the number of node features. For each node n_i, it has features $(x_1, x_2, ..., x_d)$ where $d = D$.

2. A graph matrix A: an adjacency matrix that represents the structure of the graph. The size of A should be $N \times N$.

The output of this function is also a matrix denoted by$Z \in R^{N \times k}$ where k is the size of the output feature; this size is defined manually based to the requirement.

Therefore, each GCN layer can be written as:

$$H^{(l+1)} = f(H(l), A) \tag{2.15}$$

Where $H^0 = X$ and $H^L = Z$, the L is the number of layers.

The following is the basic form of a layer-wise propagation function:

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}) \tag{2.16}$$

Where A and $H^{(l)}$ is the graph matrix and the feature matrix as, respectively, mentioned above. $W^{(l)}$ is the *l-th* layer trainable weight matrix, and the σ is the activation function. The multiplication between adjacency matrix and feature matrix delivers the node information to their neighbour nodes. For each layer, it updates the node information for the next GCN layer. This propagation function has a main limitation that the adjacency matrix A is not normalised; thus, the multiplication between A and H will have an excessive change on the origin scale of the feature. Moreover, if the matrix does not contain self loops, it will lose features on the node itself. (As the diagonal of the matrix is 0, the result of multiplication on the diagonal will be 0.) Therefore, the layer-wise propagation function should be rewritten as:

$$\hat{A} = A + I \tag{2.17}$$

$$f(H(l), A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2.18}$$

Where D is the diagonal node degree matrix of $\hat{A}$ and I is the identity matrix. σ is the active function chosen manually. The propagation of the GCN is classified as the convolutional aggregator.

The propagation rule of the GCN can be considered as a combination of local transition function and local output function in the GNN. The propagation function of GCN integrates information from other nodes and produce an output for each node. Compared to the general GNN model, the GCN model is more concise. Although it is not as powerful as the general GNN, this is enough to process stock graphs.

The GCN is not a popular choice in stock prediction problem as the input of GCN requires historical feature and the stock graphs and the stock graphs are not provide directly from stock source. The paper[19] combine the LSTM and GCN, where LSTM is used to encode the input features, i.e. historical price and trading volume. According to their result, the GCN outperform the LSTM and linear regression model, and the joint LSTM and GCN model performs the best overall. In our research, we choose to use the GCN model as the benchmark, and we want to use this approach to investigate more on how graph improves the stock prediction.

2.4 Transformer

The attention mechanism was first proposed in 2014[34], has become popular in deep learning. The transform is a neural network that consists of attention mechanism; more precisely, the transformer block consists of self-attention and a feed-forward neural network. A trainable neural network model based on the transformer can attain a more accurate prediction by stacking the transformer block. The attention mechanism overcomes the limitation of a recurrent neural network (RNN) where the calculation of the current time step is highly dependent on the previous time in a RNN; the attention allows the calculation to process in parallel.

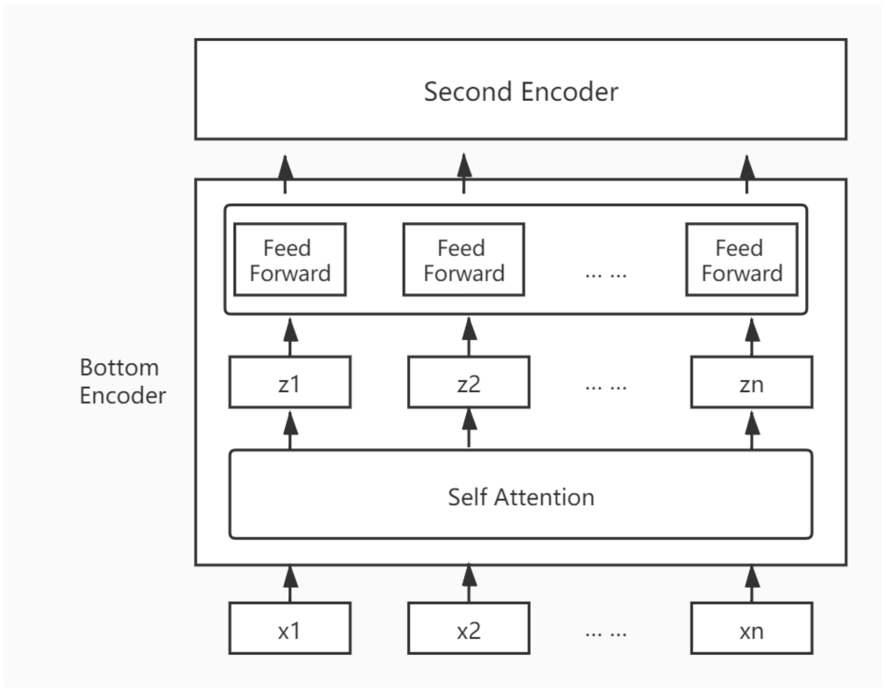


Figure 2.3: Encoder Structure

The transformer is essentially an encoder-decoder structure. The input enters the encoder block that consists of two sub-layers: a multi-head self-attention layer and a fully connected feed-forward network[35]. The structure of an encoder block is shown in Figure 2.3, and every encoder block is connected by a residual connection. The decoder layer is similar to the encoder layer apart from an additional attention layer. The encoder only receives a list of input-embedding vectors at the bottom encoder. The self attention layer takes the input $X = x_1, x_2, \ldots, x_n$ and produces an output $Z = z_1, z_2, \ldots, z_n$. The self-attention feature allows each vector to look at the positions of other input vectors, which can facilitate better encoding.

To calculate self-attention, three vectors are needed: a query vector, a key vector and a value vector(The vector sets are denoted by Q, K and V respectively).

$$Attention = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.19}$$

Where the QK^T produces a score and this score defines how much focus on other positions is needed. The score is then divided by the dimension of the key vector. This paper chooses to divide by the square root of the dimension in order to achieve a more stable gradient. The score needs to be normalised by passing through a softmax operation. The final operation involves multiplying the normalised score by the value vectors. In the actual training, for the bottom encoder, each of the Q, K ,V values are generated by multiplying the input vector with three weight matrices W_Q, W_K, W_V respectively.

The multi-head attention mechanism is to generate multiple different self-attention and concatenate the result matrix of self-attention. The concatenated matrix needs to be multiplied with a weight matrix to produce the final output of the multi-head attention layer.

The transformer model is highly efficient for training. The drawback of the transformer is that the model is not sensitive to positional information unless we use position embedding to fill this gap. This is an issue for sequential inputs if we use daily price information as the input. Nevertheless, this is not a problem if we use this model for the graph. Different from natural language processing(NLP) problem, the information we want to encode is undirect graphs. The swap between nodes should not affect the result. Take NLP problem as an example, according to the structure graph we showed in Figure 2.3, if we change the input feature order, i.e. swap the value of input word feature in x_1 and x_2, the meaning of the sentence will be different. However, since the self-attention mechanism is only sensitive to the input feature embedding, the change in word position does not affect the prediction result and the this not the result we expect in NLP. For example, "Alice likes dog" and "Dog likes Alice" will mean the same, if the position encoding is not given. In our stock prediction problem, the input order of nodes information is fixed, i.e. the stock we input always follow the alphabet order, we will not face the position problem as NLP. Besides, the change in node order does not affect the overall topological structure of an undirect graph.

3 Research Data and Methodology

3.1 Datasets

Before resorting to the new methods, we should set up a benchmark for this experiment. We used the data from the Yahoo Finance website. Since the stock market closes on weekends, the web page records five days of prices each week. Thus, the daily information for each stock over one year is roughly 260. We selected 504 stocks in total based on the Sector SPDR ETFs covering the stocks in different fields. The historical features for each stock include the daily open, high, low, close price, and trading volume. The research shows that these features provide a better result than using close price only[36]. The open price is related to the time when the markets open in the day. The high price refers to the highest price reached by the stock on that day and the low price is the lowest reached that day. The close price is the stocks price when the market is closed on that day. The trading volume is the total number of securities or contracts traded on that day.

The information for each stock covered the period from when the stock entered the market until 2018/08/27. Due to entering different stocks into the market at different times, some stocks only reflected the data from 2017, resulting in the problem of unequal sample size. Furthermore, the sample size for stocks recently entering the market was too small to comprise a training sample. Hence, 487 stocks were included in our final selection of experimental data sets from 2013 to 2018. In this section, we tested the quality of the stock features, hence, we selected 200 stocks starting from 2007 to 2012 as validation datasets to choose the model for the ultimate analysis.

In this test, 1131 days were used in total. Moreover, to analyse how sectors might affect a model's accuracy, we also selected different sectors to test if any specific sector might be more predictable than others. Due to the limited number of stocks, only eight stocks were contained in some sectors in total. Thus, we chose only five sectors of healthcare (61 stocks), industrials (71 stocks), consumer cyclical (80 stocks), technology (62 stocks), and financial services (74 stocks). These selected stocks were

Table 3.1: Sector abbreviations

HC	stocks belong to healthcare sector
IN	stocks belong to industrial sector
CC	stocks belong to consumer cyclical sector
TC	stocks belong to technology sector
FS	stocks belong to financial services sector

not included in the 487 stocks Dataset, for which the abbreviation is represented in Table 3.1.

3.2 Feature Selection

3.2.1 Single Stock Prediction

The first attempt is to predict the direction of the next stock close price on a single stock based on its past stock prices and volumes utilizing LSTM. Every input sample is denote by $X = (x_1, x_2, \ldots, x_n)$ where n is the size of historical features. We chose to use all the features that we obtained from the website (daily open, high, low, close price and the trading volume) and used a min-max normalization to normalise all the input features. The reason of using normalization is that the original value of price and volume is too big for back-propagation, and this may cause the problem of gradient vanishing. The normalization can scale the value into the interval $[0, 1]$. The formula for min-max normaliztion is as below:

$$\hat{x} = \frac{x - min(x)}{max(x) - min(x)}, \, x \in X \tag{3.1}$$

Where $\hat{x}$ is the normalized feature value, *max(x)* and *min(x)* denote the maximum and minimum value among the feature and the feature is calculated separately, i.e. the normalization of close price and volume where takes the maximum and minimum value of close price and volume respectively.

Our aim is to predict the next day price movement of the stock. The binary output Y is defined as:

$$Y = \begin{cases} 1 & if \, x_{close}(t) > x_{close}(t-1) \\ 0 & if \, x_{close}(t) \leq x_{close}(t-1) \end{cases} \tag{3.2}$$

The model consisted of two LSTM layers and an output dense layer. The activation function for LSTM was a rectified linear activation function (ReLU), and the sigmoid was chosen for the final dense layer as we were solving classification problems.

The result for predicting the stock price movement on single stock had about 50.11% accuracy. The result showed that the historical information for a single stock was not enough to predict the quote changes and it is hard to identify the optimal length of historical feature.

Instead, we chose to use data from multiple stocks to predict a single stock price. Hence, each sample was denoted by matrix X with size N and N represented the number of stocks that were used to assist for prediction of the chosen stock. The result of using multiple stock features to predict single stock performance was about 50.12%. This outcome showed that the method could not improve accuracy. Two possible reasons include the following: because stocks are independent, the performance of other stocks did not influence the chosen stock, and LSTM was incapable of processing the other stocks' information. Since the stock price is not independent, interaction should exist between stocks; therefore, it is more likely that the LSTM model caused a problem.

3.2.2 Multiple Stock Prediction

The experiment aimed to predict multiple stocks' price performance. Although the results showed that the LSTM model did not effectively improve a single stock prediction given information from other stocks, we decided to use a time distributed function to predict multiple stocks where weights were shared for all stocks during training instead of using other stocks' features for a single stock. This represented an extended model of predicting a single stock price given its past information.

The other reason we chose to use LSTM was because it could test whether the preprocessing on input features was correct for training. For this LSTM model, each sample was a matrix; each row of the matrix was the stock denoted by $X_N = (x_1, x_2, \ldots, x_n)$, and n was the number of features under consideration and N is the number of stocks. The feature included normalised daily closed price and trading volume. The procedure for generating the features of each stock sample was similar to the inputs of the previous single stock prediction model.

The model used three LSTM layers to solve a classification problem. The output was binary, denoted by Y, where $Y = (y_1, y_2, ..., y_n)$, n is the number of stocks, $y_n = 1$ if the next day stock price is greater or equal to the current day, or $y_n = 0$ if the price decreased. For this experiment, we selected 200 stocks, and the sample size was 1000. Besides, for each historical feature(i.e. close price) of the stock, we take ten days of data so that the n is equal to 50 as we have five different type of historical features.

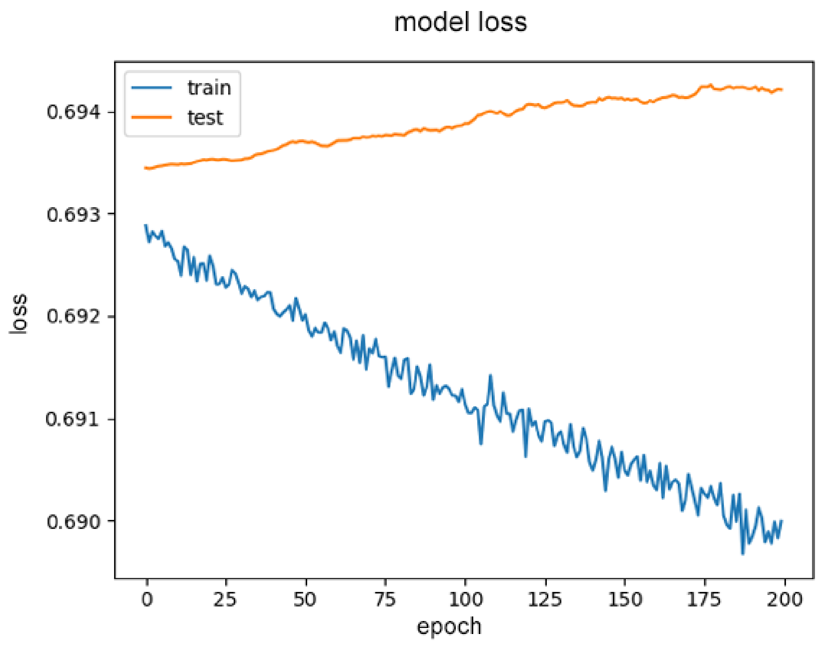


Figure 3.1: Loss of LSTM model with multiple stocks

The loss of using LSTM was as follows in Figure 3.1. The plot shows that the test loss did not converge although the training loss converged. One possible cause for the problem was that the normalised past prices and volumes did not make many contributions to the prediction of price for the next. Instead of using min-max normalisation, we defined a function that compared the daily price and volume:

$$X_n = x_1, x_2, \ldots, x_d \tag{3.3}$$

$$x_i = \ln\left(x_t - x_{(t-1)}\right), 0 < i < d \, d \in \mathbb{N}, \, x_i \in X_n \tag{3.4}$$

Where X_n is the feature vector of stock n with size d. x_i was the return price, and x_t is the stock price of at time t. The reason to use natural logarithm is to scale down the feature values to avoid vanishing in gradient. A similar comparison processing was performed for the volume of trading before adding it to the training. In addition, the size of the dimension of the input increased to three to include more information. The return not only compared the current price with that of the previous day but also made additional comparisons with prices from three and five days before. The days is chosen because there evidence shows that release of learning report will have impact on the stock price return and the level of the impacts is related to the lag time[37]. The lag time of the interim report is about three days and lag time of annual earning report will take a week. Therefore, the feature for a stock was defined as follows:

$$x = (x_1, x_2, x_3) \tag{3.5}$$

$$x_1 = \{(x_1, x_2, ..., x_d)\}, n \in \mathbb{N} \mid x_i = \ln(x_t - x_{t-1}), 0 < i < d \tag{3.6}$$

$$x_2 = \{(x_1, x_2, ..., x_d)\}, n \in \mathbb{N} \mid x_i = \ln(x_t - x_{t-3}), 0 < i < d \tag{3.7}$$

$$x_3 = \{(x_1, x_2, ..., x_d)\}, n \in \mathbb{N} \mid x_i = \ln(x_t - x_{t-5}), 0 < i < d \tag{3.8}$$

Now for each stock, the feature was a $3 \times N$ matrix. The loss of using adjusted inputs with $t = 10$ was as showed in Figure 3.2. Compare to feature using only the min-max normalisation on the value, the loss of test sets decreases. The loss shows that the increment in price is a better input feature for the stock prediction problem.

3.3 Stock Graph Construction

The input for GCN consists of two parts: a graph and a feature matrix. Graphs of the stocks were not available on the websites; thus, we attempted to use the methods introduced in the literature review section to

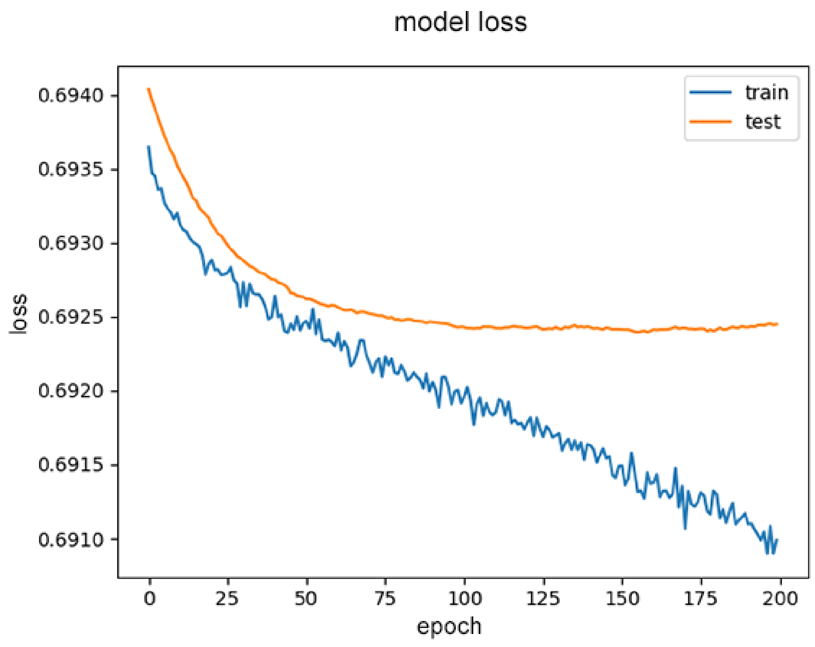


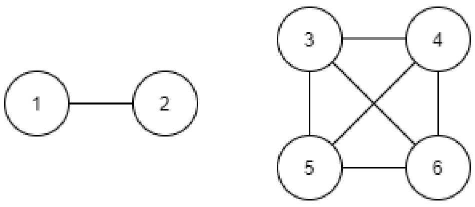
Figure 3.2: Loss of LSTM model with prepocessed multiple stocks

generate graphs and compare how different graphs might influence the model's accuracy. Compared to the LSTM model, the GCN model was capable of processing the internal information for stocks via the graph. Accordingly, we expected to see an improvement in accuracy.

3.3.1 Sector graph

We decided to use three different methods to generate stock graphs. The idea was to compare three different types of graphs generated by distinct methods: sector-based, correlation distance and DTW.

The sector-based graph is the most straightforward graph as it only considers whether the stocks belong to the same sector. This type of graph is only partially fully connected if two stocks are in the same sector. Moreover, the sector graph has no weight on the edges, so the corresponding adjacency matrix is a binary matrix. A sample of the sector graph and the corresponding adjacency matrix is shown in Figure 3.3.



$$\begin{array}{cccccc}
0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{array}$$

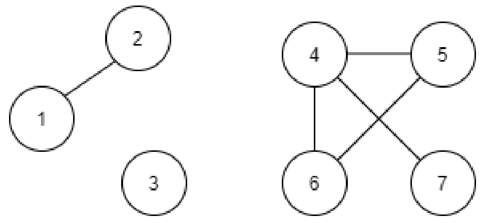
Figure 3.3: Sector based graph and the correspoding adjacency matrix

3.3.2 Correlation graph

Note that the correlation graph is not fully connected, and for this study, a threshold was set to make the stock graph unweighted. Differ from the sector-based graph that is fully connected when the stocks are in the same sector, the correlation graph is partially correlated, where the stock has an indirect dependency on other stocks[38]. For each stock, we used 60 days of return close prices to calculate the correlation as three months has been proved the optimal setting for building a correlation graph[39]. The return price function is $r_i(t) = \ln\left[\frac{p_i(t)}{p_i(t-1)}\right]$,where the t is the day and *p(t)* is the price on that day. The return price is equal to the natural logarithm of the current price and the price from the previous day. Correlation revealed that the threshold we were using for the edges was set to 0.85. A sample graph according to the paper and the corresponding adjacency matrix is shown in Figure 3.4.

3.3.3 DTW graph

The DTW method measured the similarity between two stocks' price sequence. The edges of the DTW-based graph indicated whether the stock price range was consistent. The input for DTW was similar to the correlation method, which took 60 days' values for each stock price. Although the DTW method allowed unequal input length, defining a different time



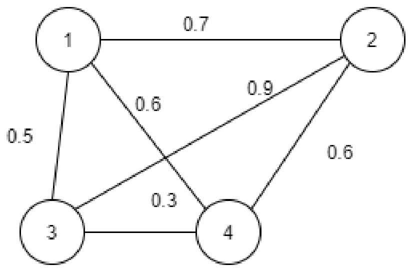
$$\begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{matrix}$$

Figure 3.4: Correlation graph and the correspoding adjacency matrix

range for each stock would be difficult and even unreasonable. Since the correlation graph was unweighted and partially connected, to increase the diversity of graph types, we kept the edge values of the DTW-based graph, causing the graphs to be fully connected and weighted. A sample graph and corresponding adjacency matrix are shown in Figure 3.5.

3.4 Model for Benchmark

The benchmark for the experiment used LSTM and GCN, and the model aimed to predict whether the stock price would increase the next day compared to the current price. The reason for choosing the RNN model was that the input feature for the stock was chronologically formed and the RNN model was suitable for time-dependent data[40]. The future performance of a stock is relevant to its past performance, and LSTM was able to catch that hidden information during training. The GCN has been applied to stock prediction and proved effective. Since the datasets we are using differ from that in the paper[19], the setting will be slightly different.



$$\begin{matrix} 0 & 0.7 & 0.5 & 0.6 \\ 0.7 & 0 & 0.9 & 0.6 \\ 0.5 & 0.9 & 0 & 0.3 \\ 0.6 & 0.6 & 0.3 & 0 \end{matrix}$$

Figure 3.5: DTW graph and the correspoding adjacency matrix

3.4.1 GCN Prediction with Graph Only

The first attempt used three GCN layers with the graph only to predict the direction of the next day's stock close price movement(i.e. whether the stock close price for the next day would increase). The layer setting we use is similar to the paper[19]. The graph was denoted by matrix *G =(V,E)* where *V* is the vertices(nodes) of the graph and *E* is the set of edges. *A* is the adjacency matrix of G with size *N*, where *N* was the number of stocks. Each row of *A* indicated how the current node connected with other nodes (i.e. the relation between nodes). According to Kipf's paper[32], the GCN layer without feature input already evidences an excellent performance on the Zachary karate club network[41].

The accuracy plot shown in Figure 3.6 uses correlation graph only. In this experiment we selected 200 stocks, the sample size was 1000, the graphs were correlation-based and the days taken for generating graph is 60 according to the paper[39]. Therefore, according to the GCN propagation rule in Formula 2.13, the matrix multiplication for each GCN layer fully depended on the normalised adjacency matrix.

However, the accuracy plot indicates that GCN without the feature information cannot present a precise prediction on the future direction of the stocks. This result may be caused by two possible causes. First, unlike Zachary'skarate club problem with a graph displaying a clear and distinct cluster, the stock market graph generated by the correlation method changed over time, causing unclear and unstable clusters (the

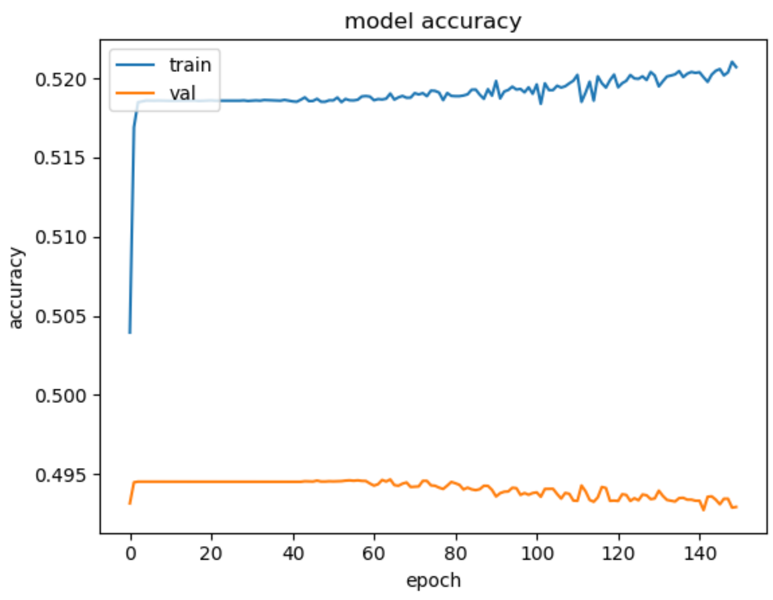


Figure 3.6: Accuracy of GCN with correlation graph only

same stock may belong to different clusters over time). We also tested the sector-based graph and DTW graph. Although the sector-based graph was fixed for the entire training process, the result was similar to the correlation-based graph in terms of accuracy less than 50%. This proves that the edges' information does not provide enough information in stock prediction. In the stock price prediction problem, the information about the stock itself was more important than the graph. Thus, it is essential to include stock features, and the graphs should work as an inductive bias rather than a major indicator.

3.4.2 GCN Prediction with Graphs and Features

According to the propagation rule mentioned in 2.3.2, the input feature size must be $N \times D$, where D is the number of features. Therefore, each stock feature required a vector instead of a matrix. To reduce the dimension of stock feature, we could choose to flatten the matrix into vector form or alternatively add an encoding layer before putting the features into the GCN layers. In this section relates how we chose to flatten the matrix simply, because we want to test how the LSTM based encoding improves the accuracy in the later section.

The model was formed of three GCN layers; the active function used ReLU for the first two layers and sigmoid for the output layer. The graphs needed to feed into each GCN layer. For the sector-based graph, the feed-in graph did not change throughout the training, while for the remaining types of graph, the graph changed based on the sample. The model with three GCN layer meets the following form:

$$\tilde{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}} \tag{3.9}$$

$$Y = softmax(\tilde{A}ReLU(\tilde{A}ReLU(\tilde{A}XW^{(0)})W^{(1)})W^{(2)}) \tag{3.10}$$

Features included trading volumes, return open, high, low and close prices. The preprocessing step was the same as mentioned in section 3.2.2 with the exception that the output matrix the feature size is $N \times D$, where N represented the number of days selected.

3.4.3 LSTM + GCN

Hence, was indicated that GCN produces a better result than the LSTM model. The original input feature utilized for LSTM is a matrix giving the input 4 dimensions (batch size, number of stocks, 3xN feature). As mentioned, the input feature requires a vector instead of a matrix; thus, we flattened the matrix into a vector form. Instead of flattening the matrix, we add an LSTM before the GCN layers. The LSTM layer performs as an encoder layer, producing an embedding for each stock. The LSTM result is brought into the GCN layer with the stock graph as the input. The model structure is represented in Figure 3.7.

The main mechanism in LSTM was the input, forget and output gate. The forget gate allowed LSTM to determine whether the information was useful. As LSTM has proved powerful on long-term dependent data, the LSTM layer was expected to produce a better stock representation than the normalized features. In the code, the return sequence was set to true so that LSTM could generate an output at each neuron. The GCN layer integrated the encoded stock information, and the updated stock information was then passed to the next GCN layer. In the experiment, we chose to use three GCN layers, which could each be considered as a walk process. An increasing number of layers meant more in terms of what each node could receive from other nodes. Since the GCN layer updated each node based on neighbour- and self-information, the

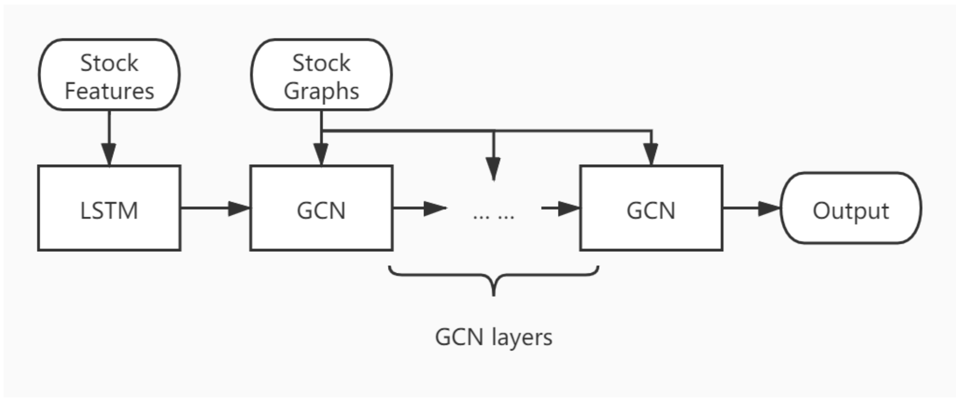


Figure 3.7: Structure of LSTM+GCN

increase in layers allowed the node to learn information that was not directly connected to it. However, nodes having no direct connection meant that the impact on the target node was relatively small compared to that of the neighbour nodes. Therefore, the number of GCN layers should be limited.

The main mechanism in LSTM included the input, forget, and output gates. The forget gate allows LSTM to determine whether the information was useful. As LSTM is proved powerful on long-term dependent data, the LSTM layer was expected to create a better stock representation than the normalized features. In the code, the return sequence is set to true so that LSTM could generate an output at each neuron. The GCN layer integrates the encoded stock information, then, the updates stock information is passed to the next GCN layer. An increasing number of layers meant more in terms of what each node could receive from other nodes. Since the GCN layer updates each node based on neighbor- and self-information, the increase in layers allows the node to learn directly connected information. However, nodes with no direct connection means that the impact on the target node was relatively small compared to that of the neighbor nodes. Therefore, the number of GCN layers should be limited.

3.4.4 Transformer + GCN

Instead of using LSTM, the second attempt sought to replace LSTM via the transformer to learn the embedding of stocks. Since the transformer block could not process the matrix-formed feature, we flatten the matrix into vector form in preprocessing, yielding a three-dimensional input with shape as the number of stocks and the features (vector). The structure of this model is shown in Figure 3.8, which was similar to that of the LSTM + GCN model except for LSTM replaced by a transformer block.

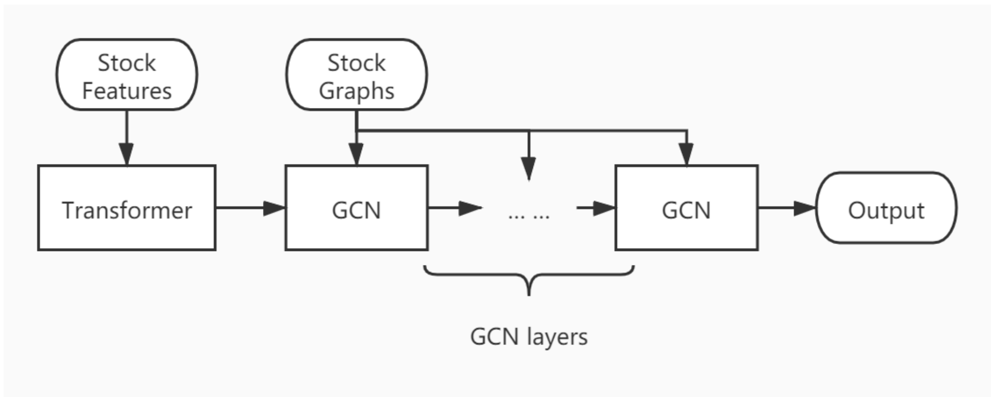


Figure 3.8: Structure of Transformer+GCN

The transformer is essentially an encoder-decoder structure. The input for the first encoder layer was the stocks' feature (normalised open, close, high, low price and volume in vector form). Each encoder contains self-attention and there is a residual connection between encoder and decoder in each encoder layer. In the NLP field, this should be well provided to encode information than the LSTM[35].

3.4.5 Sample results

Before we go to the larger data sets, we should check the effect of using the encoding technique. We test the model on samples with 200 stocks and used three GCN layers for all models, for which the results are presented in Table 3.2.

The results proved that the accuracy was enhanced by the encoding layer. Keeping the same number of GCN layers, we placed an LSTM layer before the GCN as an encoder. The accuracy of the transformer model resembled that of the LSTM+GCN model on this set.

Running all methods using sample data involving 200 stocks yielded better results compared to the normal GCN model. Although we expected that the transformer could lead to further improvement than LSTM,

Table 3.2: Results of using simple GCN model and joint encoder and GCN model. The test is on small sample datasets.

Model	Accuracy(%)
GCN	52.78
LSTM+GCN	53.95
transformer+GCN	54.12

the results indicate that the transformer and LSTM exhibited similar performance. In terms of the time complexity, the transformer takes longer than LSTM. Although the transformer performed well in natural language processing(NLP), the ability to encode stock features is not as powerful as expected. This could be caused by two possible reasons. First, the transformer model typically performs much better than LSTM on most of the problems, especially on NLP, however, the experimental results demonstrated that the transformer did not surpass LSTM on small datasets[42]. The transformer performance is limited, and the model is easily overfitted when applied to a small dataset. Second, the main part of this model is still based on GCN. As a result, the advanced encoding method has no significant impact on the final prediction accuracy.

4 Implementation

4.1 GCN with multiple graphs

A traditional GCN input only takes one graph because the relation between nodes is consistent. Our idea is to give GCN multiple graphs, instead. The graphs generated for stocks in our work had different meanings on their edges. Thus, various graphs should be provided for the model with different information. The method used concatenation of the output of GCN with different graphs and applied a feed-forward layer to the concatenated output.

According to the propagation rule of GCN in Formula 2.16, the normalized adjacency matrix will be multiplied with the feature matrix, where the size of the normalized adjacency matrix is $N \times N$ and the feature matrix is $N \times D$. Then, the multiplication with a weight matrix of size $D \times k$ is connected and the size of the GCN output should then be $N \times k$. Instead of inputting only one graph, we concatenated the normalized adjacency matrices of graphs for the prediction. For each stock graph, we applied the GCN layer to it, and we concatenated the output of the GCN layer to produce a new vector. The structure of the model is shown in Figure 4.1. We let the output of each last GCN layer with the size $N \times 1$, hence, the size after concatenation will be $S \times 1$, where the value of S is the number of input graphs times N.

Since the output should produce a vector denoting whether the stock price increased or not, the output size should be $N \times 1$. Therefore, we connected a feed-forward layer to the GCN to produce an output with a size of $N \times 1$.

Compared to the normal GCN model utilizing only one graph as input, the combination of various graphs should provide the model with more information. With the help of multiple graphs, we expect to outperform the traditional GCN model by the result.

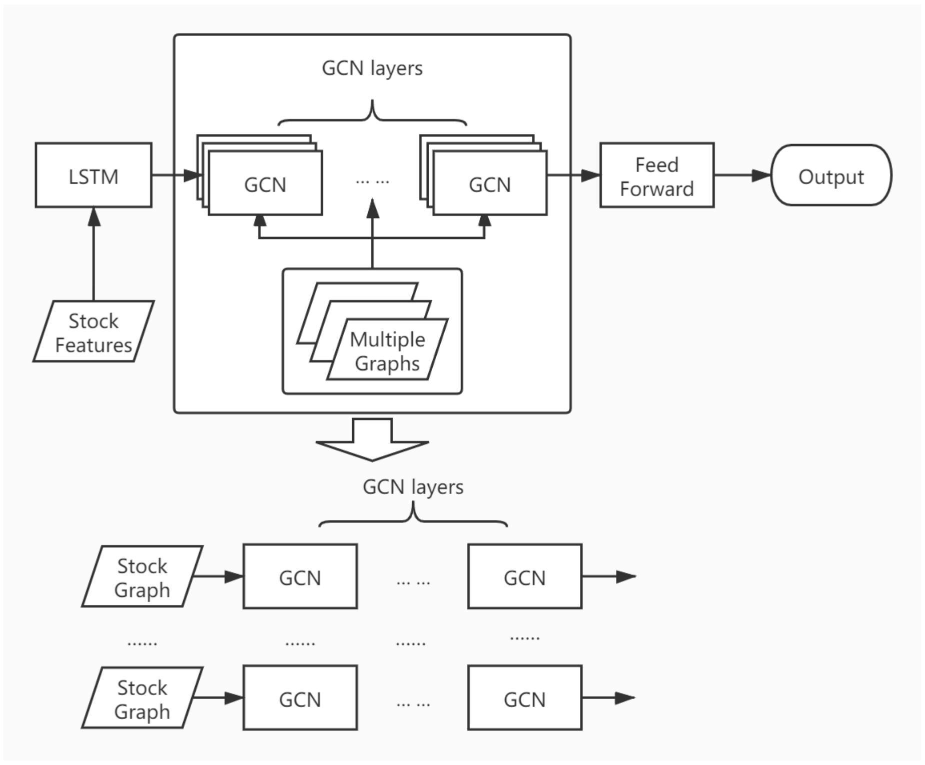


Figure 4.1: Structure of Multi-graph GCN

4.2 Transformer for stock prediction

The reason for using the GCN in our work is to allow the model not only to focus on the stock's own features but also to learn from other stocks' features. The results of using LSTM shows that the model is not suitable for processing multiple stocks. The multiplication in the GCN allows the nodes to pass information to each other. The attention mechanism has a similar function in terms of passing others' information.

In other words, we were seeking a model to learn stock embedding using both self and neighbors' information. The attention mechanism in the transformer, which is extensively used in the natural language processing field, will allow the model to locate the critical feature. Treating each stock as the words in a sentence, the attention mechanism learns a weight for each word in the sentence. Different weights on a word indicate the degree of influence on the current word. Thus, attention can find the degree of influence of other stocks on the current stock.

The "limitation" of the GCN is the fixed graphs. It means that the information exchange from nodes to nodes is pre-defined. Therefore, the method requires the graphs to have a strong relation to the prediction aim. This limitation is not important in Kipf's paper[32] since the graphs normally utilized in the GCN provide critical information to the prediction. The problem presented in Kipf's paper[32] is related to paper classification in which the problem is the graphs based on the paper citations. The difference between stock graphs and citation graphs is that the citation graphs have a direct impact on a paper's classification. Papers with the same citations are more likely to be in the same field. Moreover, the example involves semi-supervised learning where the label for some of the nodes is already known and the aim is to predict the remainders' paper type. However, the stock graphs generated in our work had a direct contribution to the next day price performance. Furthermore, it is not reasonable to have a stock graph with some of the node's label known since the movement of the stock price can not be known in advance. Figure 4.2 represents the close price changes of two stocks in the same sector. The plot shows that it is hard to determine if one stock will increase based on the change in the other stock in the same sector.

Even though the graphs could enhance the information not contained in the features, it is difficult to judge whether the defined relations between stocks have significant impacts on stock performance. For instance, the sector graphs indicate whether the stocks are in the same sectors, however, the stock performance varied even for stocks in the same sectors. Although some sectors performs generally better than others, it is difficult to apply this information to the daily stock price. The results of using correlation graphs were better since the highly correlated stocks were more likely to provide important information, however, it is difficult to assert that such graphs are optimal selections for the price predicting problem.

4.2.1 Transformer + LSTM

Compared to how a GCN passes information from nodes to nodes, transformers eliminate the prior knowledge. As a first step, we created three vectors for each stock: *query*, *key* and *value*. The first step is to create these three vectors by multiplying the original stock embedding by the corresponding trainable matrix. The second step is to calculate

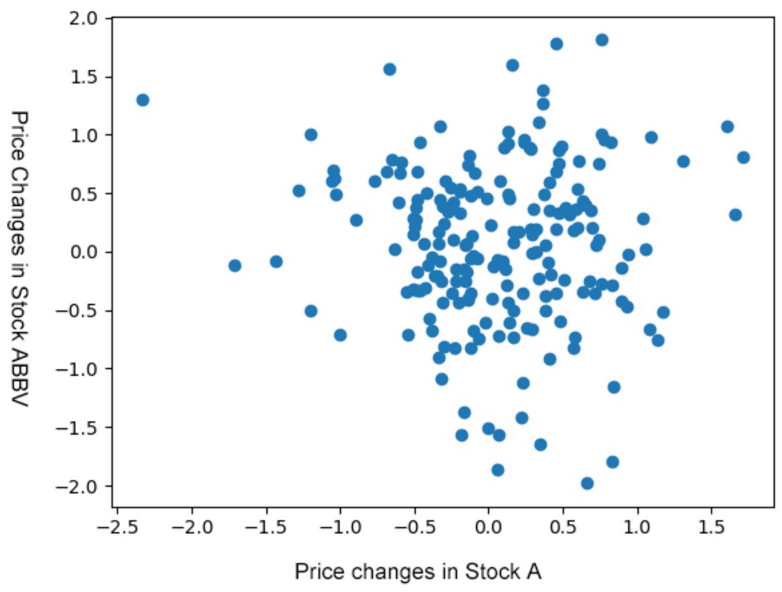


Figure 4.2: Plot of price change of two stocks that in the same sector over the same time period.

the score, in which the information from other stocks was incorporated. This score determines the degree to which the model should focus on other parts. In other words, this was equivalent to building a stock graph. Assuming the score denoted by vector S, the size of the score is N where N is the number of stocks in our problem. Each stock has a query and key vector, denoted by q and k respectively. The elements in S_n are defined as follows:

$$S_n = (s_1, s_2, ..., s_N) \tag{4.1}$$

$$s_n = q_n k_n,\ 0 < n < N\, n \in \mathbb{N}, \tag{4.2}$$

where n represents the n^{th} stocks. The third step includes dividing the score by the key dimension to have a stable gradient and pass the results through a softmax function. The score generates by the softmax function indicated the extent to which each stock is expressed under the current stock. Normally, we try to place more attention on the current stocks. The

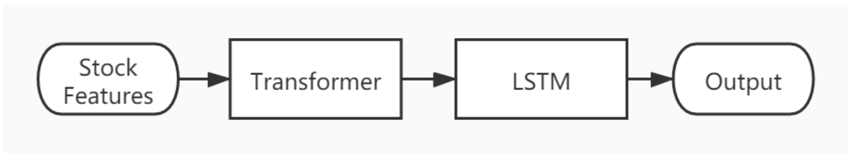


Figure 4.3: Structure of tranformer+LSTM

fourth step is to multiply the value vectors by the corresponding softmax output to produce two new vectors with the same size as the value vector. In this step, the attention mechanism integrates information from other stocks. The ultimate step sums these two vectors and produces the self-attention output for the n^{th} stock. In practice, these are all performed in matrix form to enhance the processing speed.

The multi-head attention makes the transformer different from normal attention mechanisms, allowing the attention to combining information from different aspects. This function is similar to what we expected with the multiple-graph GCN. [43]The Figure 4.3 shows the structure of joint transformer and LSTM model.

Using this simple model structure, we connect the transformer with LSTM. Pure LSTM presents a problem where the model only concentrated on the current stock; in contrast, the input feature through the transformer already contained the information from other stocks. Thus, the result should be improved in comparison to the pure LSTM model. Within the transformer, we utilized 20 self-attention layers. Problems such as the natural language processing problem normally require position embeddings to record the order of the inputs since the order of words in sentences is critical and the words in different positions will have different references. Nevertheless, the order of stock was not important since a change in stock position should not influence the prediction result. Therefore, position embedding is not used in the present model.

We examined this model on the same datasets (200 stocks), yielding an accuracy of 54.25%. In comparison to the model using transformer+GCN, a similar performance revealed that the transformer can replace the way for GCN passing the information. It proves our presumption that the GCN model needed the input graph to be highly correlated with the prediction aim. We expect that the result could be better on larger datasets.

4.2.2 Transformer with graph masks

The transformer's results on small samples demonstrated its ability to solve the stocks prediction problem without using graphs. Contrary to using the graphs fed into the GCN model in the present work, as a medium to convey information, we employed graphs as a filter instead. The possible reason for the non-optimality of GCN for stock graphs is that the GCN model relies on the information given by the graphs. However, owing to the complicated character of the stock market, it is difficult to have a graph able to summarize the whole market stock relation. Therefore, we used a graph to assist the transformer rather than having graphs as the main factor. In other words, the idea is to apply the graph as a mask to the score matrix to make a sparse tranformer[44].

According to the formula of attention, multiplying between the query matrix and the key value will produce a matrix output with size N where N is the number of the stock. This is where we could apply a mask: a matrix that would decide the possibility of obtaining information from the original source.

In the NLP problem, the mask is applied on the upper triangle of the matrix since, in practice, words at the beginning of a sentence should not contain information related to words not yet appeared in the sentence. Our idea is to replace the masks with a binary stock graph. Taking sector graphs as an example, when applying the sector graphs to the score matrix, the stock will only learn the embedding from the stocks in the same sector. The function is as follows:

$$Attention = softmax\left(\frac{QK^T}{\sqrt{d_k}} \cdot A\right) V, \tag{4.3}$$

where Q, K and V are the matrix form of query, key and value, respectively. The adjacency matrix (must be binary) was denoted by A, and k denotes the dimension of the key. The dot product removed scores not taken into consideration. The graph mask and multiplication between normalized stock graphs and features were different since the score matrix of stocks was trainable, however, the multiplication of normalized stock graphs and features was fixed.

5 Experiment and Evaluation

5.1 Preparation for Experiment

We use the stock data from the New York Stock Exchange and the data generated from the Yahoo Finance website. We selected 487 stocks based on Sector SPDR ETFs for testing (each containing data between 2013.11.18 and 2018.8.27). The details are explained in Section 3.1. Moreover, we also selected five sectors, healthcare (61 stocks), industrials (71 stocks), consumer cyclical (80 stocks), technology (62 stocks), and financial services (74 stocks), to test more predictability of the specific sector.

For each sample stock, the features included the stocks' trading volume, open, high, low, and close price. Preprocessing is required for all input features according to Section 3.2.2.

We utilized the LSTM and GCN joint with LSTM as a benchmark. For the GCN model, we will test with different graphs generated from Section 3.3 and chose the optimal result as the benchmark.

The prediction target is a binary vector recording whether the close price incremented in comparison with the open price.

Since more than 50% of the stocks could face a price increase, we need to calculate the average percentage of increase to avoid an extreme case where the model was not learning, i.e. the model would consider increasing all the stock prices. We expect the model accuracy to exceed the percentage of the price increase. The average percentage of increment in the testing sets is shown in Table 5.1.

All the models are built-in Python utilizing the Tensorflow framework. The main objective of the experiment is to examine the improvement of prediction accuracy by the constructed graphs. We expect that the result of the proposed model using graph information surpasses the result produced by the model without using graphs. Moreover, we want to examine whether the outcome of the model with the information of multiple graphs can outperform the general GCN model. Additionally, the experimental data including different sectors can examine if certain sectors are more predictable than others. We split the datasets into training, validation, and testing parts. The details of the split are shown

Table 5.1: Average percentage of increase in stocks' price

Sectors	Percentage of increase in price(%)
Healthcare(HC)	50.13
Industrials(IN)	50.21
Consumer Cyclical(CC)	50.21
Technology(TC)	49.97
Financial Services(FS)	50.23
487 stocks	50.31

in Table 5.2.

Table 5.2: Split of datasets

Sectors	Training	Validation	Testing
Healthcare(HC)	48,251	7,991	12,749
Industrials(IN)	56,161	9,301	14,839
Consumer Cyclical(CC)	71,190	11,790	18,810
Technology(TC)	49,042	8,122	12,958
Financial Services(FS)	58,534	9,694	15,466
487 stocks	385,217	63,797	101,783

5.2 Results of Using Various Graphs

The graphs generated in the present work follows the method presented in Section 3.3. we tested these graphs with a pure GCN model on the 487 stocks. The test model included three GCN layers and we used binary cross-entropy as the loss.

Table 5.3: Accuracy of GCN with different stock graphs

Graph Type	Accuracy(%)
Sector based	51.87
Unweighted correlation	**52.78**
DTW (fully connected)	52.13

The compared results are presented in Table 5.3. The accuracy of GCN with scale-free correlation graphs is around 52.78%. The weighted fully connected DTW graphs and sector graphs could not play a positive way in the prediction. Since the definition of matrix normalization that summary of each element in the row is unity, the values in the matrix after normalization are very small. In addition, the fully connected graph remain non-zero elements. The model utilized multiplication between the normalized matrix and the features matrix to merge information from other stocks. Therefore, multiplication between features and the normalized matrix caused the elements in the output matrix to split evenly. Although the weight matrix is added in the propagation, the model lost its focus since the values in the matrix were almost evenly distributed. Compared to the weighted graphs, the unweighted correlation(binary) graph produced a better result. In the paper[25], it compares the weighted and unweighted correlation. The experiment in the paper indicates that weighted correlation makes the connection of the graph become completely random and causes the learning become difficult to concentrate on the critical information.

The result of using sector-based graph is the worst thus far. Although both sector-based graphs and unweighted correlation graphs are partially connected and binary, the sector-based graphs are fixed and unchangeable throughout the time. Since the market information is changing over time, the sector may not provide enough information to support the prediction. This result revealed that the sector-based the graph was not appropriate for the price prediction problem.

5.3 Parameter Setting

In section 3.2.2, we explain how the historical feature selection and preprocessing affect the training loss. We tried various setting on the number of days data included in the stock feature(5, 10, 20, 60 days denote one week, two weeks, one month, and three months respectively, one week include only five days data as the market closes on weekends). We want to select the optimal feature set for the model. Table 5.4 represents the result of using the various lengths of historical features.

Table 5.4: Accuracy of LSTM+GCN with different length of historical features

Length	Accuracy(%)
5 days	52.13
10 days	**54.23**
20 days	53.78
40 days	53.57
60 days	53.32

According to the accuracy table, the GCN using a 10-days historical feature performs well than others. We also plotted a histogram in Figure 5.1 indicating the 5-days of past performance is not enough for the prediction. However, the increase in the length of days does not enhance the accuracy necessarily. The best setting for lookback window is 10 days where the accuracy is 54.23%. The accuracy decreases gradually after the 10-days length and this may due to excessive information. From the rest of the experiment, we will use the 10-days of historical information as the input feature size.

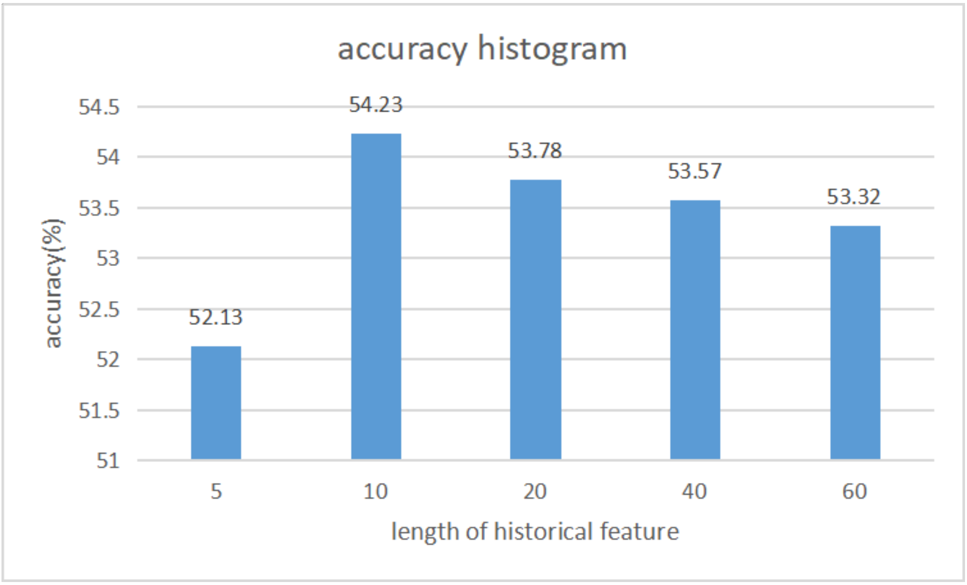


Figure 5.1: Accuracy histogram of LSTM+GCN with different length of historical

According to the GCN propagation rules, we can define the output size of each GCN layer manually. The size of the output will become the feature size of the next GCN layer. In the experiment, we evaluate the model with different size of output. The model we use for evaluation is consists of two GCN layer, so we adjust the output size for the first GCN layer. The output size of the second layer is always fixed according to the required target format. From the histogram in Figure 5.2, the result of having 128 as input size produce the best result 53.93%. The increase in dimension of output does not necessarily produce a better result. The worst performance from our result is 52.78% when the output dimension is 256. From the plot, we can see that the accuracy gradually decreases after it reaches its best performance.

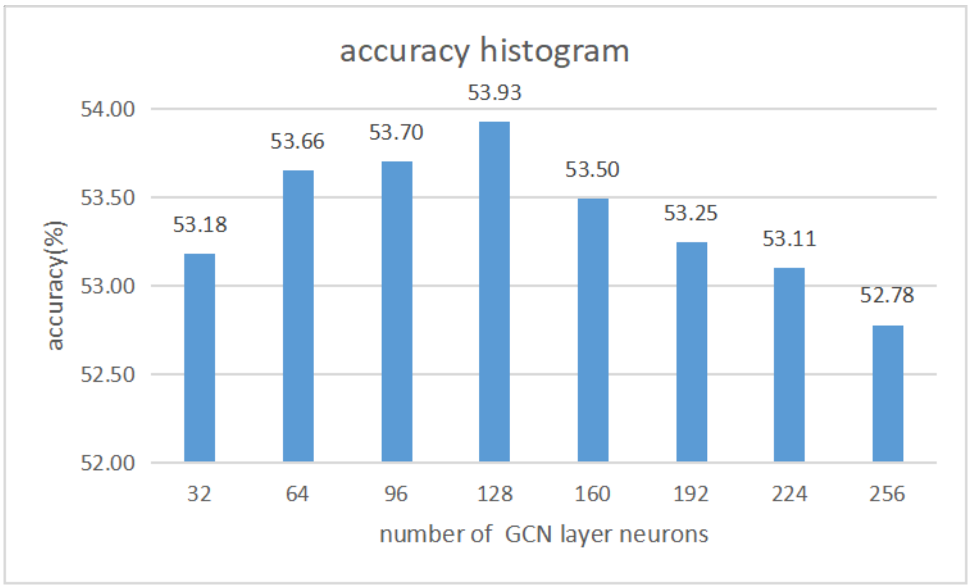


Figure 5.2: Accuracy histogram of LSTM+GCN with different neurons

Apart from the neurons, the number of GCN layers also influences the prediction accuracy. Each GCN layer is a process of aggregating the information from the neighbour stocks, and the increase in GCN layers is equivalent to increasing the length of a walk in the graph. In graph theory, the length of a walk is the number of edges included in a walk and walk is a finite or infinite sequence of edges and vertices[45]. The increase in GCN layers means that the nodes can gather information from the nodes that are not directly connected to it and the range it can

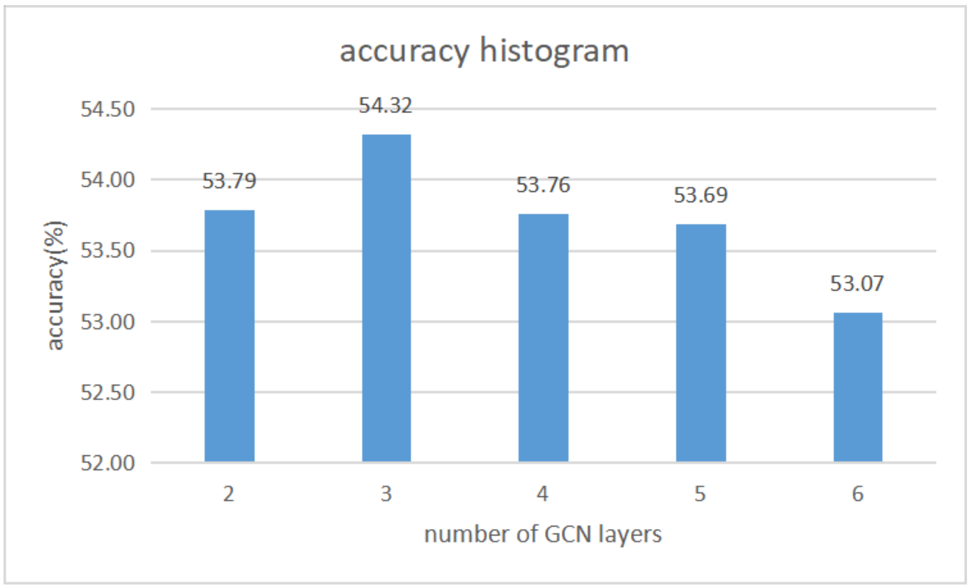


Figure 5.3: Accuracy histogram of LSTM+GCN with different number of GCN layers

reach depends on the number of GCN layers. However, the distant nodes should have less impact on current node, so the increase in layers not always improves the accuracy.

Then, we build an LSTM+GCN model to evaluate the impact of layers. In our model, the number of neurons is identical in each GCN layer except the last one. We start from two GCN layers to examine the model with a different number of layers and test five different configuration in total. The corresponding histogram is shown in Figure 5.3. The model reaches its best performance, 54.32%, when using three GCN layer, and the accuracy gradually decreases after that.

Overall, the best parameter setting for GCN model is using three layers with 10-days of lookback window and 128 layer-wise output size.

5.4 Multi-graph GCN Results and Evaluation

In this part, we test the multi-graph GCN model on six different datasets compared to our benchmark, i.e. LSTM, and GCN. Since the stocks in the same sector is a fully connected graph, we use industry type to distinguish the stocks as mentioned in section 2.2.1. According to the discussion in Section 5.4, we choose to use the combination of sector graphs, the correlation graph, and DTW graphs. Although the sector graphs have not produced a good result, we still keep it for the only graph that contains information unrelated to stock prices and trading volumes.

Table 5.5: Experiment results of on stock price movement with multi-graph GCN compared with benchmark

	Accuracy(%)					
Model	HC	IN	CC	TC	FS	487 stocks
LSTM	50.67	51.13	51.13	50.87	51.03	51.13
GCN	52.27	52.51	52.55	52.32	52.43	52.78
LSTM+GCN	53.89	53.51	53.13	53.67	53.51	54.32
Multi-graph GCN	**54.21**	**54.02**	**54.18**	**54.73**	**54.43**	**54.89**

The results in Table 5.5 showed that the multiple GCN has no significant improvement compared to the general LSTM+GCN model. It is observed that the accuracy of using multi-graphs GCN model is about 3.7% higher than the LSTM model and the result on other sectors is also about 3.5% higher than the LSTM model. The results indicated the effectiveness of the way in combining graphs with the neural network method for predicting stock performance. However, since the graph was critical to the GCN based model, the performance could be improved more by generating a more suitable stock graph. Neither the sector-based graph nor the DTW graph was the optimal selection for the stock prediction problem as tested in Section 5.2. The results from using sector-based graphs were not decent compared to those demonstrated by correlation graphs. It is indicated that the information provided by the sector-based graphs is limited. Therefore, the model learning highly depends on the correlation graphs which causes prediction results similar to the LSTM+GCN model.

5.5 Transformer with Graphs Results and Evaluation

In this part, we test the transformer model and transformer with a graph mask model on six different datasets compared to our benchmark, LSTM and GCN. According to section 4.2.2, the mask must be binary, hence, we utilized correlation graphs as it produces a better result in Section 5.2.

Table 5.6: Experiment results of on stock price movement with transformer compared with benchmark

	Accuracy(%)					
Model/Sample	HC	IN	CC	TC	FS	487 stocks
LSTM	50.67	51.13	51.13	50.87	51.03	51.13
GCN	52.27	52.51	52.55	52.32	52.43	52.78
LSTM+GCN	53.89	53.51	53.13	53.67	53.31	54.32
Transformer	53.07	53.23	53.73	53.95	53.32	54.47
Transformer+mask	**55.68**	**55.93**	**55.85**	**55.17**	**55.97**	**56.77**

In general, the accuracy of predicting stock performance shown in Table 5.6 is not high enough for the relatively weak characteristics of the stock market. From the results, it is observed that the pure transformer model is 3% higher than the LSTM model, and 1% higher than the pure GCN model.

In the term of the discussion for implementation, we expect that the transformer model would outperform the LSTM+GCN model. However, the prediction accuracy is quite close to the LSTM+GCN model, which is not consistent with our expectation. This may be the complicated market that results in the distraction in the attention mechanism.

The result of the transformer model using a binary stock graph as a mask is about 5.6% higher than the LSTM, and about 2% higher than the LSTM+GCN model. The transformer with masks did exceed the general transformer model by utilizing the financial service datasets. The improvement is not significant but still relative good in general when using masks.

In term of predictability, it is difficult to determine in some sectors are more predictable than others. One observation is that the results on larger datasets are better than the small dataset.The sample size we used for experiment is relatively small compared to the total number of stocks in the market, we can expect a model accuracy better for larger datasets.

Overall, a relatively good prediction was produced by the transformer on the stock data. The result reveals that stock graphs can effectively improve the accuracy and the transformer is a better algorithm for the stock prediction problem.

5.6 Summary

In total, the model utilizing graphs outperforms the model, not including stock graphs. It reveals that the stock graphs can effectively enhance prediction accuracy. We found that larger datasets are more predictable than small sets. The graph-based on correlation coefficient is the best choice among the three types of graph. The result of using a transformer with a mask is about 2% higher than the multi-graph convolutional neural network and 5% higher than the LSTM model. Overall, the transformer with graph mask has the best performance on all the datasets.

6 Conclusion

6.1 Summary

In our research, we focus on the improvement of the stock price prediction by using interactive information between different stocks. We use graphs structure to represents such relation. Besides, we attempt to apply the transformer model for learning the stock relation automatically. We propose a joint GCN model to process multiple graphs to deal with the structured graph data. In our experiments, the results from transformer and multi-graph GCN model are similar to the single graph GCN model. Since the transformer performs similar to the GCN model without using graphs as prior knowledge, we expected the result would be better by providing the model with stock graphs. Hence, we propose the transformer model with graph masks. The outcome of transformer with masks outperforms other models we build in our research. In the experiment, we investigate how various type of graphs influence the prediction result. We find that the graphs from correlation coefficients are better than other constructed graphs.

6.2 Advantages and Limitation

The advantages of our work are that we found the pre-processing possessed a considerable impact on the result. We defined a function for calculating the return price in the methodology section and the result indicates that the return price effectively enhances the loss compared to the min-max normalized price.

From the experimental results, both the multiple graphs GCN and the transformer with the mask model outperformed the LSTM model and the extreme case (assuming all the stock prices increase). It is proved that the transformer is effective in solving stock forecasting problems and the combination with a single graph produced a better result compared to the GCN model.

Our experiment proved that the graphs are effective for stocks prediction problem. In the experiment, we compare the effect of various graphs and found that correlation graphs have a better effect compared to the other two graphs.

There are also some limitations in our work. We list them as the following:

- Due to the complexity of the stock market, the accuracy of predicting the direction of stock prices is generally poor. Besides, the New York Stock Exchange contains different datasets. This may influence the final accuracy. An additional complicating factor is that recent stock performance can be completely different from the former exemplified ones. Our experiment focuses on the datasets from 2013 to 2018 and the market is relatively stable. The model may not have a good prediction, when unexpected shock happens in the world.

- The GCN network is strongly dependent on the graphs. The methods we used to create stock graphs can not provide the GCN with sufficient information. The GCN needs a suitable graph to produce a better result. In the experiment section, we test the GCN model with four different graphs. The results show that the unweighted partially connected graph is the best choice. However, compared to the transformer model, the performance of GCN is highly dependent on the information of graphs . Thus, finding a suitable method for generating stock graphs promises to improve GCN results.

- Although the transformer works relatively well on the stock datasets, the pure transformer model performs similarly to the LSTM+GCN model. The improvement with masks is not very large. The time complexity of GCN is $O(k \cdot n \cdot d^2)$ and the time complexity of transformer is $O(n^2 \cdot d)$, where k is the kernel size, n is the input length and d is the dimension of input elements. When $n < d$, self-attention is faster than the convolution network. However, in our research, the size of stock feature is less than the number of stock, i.e $n > d$. If we consider the time cost, the transformer is not the optimal choice.

- Although the result of transformer with masks outperformed the presented methods, the mask only allows binary graph. According to our displayed graph construction methods, the model cannot process the DTW-based graphs. As a result, the type of input graphs are limited, and we cannot comprehensively analyse the model

6.3 Future Work

- The experiments indicated that the graph neural network is effective in the stock prediction problem. The graphs constructed by correlation and DTW methods are based on the stocks' past performance. Therefore, the information provided by the graphs still focuses on the prices. Moreover, the graph neural network relies on the information provided by the graphs. Hence, for future research, the graph construction method can focus on other information such as shareholding and news rather than price. For the multiple graphs GCN, we provided the graph not comprehensive enough. Future research could concentrate on more different graph construction approaches, and feed more graphs into the model may help enhance accuracy.

- For future experiment, the stock features can be extended. The current investigation only includes historical price and trading volume information. As mentioned in the introduction, behavior finance proposed that the stock movement is influenced by public mood. Thus, the input feature could include information from news and social media.

- We used the undirect graphs indicating that if there exists an edge between two nodes, these two stock has the same impact on each other. Nevertheless, in reality, the impact on stocks is different. One stock can affect the other more but not be affected by the other one. Therefore, for future investigation, we can design the graph construction method to generate indirect graphs to represent the unequal relation between stocks.

- Our transformer uses only one graph as a mask, hence, there is only one graph included in the model. It is worth trying to feed more graphs to the transformer model.

In conclusion, future works can concentrate on feature preprocessing and graphs parts. The feature can be extended to cover more useful information. It means to prolong the field of the information covered by the feature, instead of incrementing the length of historical features. Furthermore, based on the results indicating the effective improvement of the result by the graphs, it is essential to design new graph construction methods and combine graphs and neural networks.

Bibliography

[1] E. F. Fama, "Random walks in stock market prices," *Financial analysts journal*, vol. 51, no. 1, pp. 75–80, 1995.

[2] ——, "The behavior of stock-market prices," *The journal of Business*, vol. 38, no. 1, pp. 34–105, 1965.

[3] ——, "Efficient capital markets: A review of theory and empirical work," *The journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[4] B. G. Malkiel, "The efficient market hypothesis and its critics," *Journal of economic perspectives*, vol. 17, no. 1, pp. 59–82, 2003.

[5] F. Black, "Noise," *The journal of finance*, vol. 41, no. 3, pp. 528–543, 1986.

[6] H. White, "Economic prediction using neural networks: The case of ibm daily stock returns," in *ICNN*, vol. 2, 1988, pp. 451–458.

[7] R. Lawrence, "Using neural networks to forecast stock market prices," *University of Manitoba*, vol. 333, pp. 2006–2013, 1997.

[8] K.-j. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003.

[9] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.

[10] J. Liu, H. Lin, X. Liu, B. Xu, Y. Ren, Y. Diao, and L. Yang, "Transformer-based capsule network for stock movement prediction," in *Proceedings of the First Workshop on Financial Technology and Natural Language Processing*, 2019, pp. 66–73.

[11] A. Mittal and A. Goel, "Stock prediction using twitter sentiment analysis," *Standford University, CS229 (2011 http://cs229. stanford. edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis. pdf)*, vol. 15, 2012.

[12] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[13] L.-C. Cheng, Y.-H. Huang, and M.-E. Wu, "Applied attention-based lstm neural networks in stock prediction," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 4716–4718.

[14] A. M. Rather, A. Agarwal, and V. Sastry, "Recurrent neural network and a hybrid model for prediction of stock returns," *Expert Systems with Applications*, vol. 42, no. 6, pp. 3234–3241, 2015.

[15] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," in *2017 international conference on advances in computing, communications and informatics (icacci)*. IEEE, 2017, pp. 1643–1647.

[16] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[17] H. Fu-Yuan, "Forecasting stock price using a genetic fuzzy neural network," in *2008 International Conference on Computer Science and Information Technology*. IEEE, 2008, pp. 549–552.

[18] Q. Ye, L. Wei *et al.*, "The prediction of stock price based on improved wavelet neural network," *Open Journal of Applied Sciences*, vol. 5, no. 04, p. 115, 2015.

[19] Y. Chen, Z. Wei, and X. Huang, "Incorporating corporation relationship via graph convolutional neural networks for stock price prediction," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1655–1658.

[20] C. Carvalho, N. Klagge, and E. Moench, "The persistent effects of a false news shock," *Journal of Empirical Finance*, vol. 18, no. 4, pp. 597–615, 2011.

[21] D. M. Cutler, J. M. Poterba, and L. H. Summers, "What moves stock prices?" National Bureau of Economic Research, Tech. Rep., 1988.

[22] A. Joulin, A. Lefevre, D. Grunberg, and J.-P. Bouchaud, "Stock price jumps: news and volume play a minor role," *arXiv preprint arXiv:0803.1769*, 2008.

[23] J. Y. Campbell, S. J. Grossman, and J. Wang, "Trading volume and serial correlation in stock returns," *The Quarterly Journal of Economics*, vol. 108, no. 4, pp. 905–939, 1993.

[24] K. T. Chi, J. Liu, and F. C. Lau, "A network perspective of the stock market," *Journal of Empirical Finance*, vol. 17, no. 4, pp. 659–667, 2010.

[25] C. K. Tse, J. Liu, F. C. M. Lau, and K. He, "Observing stock market fluctuation in networks of stocks," in *Complex Sciences*, J. Zhou, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 2099–2108.

[26] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[27] M. MÃŒller, *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. Dynamic Time Warping, pp. 69–84. [Online]. Available: https://doi.org/10.1007/978-3-540-74048-3_4

[28] M. E. Newman, "Analysis of weighted networks," *Physical review E*, vol. 70, no. 5, p. 056131, 2004.

[29] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[30] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[31] S. Ryu, J. Lim, S. H. Hong, and W. Y. Kim, "Deeply learning molecular structure-property relationships using attention- and gate-augmented graph convolutional network," *arXiv preprint arXiv:1805.10988*, 2018.

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[33] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[34] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[36] K. Chen, Y. Zhou, and F. Dai, "A lstm-based method for stock returns prediction: A case study of china stock market," in *2015 IEEE international conference on big data (big data)*. IEEE, 2015, pp. 2823–2824.

[37] A. E. Chambers and S. H. Penman, "Timeliness of reporting and the stock price reaction to earnings announcements," *Journal of accounting research*, pp. 21–47, 1984.

[38] K. Baba, R. Shibata, and M. Sibuya, "Partial correlation and conditional correlation as measures of conditional independence," *Australian & New Zealand Journal of Statistics*, vol. 46, no. 4, pp. 657–664, 2004.

[39] D.-M. Song, M. Tumminello, W.-X. Zhou, and R. N. Mantegna, "Evolution of worldwide stock markets, correlation structure, and correlation-based graphs," *Physical Review E*, vol. 84, no. 2, p. 026108, 2011.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[41] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.

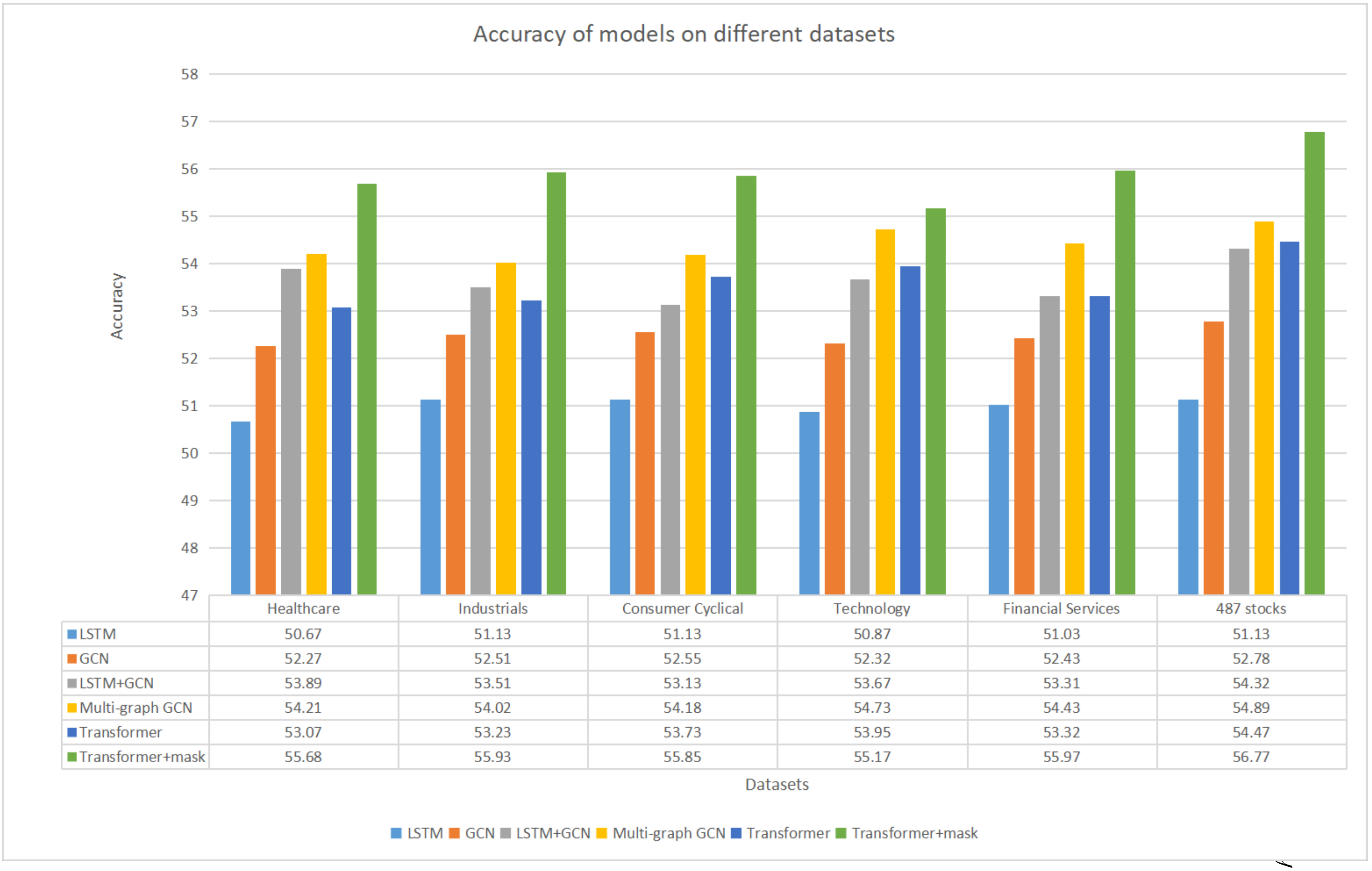
[42] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, and H. Ney, "A comparison of transformer and lstm encoder decoder models for asr," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 8–15.

[43] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *arXiv preprint arXiv:1905.09418*, 2019.

[44] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[45] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, NJ, 1996, vol. 2.

Appendix



	Healthcare	Industrials	Consumer Cyclical	Technology	Financial Services	487 stocks
LSTM	50.67	51.13	51.13	50.87	51.03	51.13
GCN	52.27	52.51	52.55	52.32	52.43	52.78
LSTM+GCN	53.89	53.51	53.13	53.67	53.31	54.32
Multi-graph GCN	54.21	54.02	54.18	54.73	54.43	54.89
Transformer	53.07	53.23	53.73	53.95	53.32	54.47
Transformer+mask	55.68	55.93	55.85	55.17	55.97	56.77

Figure .1: Histogram of accuracy of different models on different datasets