Final Report: Progress Report

COMP 4801 Final Year Project

Project Title: Financial Market Prediction by Deep Learning Neural Network

Student: 3035183552 ZHANG Yuqing

Supervisor: Kwok-Ping Chan

Date of Submission: 15/04/2018

Abstract

Deep learning, as a method of machine learning based on data representation learning, is used widely in the field of financial variables forecasting in recent years. This project taking DJIA as an example, explores the usage of deep learning and CNN in forecasting stock indices. This report elaborates the background and process of the project. The project is significant because it explores a new application of CNN and has practical value in financial markets.

Table of Contents

List of Figures

List of Tables

Abbreviations

CNN	Convolutional Neural Network
MLP	Multi-layer Perceptron
GDP	Gross Domestic Product
DJIA	Dow Jones Industrial Average
WSJ	Wall Street Journal

1. Introduction

Deep learning networks have been very popular in machine learning. In this project, I will use deep learning and convolutional neural network to build a model. The model could be used to predict the future stock index, which can also be applied in many other fields.

The introduction firstly introduces the background knowledge which is deep learning and convolutional neural networks. Then the Dow Jones Industrial Average, which is the study target for this project, is included. The literature review of this project is also introduced in the introduction. Based on the relative study, my motivations and objective of this report are also mentioned in this part. Finally, the introduction part includes the scope of this project. The content which is included and not included of this project would be clarified.

1.1 Background knowledge

At present stage, through all the research and investigation, it is found that deep learning is a branch of neural network, which can be assumed as one of the direction of development of neural network *(Taghi, 2013).* In this project, the concept and applications of deep learning and convolutional neural network are significant for building the model and making prediction.

1.1.1 Deep learning

Deep learning is a new field of the study of machine learning, of which motivation is to establish and simulate the neural network of people's brain which is used to study and analyze. It imitates human's brain to interpret the data, such as images, sound and text.

The concept of deep learning stems from the study of artificial neural networks. For example, a multi-layer perceptron with hidden layers is a structure of deep learning. Deep learning combines low-level features to form more abstract high-level representations, attributes, categories, or features, to discover distributed representations of data.

There are multiple layers in a deep learning neural network. For each layer, features are extracted, abstracted and transformed to the next layer. A simple structure of deep learning can be showed as Figure1, the input values are sent into the input layer, whose feature is generated and sent to the next layer. Then the data in this layer also would be identified and sent to the next layer. This process will continue until the neural network proceeds the output layer.

The application of deep learning is broad which covers multiple fields. For example, deep learning is applied in speech recognition acoustic model training making speech recognition error rate reduced by 30% *(Taghi, 2013)*. Apart from that, deep learning also performed ideally in computer vision, natural language processing, audio recognition and so on.

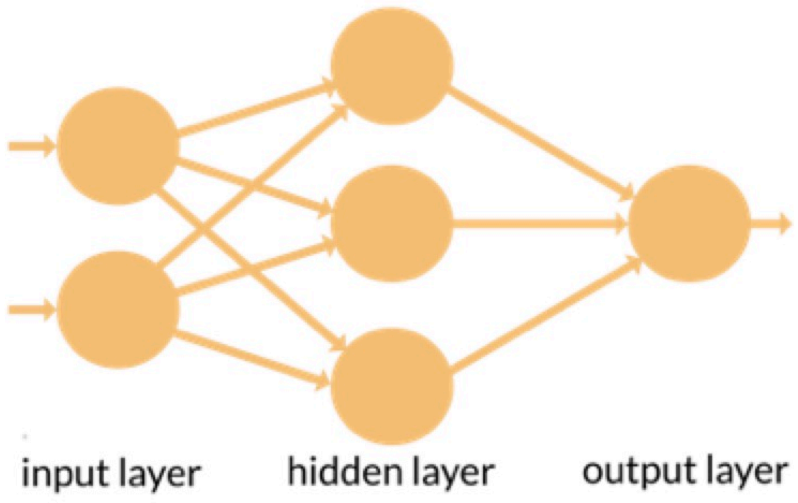


Figure1: A two-layer neural network

1.1.2 Convolutional Neural Network

Convolutional Neural Networks (CNN) are biologically-inspired variants of MLPs, is one of the artificial neural networks.

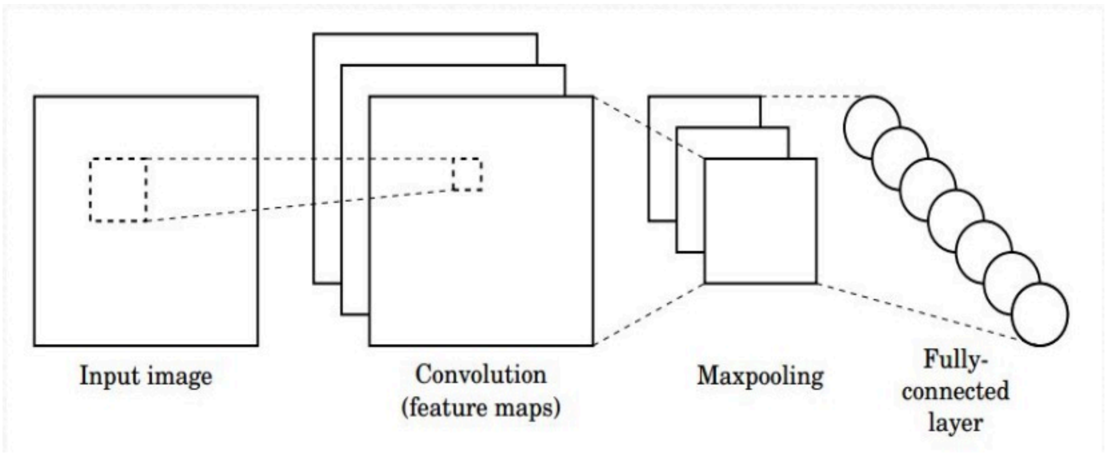


Figure 2: A CNN with three types of layers.

As shown in Figure 2, there are three types of layers in convolutional neural networks, which are convolutional layer, pooling layer and fully-connected layer. During the step of data modeling, the number of layers and the parameters for each layer are decided.

In CNNs, each filter is replicated across the entire visual field. These replicated units share the same parameterization and form a feature map, which means we can simplify the model by reducing the number of parameters. To form a richer representation of the data, each hidden layer is composed of multiple feature maps.

The training process of CNN can be divided as four steps, two phases.

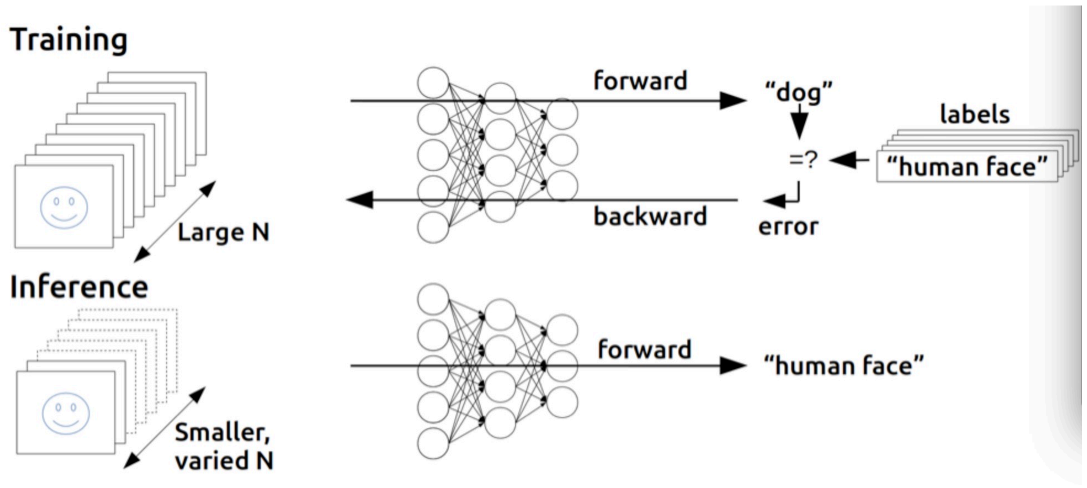


Figure 3: Forward and Backward propagation in CNN

The first phase is forward propagation, of which process is shown in Figure 3.

Step 1: Take a sample (X, Yp) from the sample collection and import the X into the network;

Step 2: Calculate the actual output Op;

During this phase, the information is transmitted progressively from the input layer to the output layer, and the work for network is calculation.

The second phase is backward propagation:

Step 3: Calculates the difference between the actual output Op and the corresponding ideal output Yp;

Step 4: Using the minimum error method, adjust the backward propagation adjustment weight matrix.

Figure 4 is an application of CNN, which shows the input values and the pooling layer with 2*2 filters and stride 2. During this project, the application of CNN is similar to the structure shown in Figure 5.

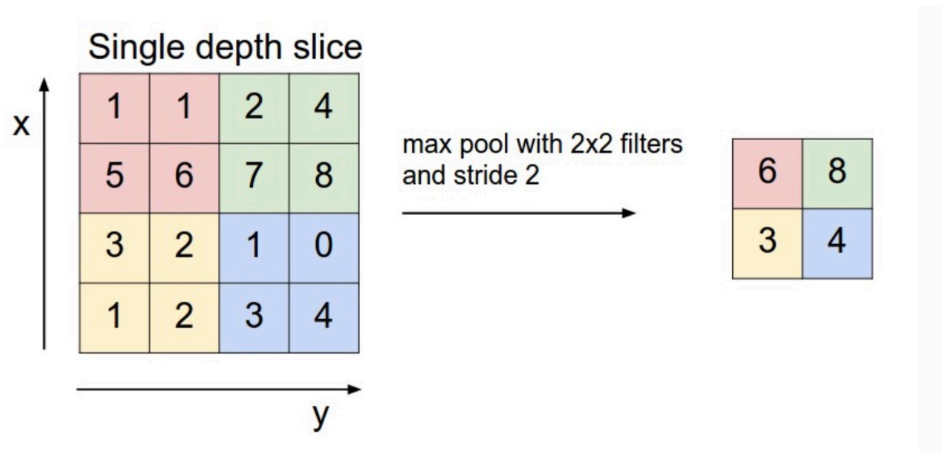


Figure 4: An example of convolutional neural network application

1.2 Literature Review

This project is inspired by the study of ImageNet classification, which divides 1 million 200 thousand high-definition images in ImageNet Large-Scale Visual Recognition Challenge-2010 contest into 1000 different categories by training a large-scale deep convolutional neural network and achieves 17% top five error rate *(Krizhevsky, 2012).*

The architecture of the study of ImageNet classification was mainly divided into convolutional layers and fully-connected layers as showed in Figure 5. Although the data type was studied in my project is different from the data type used in the ImageNet classification, the ideas of feature induction in these two projects are similar. Figure 6 shows the road map of the data transformation between adjacent layers of these two projects. In the convolutional neural network of my project, layer (m-1) transfers data to layer m and layer m induces features and transfer them to the next layer.

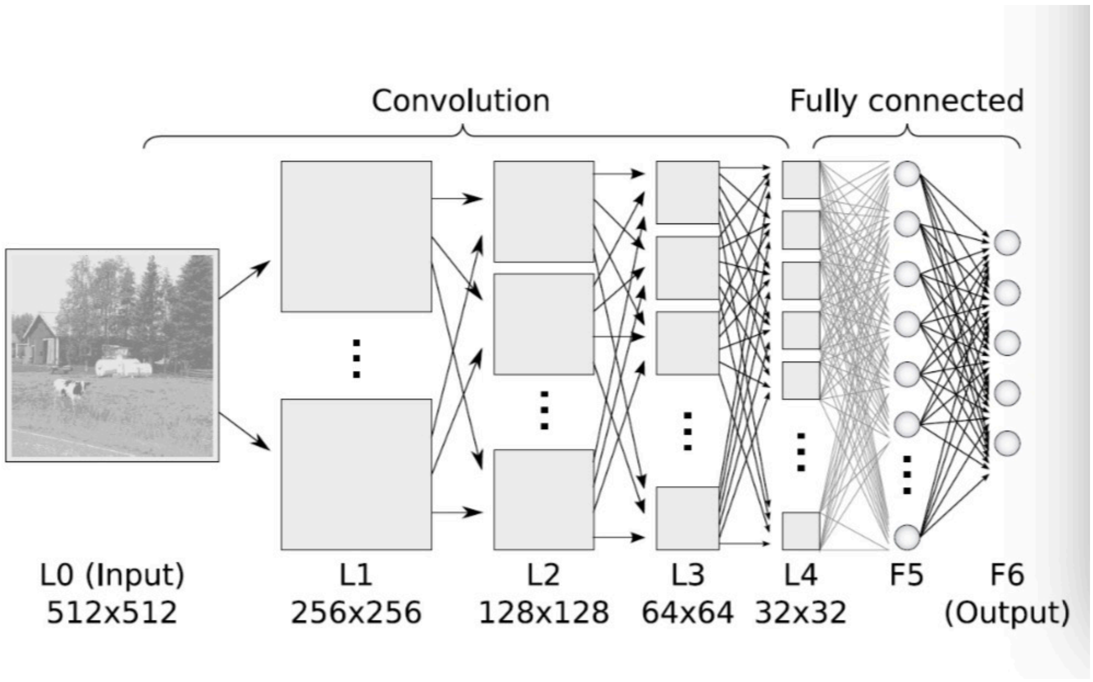


Figure 5: Architecture of convolutional neural network applied in ImageNet

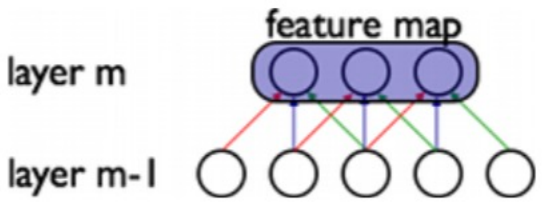


Figure 6: Feature transformation between two adjacent layers

In the study of ImageNet classification using CNN, RuLU functions are applied for activation function in the neural network. It is found by *Krizhevsky et al* that the convergence rate of SGD obtained by ReLU is much faster than that of sigmoid/tanh *(Krizhevsky, 2012)*. Figure 7 shows that it is six times faster for a convolutional neural network with ReLUs which is in solid line than the same convolutional neural network with tanh neurons which is in dashed line for reaching a 25% training error rate.

In this project, several different activation functions are compared. Considering the advantages of ReLU activation function discussed in Krizhevsky's paper, I applied ReLU as activation function in the neural network.

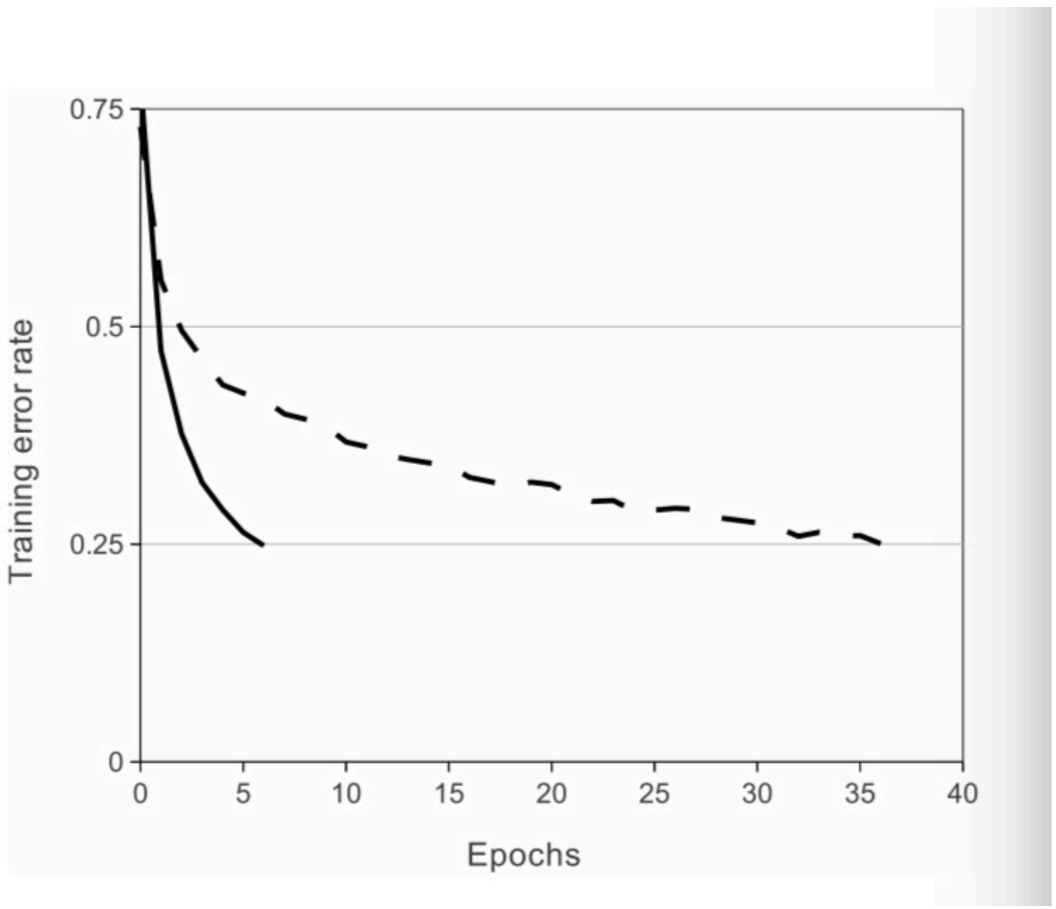


Figure 7: Comparison between ReLU and tanh in same convolutional neural network to reach a 25% training error rate

1.3 Dow Jones Industrial Average (DJIA)

The Dow Jones Industrial Average is stock market indices created by the Wall Street Journal Charles Doug, which was firstly launched in May of 1896. This index is used as a measure of the development of industrial composition in the U.S. stock market, is the oldest American market index. Because of the effect of compensating stock split and other adjustments, it is only a weighted average and does not represent the average of the value of the constituent stock.

To be chosen as a dataset for study, the Dow Jones Industrial Average includes the index of 30 largest and most famous listed companies in the United States. Considering the reliability and representativeness of the study result of the project, data of DJIA is used for its extensive coverage of the stocks of different industrials. Figure 8 shows the sector breakdown of DJIA, from which the various fields of stocks covered by DJIA can be observed. Additionally, the time range of DJIA is suitable for a long-term study. For the sake of the accuracy of the study, the stock data of at least thirty years should be collected. DJIA is mature enough to fulfill the requirements of the study.

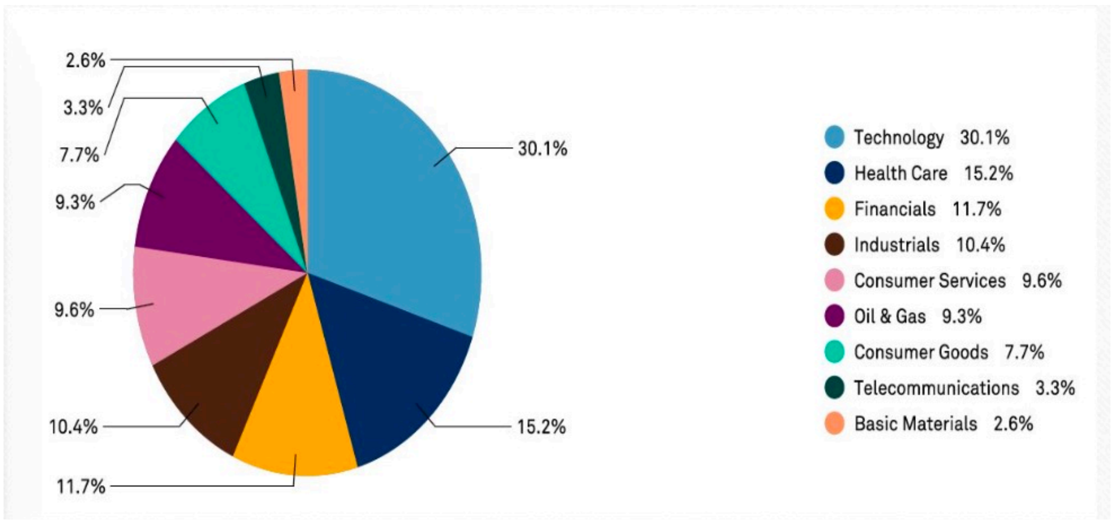


Figure 8: Sector breakdown of DJIA

During the preliminary preparation and research of the project, there are some other stocks and markets under consideration. For example, the data of S&P 500 index was also taken into consideration. However, after implementing relative research, it was found that there were already substantial similar studies based on the data of S&P 500 index. After comparison and research, DJIA was chosen for the sake of novelty.

1.4 Objective

Because the volume of trading in stock market is continuously increasing, forecasting the future financial variables based on the previous data can make significant profit. Even though the efficient market hypothesis states that forecasting the future market price is impossible *(Peter, 1991),* there are no efficient evidences to support this statement. Additionally, related study shows that using time series makes it possible to forecast future financial variables *(Taghi, 2013).*

Deep learning becomes a new model which is outputting the result by training data in recent years, and it is used widely in the field of predicting financial variables. In addition, CNN is the artificial neural networks which generating the features of data by self-learning based on the database. As mentioned in 1.1.2, the usage of CNN reduces the parameters needed when building the models.

With the thought of deep learning, the objective of this project is to develop a CNN model to predict the trend of financial variables. DJIA indices are used as study objective in this project. This project intends to predict the trend of DJIA of the fourth business days, based on the trading data of previous 3 business days.

In this project, whether the value of indices will rise or drop is considered and evaluated. The result of predicted trend should with a satisfactory accuracy, which is at least greater than 0.5.

Apart from that, an analysis about the performance and the limitation of the model are conducted. During this stage, the range of applications of the model is analyzed. Considering the situation of the implementation, if the accuracy and processing time are both satisfactory, relative applications will be suggested. For example, with corresponding parameters, the model can be used to predict weekly/monthly/yearly inflation rate based on previous performance and related financial variables. This application forecasts inflation and reduce economic loss caused by economic bubble. Short-term trades in stocks and futures will also be implemented based on the result of this project.

1.5 Scope

The Dow Jones Industrial Average in the past thirty years (1st Oct, 1998 ~ 30th Sept, 2017) is the study objective in this report. For the sake of the accuracy, some other financial indexes such as GDP and interest rate are also included. Because the study objective is the U.S. market index, the financial indexes of other area is not included in our project.

In this project, only the idea of CNN will be applied to build the model rather than other neural networks. In this project, it does not consider the parameters and factors which CNN does not included. As mentioned in 1.1.2, CNN requires less parameters to build the neural networks. Apart from that, this project will explore applications of CNN in financial predictions considering its satisfactory performance in image recognition.

In this project, the open source programming language, Python is applied as technological support. Other programming languages will not be used. Python is used widely in the field of deep learning. The mainly reason is that there are abundant practical libraries available which can be applied to complete complicated algorithms and applications. In this project, Tensorflow is used in building convolutional neural network model. Additionally, anyone can create a Python package and submit it to PyPI (Python package index), which makes it easier and more convenient to fulfill the requirements of this project.

2. Methodology

The objective of this project is to forecast whether the index of one particular stock will rise up or fall down based on the past trading data.

The nuclear of this process is data, which calls for collection, preprocessing, modeling and training according to the experimental sets. Apart from that, the assessment of the CNN model evaluates the performance of the model. Certain technologies and tools are applied for preprocessing the data and testing the model I build. The following part explains the methodology of this project for each step.

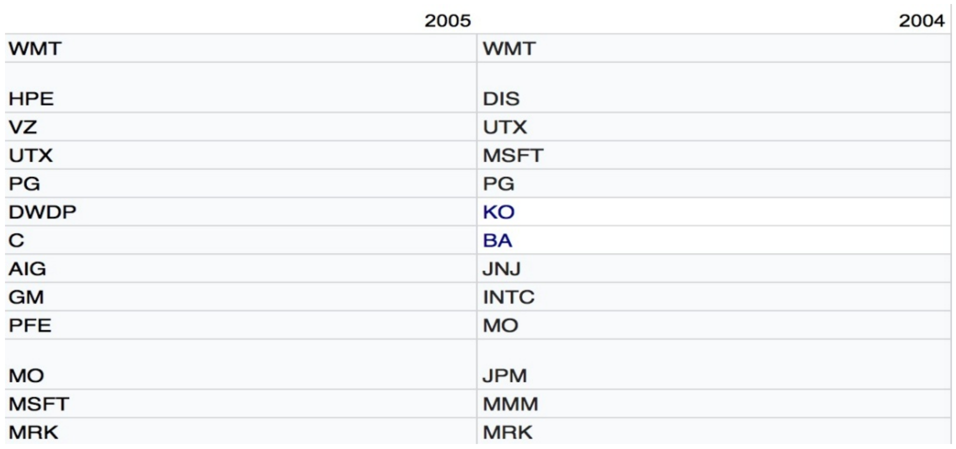
2.1 Data collection

There are various sources for collecting the previous stock index. Yahoo Finance and Wall Street Journal are the main sources in this project.

Considering the expected accuracy of the model, I decided to take the stock index of past three decades as the sample of this project. The closing price of the trading market is referred as the index of each day.

In this project, 11 variables in total are collected for reference.

Firstly, during the preliminary research, it is found that the constituents of DJIA had been changed multiple times during the past thirty years. Under this situation, the constituents of DJIA for each year are collected to build a table as shown in Table 1, which is a part of the constituents of DJIA from 2004 to 2005. To simplify the process of recording, the stock codes are used instead of company names. After collecting the constituents of DJIA, the DJIA for each company each year for the past 30 years is collected for data training.

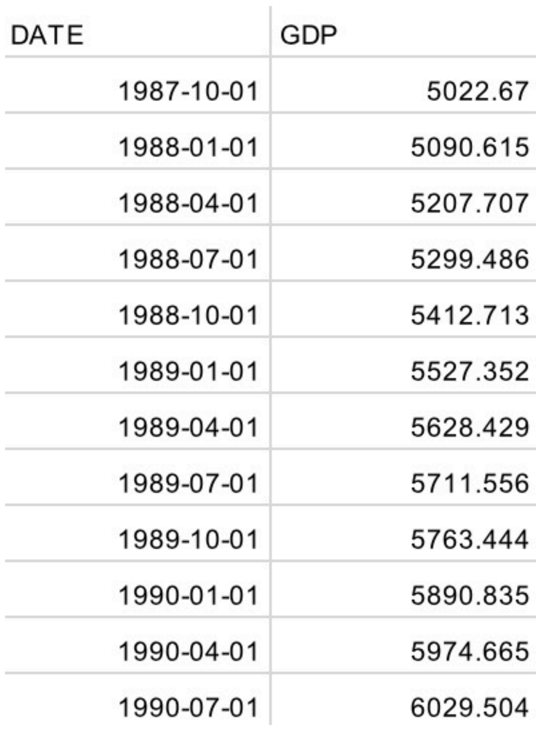


2005	2004
WMT	WMT
HPE	DIS
VZ	UTX
UTX	MSFT
PG	PG
DWDP	KO
C	BA
AIG	JNJ
GM	INTC
PFE	MO
MO	JPM
MSFT	MMM
MRK	MRK

Table 1: Part of constituents of DJIA from 2004 to 2005

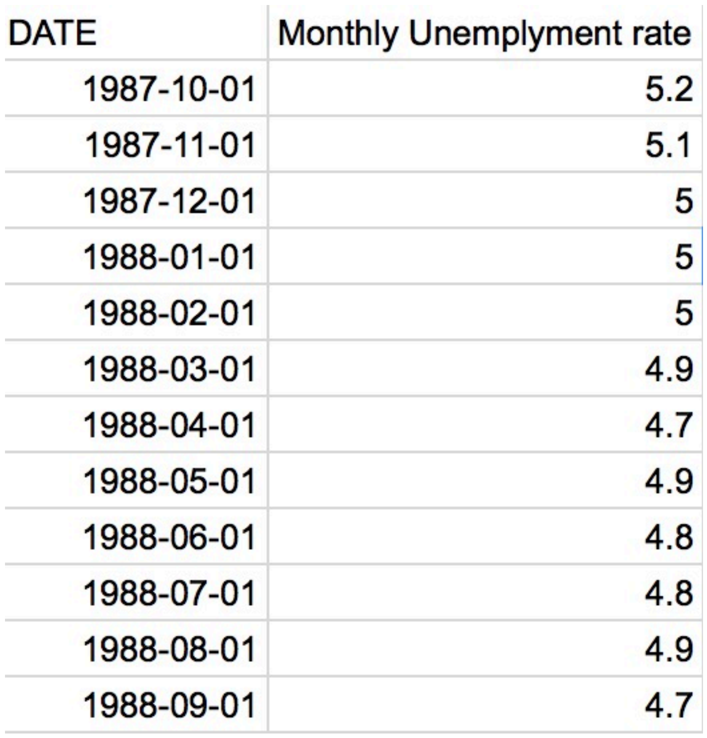
Secondly, for each stock, six daily varieties are collected, which are open value, close value, high value, low value, adjusted close value and volume. A table with the variables and corresponding date for each company is built for reference.

Thirdly, this project also takes some macro financial indexes in consideration for the integrity of the model. Four variables are collected in total, which are monthly inflation rate, monthly unemployment rate, monthly interest rate and quarterly GDP. The value with corresponding date are collected in tables. Table 2 and Table 3 take the quarterly GDP and monthly unemployment rate for examples, show the table of macro indexes collected.



DATE	GDP
1987-10-01	5022.67
1988-01-01	5090.615
1988-04-01	5207.707
1988-07-01	5299.486
1988-10-01	5412.713
1989-01-01	5527.352
1989-04-01	5628.429
1989-07-01	5711.556
1989-10-01	5763.444
1990-01-01	5890.835
1990-04-01	5974.665
1990-07-01	6029.504

Table 2: Quarterly GDP from 1st, Oct in 1987 to 1st, July in 1990



DATE	Monthly Unemplyment rate
1987-10-01	5.2
1987-11-01	5.1
1987-12-01	5
1988-01-01	5
1988-02-01	5
1988-03-01	4.9
1988-04-01	4.7
1988-05-01	4.9
1988-06-01	4.8
1988-07-01	4.8
1988-08-01	4.9
1988-09-01	4.7

Table 3: Monthly unemployment rate from 1st, Oct in 1987 to 1st, Sep in 1988

2.2 Data preprocessing

During the process of collecting the data, we found that the format of the data was not standard. Besides, there were circumstances of data missing and suspension.

Firstly, the missing values and suspension are considered. The markets may be closed during holidays and weekends, some other factors such as major accidents and bad weather also lead to trade suspension. In addition, some companies may be absent from trading on certain days. In view of these situation, I decide to consider the missing index as its last valid index.

Secondly, the member of DJIA changed during the past thirty years. As mentioned in 2.1, a table of change of constituents of DJIA has been made. I process the data of these companies separately and use this data as testing sets.

Thirdly, the standardization of the data should be processed. On the one hand, the initial constituents of DJIA I collected were the name of each company, which cannot be processed easily. In view of this situation, the name of each company was transferred into the corresponding stock code. On the other hand, because the data was collected from different sources, the units of data were not unified. Standardization of the data is processed to unify the units and formats for easier accessibility.

In this project, I created a python program to generate a .cvs file contains a table which displays all the data used for model training. There are 13 volumes in this table, which are stock code, data, and the corresponding open value, high value, low value, close value, adjusted value, volume, quarterly GDP, monthly unemployment rate, monthly inflation rate, monthly interest rate, DJIA. Table 4 shows an example which is a part of the table which contains all the raw data before data standardization.

Name	Date	Open	High	Low	Close	Adj	Volume	GDP	Unemployment	Inflation	Interest	Djia
AAPL	1987-09-30	1.9375	2.035714	1.9375	2.017857	1.661718	30520000	19495.476	3.9	114.7	7.22	2581.919922
AAPL	1987-10-01	2.026786	2.098214	2.017857	2.080357	1.713188	29120000	5022.67	5.2	114.7	7.22	2581.919922
AAPL	1987-10-02	2.080357	2.098214	2.053571	2.089286	1.720541	24124800	5022.67	5.2	115	7.29	2601.5
AAPL	1987-10-05	2.089286	2.133929	2.0625	2.116071	1.742599	33600000	5022.67	5.2	115	7.29	2641.159912
AAPL	1987-10-06	2.125	2.125	1.982143	1.991071	1.63966	50400000	5022.67	5.2	115	7.29	2646.540039
AAPL	1987-10-07	1.982143	1.991071	1.9375	1.982143	1.632307	56000000	5022.67	5.2	115	7.29	2623.530029
AAPL	1987-10-08	1.982143	2	1.901786	1.9375	1.595544	41160000	5022.67	5.2	115	7.29	2549.449951
AAPL	1987-10-09	1.9375	1.982143	1.928571	1.933036	1.591867	36400000	5022.67	5.2	115	7.29	2555.320068
AAPL	1987-10-12	1.9375	1.941964	1.848214	1.901786	1.566133	49840000	5022.67	5.2	115	7.29	2508.159912
AAPL	1987-10-13	1.946429	1.955357	1.901786	1.946429	1.602897	40600000	5022.67	5.2	115	7.29	2479.439941
AAPL	1987-10-14	1.919643	1.928571	1.857143	1.901786	1.566133	64680000	5022.67	5.2	115	7.29	2499.350098
AAPL	1987-10-15	1.901786	1.946429	1.848214	1.857143	1.529369	87080000	5022.67	5.2	115	7.29	2473.560059
AAPL	1987-10-16	1.866071	1.892857	1.696429	1.723214	1.419078	105000000	5022.67	5.2	115	7.29	2394.419922
AAPL	1987-10-19	1.723214	1.723214	1.267857	1.303571	1.0735	119000000	5022.67	5.2	115	7.29	2363.899902
AAPL	1987-10-20	1.375	1.5	1.165179	1.232143	1.014677	142240000	5022.67	5.2	115	7.29	2046.670044

Table 4: Raw data from Sep in 1987 to 20th, Oct in 1987

After generating the table for raw data, the standardization of the data is processed. The method for data standardization is replacing the raw data with its z-score. The

$$x_{\text{scaled}} = \frac{x - \mu(x)}{stdev(x)}$$

formula for zero-mean normalization is showed in Figure 9.

Figure 9: Zero-mean normalization formula used in data standardization

After the process of data standardization, a table which contains 13 volumes

Name	Date	Open	High	Low	Close	Adj	Volume	GDP	Unemployment	Inflation	Interest	Djia
AAPL	1987-09-30	-0.1839696912981366	-0.18403117305636113	-0.18382990030375362	-0.18389537565241948	-0.17993717652454022	0.33784448914867804	0.1647847241054285	-0.13145018797867492	-9.677266783106031	9.349126039096543	-7.304325663970723
AAPL	1987-10-01	-0.18392237782255344	-0.18399812451939473	-0.18378722438976264	-0.18386226164559746	-0.17990992382720258	0.2990706697928161	-8.718083100623994	3.3403253005480917	-9.677266783106031	9.349126039096543	-7.304325663970723
AAPL	1987-10-02	-0.18389399005514911	-0.18399812451939473	-0.18376825743489533	-0.18385753084612683	-0.17990603050908477	0.16072568233110082	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.2980856399987175
AAPL	1987-10-05	-0.1838892584956271	-0.18397923926343063	-0.18376351543063899	-0.18384333950736323	-0.1798943510842184	0.4231468917315742	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.285446323566596
AAPL	1987-10-06	-0.18387033331735753	-0.1839839607096158	-0.18380619134462997	-0.1839095675210072	-0.1799488559494066	0.8884327240019171	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.283731717452543
AAPL	1987-10-07	-0.18394603456034503	-0.1840547792297338	-0.18382990030375362	-0.1839142977906537	-0.1799527492675244	1.0435280014253647	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.29106483502215
AAPL	1987-10-08	-0.18394603456034503	-0.1840500577835486	-0.18384886725862096	-0.18393795072835858	-0.17997221479913939	0.6325255162532285	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.3146736001439265
AAPL	1987-10-09	-0.1839696912981366	-0.18405950014714237	-0.18383464230800997	-0.18394031586318182	-0.1799741617229418	0.500694530443298	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.312802837892699
AAPL	1987-10-12	-0.1839696912981366	-0.18408074586181075	-0.18387731822200096	-0.18395687286659282	-0.1799877875421236	0.8729231962595724	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.327832425821997
AAPL	1987-10-13	-0.18396495973861457	-0.1840736639569213	-0.18384886725862096	-0.18393321992888795	-0.1799683214810216	0.6170159885108838	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.336985265127096
AAPL	1987-10-14	-0.18397915388727137	-0.1840878277667002	-0.1838725762177446	-0.18395687286659282	-0.1799877875421236	1.2839256814317086	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.330640047453991
AAPL	1987-10-15	-0.18398861647640613	-0.18407838487432984	-0.18387731822200096	-0.18398052580429772	-0.18000725360322561	1.904306791125499	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.338859139410568
AAPL	1987-10-16	-0.18400754218458495	-0.1841067124938877	-0.18395792804572658	-0.18405148461741233	-0.1800656512570446	2.4006116788805314	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.364080507375895
AAPL	1987-10-19	-0.18408324342757243	-0.18419641574119316	-0.1841855336284505	-0.18427382159604921	-0.18024863032525015	2.7883498724391504	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.373807008793615
AAPL	1987-10-20	-0.18426776524047375	-0.18431444607927985	-0.18424006375646382	-0.18431166587251768	-0.1802797763407056	3.431995273746458	-8.718083100623994	3.3403253005480917	-9.65512216858336	9.46305021630992	-7.4749057847308675

mentioned before and the values are the corresponding z-score of values is generated. Table 5 shows an example which is a part of the table after data preprocessing as reference.

Table 5: Training data after standardization from Sep in 1987 to 20th, Oct in 1987

2.3 Experimental sets

In the range of the study period, there are 7800 days approximately. I use 3 days as one sample, which means there are 7798 samples ideally. The data are divided into training data, validation data and testing data. In this project, 70% of the data are used for training and 30% of the data are used for validation and testing.

The goal of this project is to predict whether the stock price would rise or fall with days for the unit. So, a variable Y is introduced to represent the situation of financial variables. I set Y equals to 1 if the closing price of the last day of a week is higher than the first day, otherwise Y equals to 0. As shown in 2.1, a table with significant daily variables for each stock has been made. During this step, another table which contains the labels for each day is built, which shows the situation of increase and decrease for each stock.

2.4 Data modeling

After building the datasets table, a model is built to train the data and generate the features of the data. During this step, multiple parameters are determined. For the structure of the model, the number of layers, the size of the filter and the number of hidden units of the fully-connected layers are determined. For each layer of the model, the type and activation function are determined.

As shown in Figure 10, the convolutional neural network is mainly composed by two parts, which are convolutional layers and fully-connected layer. This part discusses the process of generation and the structure for each layer.

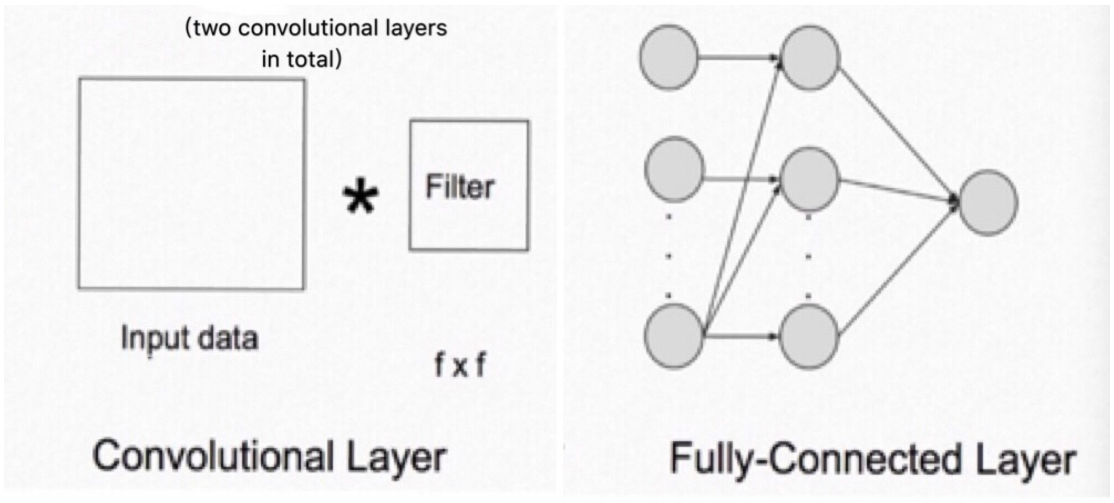


Figure 10: Process of determining each layer of CNN for each sample

2.4.1 Input layer

The parameters for the input layer are decided. In this project, I use MEGD algorithm, which is normally used in CNN models for building neural network. For the fact that I generate a one-dimension table using all the data, so the input layer is one-dimension. For the data type, I use float32, which is widely used in CNN models. The height and width for input layer are 3 and 11 correspondingly, the number of channels for input layer is 1.

2.4.2 Convolutional layers

Theoretically, increasing the number of convolutional layers improves the accuracy of the data training. However, the size of data of this project is not huge. Considering the pursuit of both the accuracy and efficiency, a convolutional neural network with two convolutional layers is constructed.

In the first convolutional layer, the size of the filter is decided as 3*3. Because the input layer is a one-dimension network, the depth of the filter is 1 as well. The number of feature maps outputted by the first convolutional layer is decided by 10. In the second convolutional layer, the size of filler is also 3*3. The depth of the filter is 10 which is decided by the output of the first convolutional layer and the feature maps outputted by the second convolutional layer is decided by 5.

In the construction of convolutional layers, batch normalization algorithm is applied. The principle of batch normalization is to normalize the corresponding activation by mini-batch, which ensures the mean value for each output signal dimension to be zero and the variance to be 1. So, batch normalization improves the local response normalization and decrease gradient dispersion by standardizing the mean and variance are standardized to be consistent.

After determining the layers and parameters in convolutional layers, the activation function is speculated. By inference, several possible activation functions are made, and the one with best performance and highest accuracy is decided as activation function in the neural network model. During this step, I take three possible activation functions into consideration.

The model will be firstly considered using Sigmoid functions as activation functions. Sigmoid is a commonly used nonlinear activation function, with the mathematical form as followed: $f(x) = \frac{1}{1 + e^{-x}}$

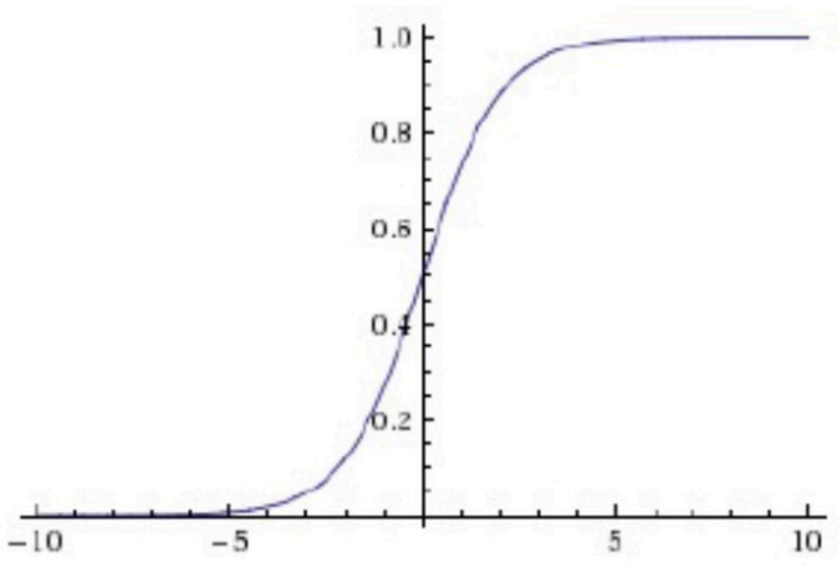


Figure 11: Function curve diagram of Sigmoid function

Sigmoid is chosen as possible activation function because it has the function codomain of [0,1], which can be observed from Figure 11. With this feature, Sigmoid can squashes input numbers into range between [0,1]. Based on preliminary research, when the output value of the activation function is limited into a relatively small range, the representation of the features is more significant by the influence of the finite weight value.

Another possible activation function for the CNN model is Tanh function, with the mathematical form as followed: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

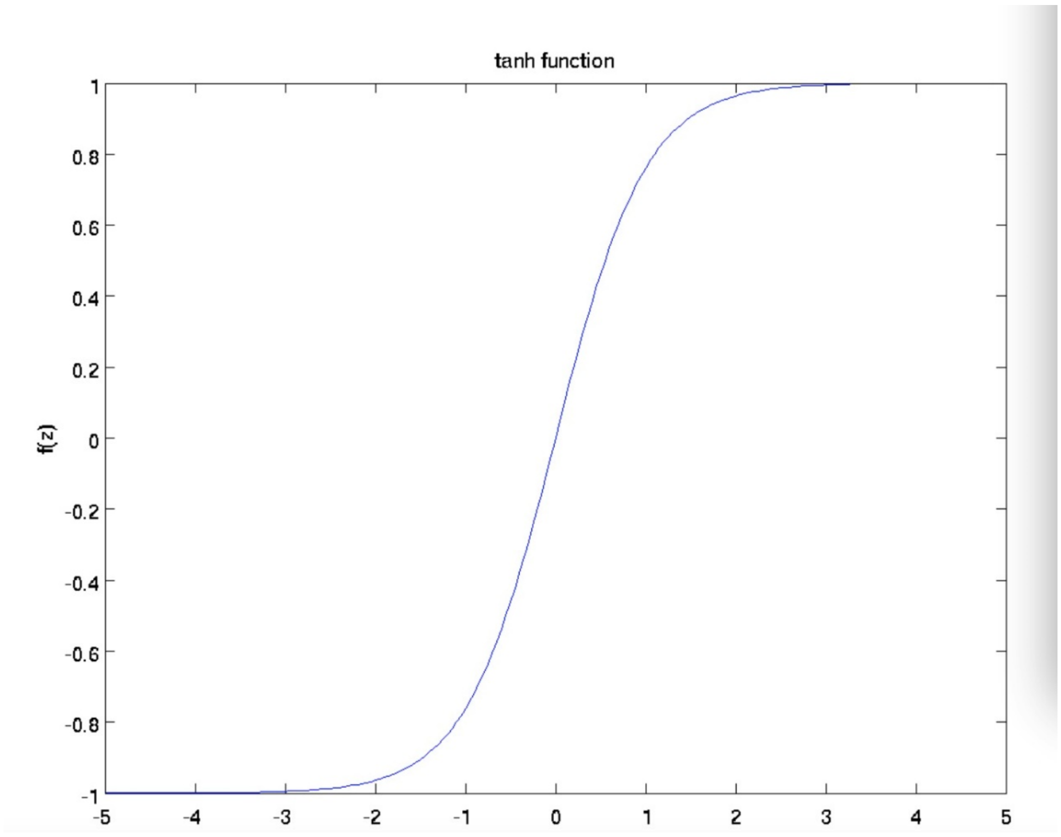


Figure 12: Function curve diagram of Tanh function

Tanh functions are actually the deformation of Sigmoid function with the transforming formula: *tanh(x) = 2sigmoid(2x) – 1*. However, the output of Tanh function has the average value of zero, which can be observed from Figure 12. Because of this feature, the output of each layer of Tanh functions are more stable than Sigmoid functions for the fact that the expectation of output will not change during the process of data modeling.

ReLU functions are also under consideration. ReLU functions are piecewise linear functions, which change all negative values to 0, while positive values remain unchanged, as showed in Figure 13:

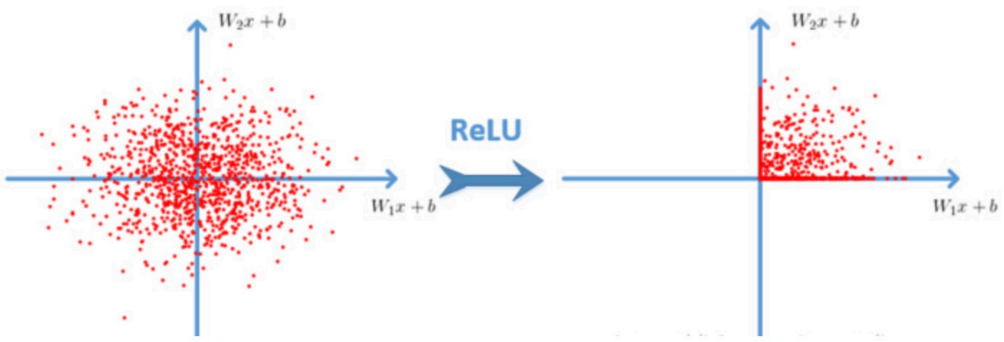
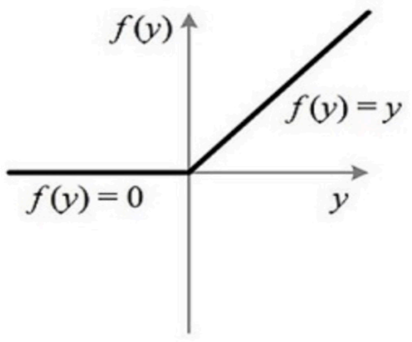


Figure 13: Point distribution after applying a ReLU function

The operation of ReLU functions is unilateral inhibition, which makes neurons in the neural network also have sparse activation. In the convolutional neural network, after N layers are added, the activation rate of ReLU neurons decreases by 2^N times (Liu, 2016). The function curve and mathematical formula for ReLU functions are showed in Figure 14:

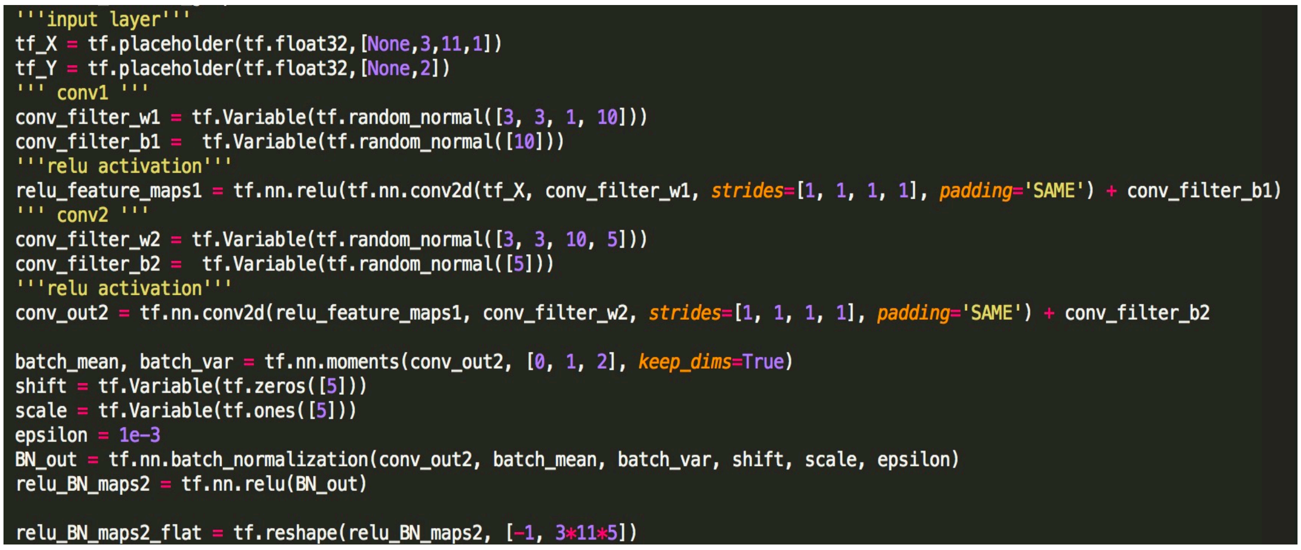
Figure 14: Function curve and mathematical formula for ReLU functions



$$ReLU(x) = \begin{cases} x & if x > 0 \\ 0 & if x \leq 0 \end{cases}$$

As mentioned in 1.2, the neural networks using the ReLU functions obtain a much faster convergence rate than that of sigmoid functions or tanh functions *(Krizhevsky et al.)*. Additionally, ReLU functions have stronger expression ability for linear functions. Comparing with sigmoid/tanh functions, because the gradient of the non-negative interval of ReLU functions is constant (equals to 0), there is no vanishing gradient problem. So, the convergence rate of the CNN model is maintained in a stable state. Considering the advantages of ReLU functions, it is applied as activation function in convolutional layers of the CNN model in this project.

Figure 15 shows the code of convolutional layers for reference.



```
'''input layer'''
tf_X = tf.placeholder(tf.float32,[None,3,11,1])
tf_Y = tf.placeholder(tf.float32,[None,2])
''' conv1 '''
conv_filter_w1 = tf.Variable(tf.random_normal([3, 3, 1, 10]))
conv_filter_b1 =  tf.Variable(tf.random_normal([10]))
'''relu activation'''
relu_feature_maps1 = tf.nn.relu(tf.nn.conv2d(tf_X, conv_filter_w1, strides=[1, 1, 1, 1], padding='SAME') + conv_filter_b1)
''' conv2 '''
conv_filter_w2 = tf.Variable(tf.random_normal([3, 3, 10, 5]))
conv_filter_b2 =  tf.Variable(tf.random_normal([5]))
'''relu activation'''
conv_out2 = tf.nn.conv2d(relu_feature_maps1, conv_filter_w2, strides=[1, 1, 1, 1], padding='SAME') + conv_filter_b2

batch_mean, batch_var = tf.nn.moments(conv_out2, [0, 1, 2], keep_dims=True)
shift = tf.Variable(tf.zeros([5]))
scale = tf.Variable(tf.ones([5]))
epsilon = 1e-3
BN_out = tf.nn.batch_normalization(conv_out2, batch_mean, batch_var, shift, scale, epsilon)
relu_BN_maps2 = tf.nn.relu(BN_out)

relu_BN_maps2_flat = tf.reshape(relu_BN_maps2, [-1, 3*11*5])
```

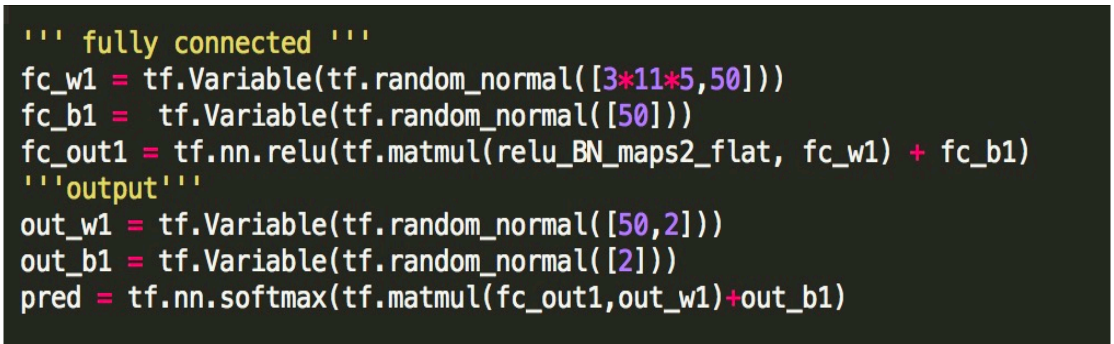
Figure 15: Programming code for implementing convolutional layers in the neural network

2.4.3 Pooling layer

As the training process of CNN which has been introduced in 1.1.2, the pooling layer should be generated with the features of the input data. Pooling layers reduce the number of feature vectors by reducing the sampling. According to the fact that the number of samples is not tremendous in this project. So, I abandon pooling layers in the architecture of neural network.

2.4.4 Fully-connected layer and output layer

In fully connected layer, each node is connected to all the nodes on the top layer, which is used to synthesize the features extracted from the front. ReLU function is also applied in the fully connected layer. The size of the fully-connected layer is 3*11*5, in which 3 and 11 are height and width for the network, 5 is number of channels generated by the neural networks. The code for building the fully-connected



```
''' fully connected '''
fc_w1 = tf.Variable(tf.random_normal([3*11*5,50]))
fc_b1 =  tf.Variable(tf.random_normal([50]))
fc_out1 = tf.nn.relu(tf.matmul(relu_BN_maps2_flat, fc_w1) + fc_b1)
'''output'''
out_w1 = tf.Variable(tf.random_normal([50,2]))
out_b1 = tf.Variable(tf.random_normal([2]))
pred = tf.nn.softmax(tf.matmul(fc_out1,out_w1)+out_b1)
```

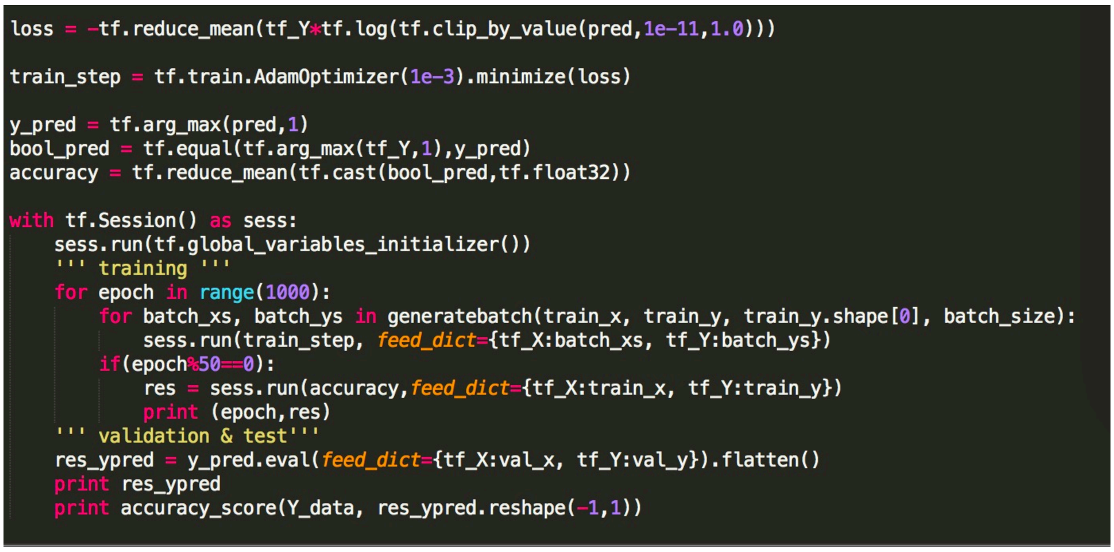
layer and output layer is showed in Figure 16 for reference.

Figure 16: Programming code for implementing fully-connected layer and output layer in the neural network

2.4.5 Data training

After completing the establishment of the convolutional neural network, data training is processed. During the process of data training, Adam optimization algorithm is applied. Adam is an optimization algorithm to find the global optimum which introduced two times square gradient correction. I applied this algorithm because comparing with the basic SGD algorithm, it is less easy to sink into local advantages and has faster speed.

In this project, the process of data training iterates for 1000 cycles. For each circle, MBGD algorithm is proceeded. The program calculates and outputs the accuracy according to the testing data for each 100 circles. The program for data training is showed in Figure 17:



```
loss = -tf.reduce_mean(tf_Y*tf.log(tf.clip_by_value(pred,1e-11,1.0)))

train_step = tf.train.AdamOptimizer(1e-3).minimize(loss)

y_pred = tf.arg_max(pred,1)
bool_pred = tf.equal(tf.arg_max(tf_Y,1),y_pred)
accuracy = tf.reduce_mean(tf.cast(bool_pred,tf.float32))

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    ''' training '''
    for epoch in range(1000):
        for batch_xs, batch_ys in generatebatch(train_x, train_y, train_y.shape[0], batch_size):
            sess.run(train_step, feed_dict={tf_X:batch_xs, tf_Y:batch_ys})
        if(epoch%50==0):
            res = sess.run(accuracy,feed_dict={tf_X:train_x, tf_Y:train_y})
            print (epoch,res)
    ''' validation & test'''
    res_ypred = y_pred.eval(feed_dict={tf_X:val_x, tf_Y:val_y}).flatten()
    print res_ypred
    print accuracy_score(Y_data, res_ypred.reshape(-1,1))
```

Figure 17: Programming code for data training and accuracy calculation in the neural network

2.5 Assessment

Firstly, the accuracy of the neural network according to the training result and the testing data is a significant assessment standard for the model. After building the convolutional neural network model, the accuracy of the model is tested for evaluation and adjustment. Figure 18 shows the process of training and testing the model in flow chart.

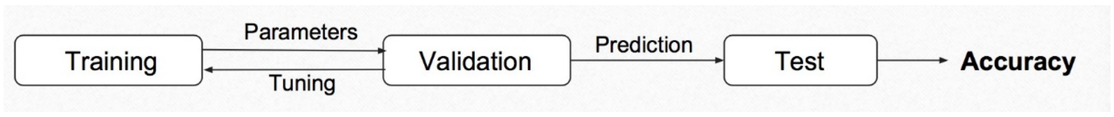


Figure 18: Process of data assessment and testing

As mentioned in 2.4.5, the program is designed to output an accuracy of the neural network for each 100 circles. For each sample in the datasets, the program calculates the expected value of Y based on the training sets. Then, the expected value of Y is compared with this sample's testing sets. After comparison for all the samples, the program calculates and outputs the accuracy of this neural network model.

As mentioned in 1.3, the target accuracy of the model is larger than 0.5. The formula to calculate the accuracy of the model is:

$$Accuracy = \frac{the\ number\ of\ correctly\ classified\ trend\ of\ a\ week\ in\ the\ test/trainging\ samples}{total\ number\ of\ test/training\ samples}$$

If the accuracy is greater than 0.5, the neural network model is assumed satisfactory. If there's no accuracy which is greater than 0.5, new models with different layers and structures may be built and tested.

Secondly, the processing time of training data is also the assessment standard for the neural network. Considering the actual demand, neural networks with long processing time are not practical even with high accuracy. In this project, processing time for each accuracy was outputted for each experiment is recorded for evaluation.

2.6 Applied technologies & tools

As mentioned in 2.5, the processing time of the model one of the assessment standards. Considering the large data size of the project, there is possibility that the processing time may not be satisfactory. Therefore, a GPU is provided by my supervisor. The GPU has the ability of holding the workload to computing intensive parts of applications and accelerating the calculations.

For processing the data, I apply python language to train data and calculate the accuracy. Apart from that, some deep learning frameworks such as Caffe, Torch, Theano, TensorFlow are also under consideration to build multi-layers convolution neutral networks *(Ketkar, 2017).* In this project, TensorFlow, which explored by Google, is used for building deep learning neural network. In addition, Theano, which is a Python library for fast numerical computation, is used for data intensive operation.

3. Difficulties

During the process of data collection, data preprocessing and data training, I have met some difficulties in the different stages of the project. For each problem in the project, corresponding solutions and measures have been raised and implemented. Some of the difficulties and risks are solved, others still need future study and improvement.

3.1 Difficulties in cleaning data

Data collection and cleaning are foundations of this project. So, the accuracy of the data is significant for the result of the neural network model. There are two main difficulties in cleaning data.

On the one hand, there is a situation that one company replace the other one in the list of DJIA. For example, Eastman Kodak Company was replaced by American International Group in 2004. If I drop all the data of the companies which are related to the member changing, the size of datasets will not be large enough. In view of this situation, I decide to use these data as testing sets. If the time range of the data for a company is not long enough, then it cannot use this data to train the model considering its accuracy. Still, it can fit the features of the data.

On the other hand, as mentioned in 2.2, the value of stock indexes may be missed due to many reasons. I treat the stock index of a missing day as the same as its last valid index. However, the accuracy of the data cannot be guaranteed using this method. More efficient solution will be proposed in the future study.

3.2 Speculation for activation function

According to the datasets, I cannot find out obvious regular distribution of the data. So choosing an activation function without foundation faces the possibility that the model show unsatisfactory performance. This risk requires substantial extra time and energy to re-build models and test them. Additionally, this risk has large impact of both the progress and the result of the project.

Initially, Sigmoid functions and Tanh functions have been considered as activation functions. However, both two functions have a common weakness for the CNN model.

Taking Sigmoid function for example, when the input is relatively large or small, its gradient will infinitely close to zero. This circumstance can be observed in Figure 19, as the value of derivative of Sigmoid is close to zero when x is large or small enough. The effect to the model of the input value whose absolute value is large is weak. This weakness of Sigmoid function reduces the gradient and accuracy of the model, which shared by Tanh function.

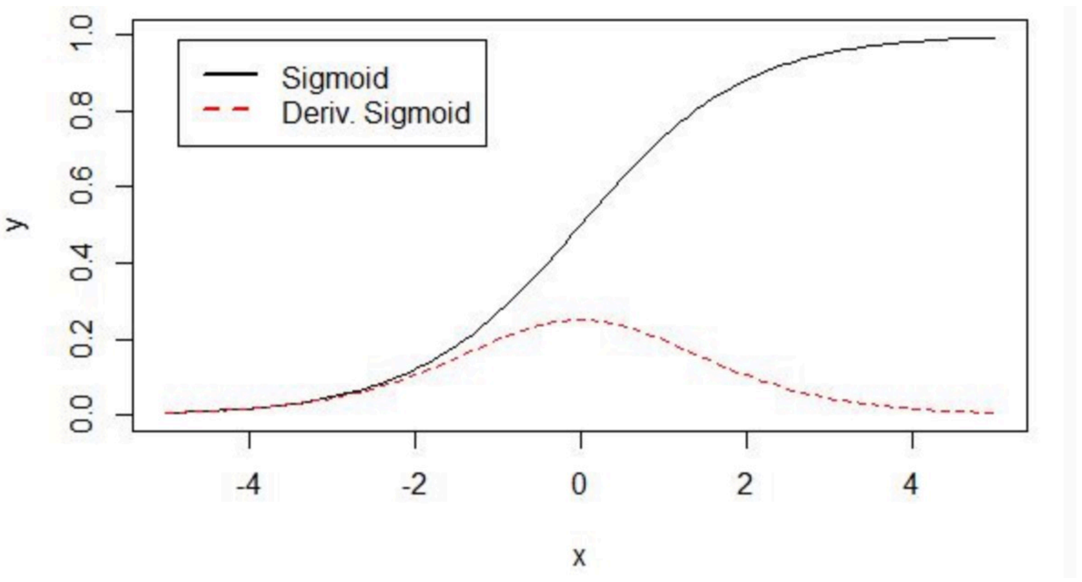


Figure19: Function curve diagram of Sigmoid function and its derivative.

To overcome the weakness of Sigmoid and tanh functions, I take ReLU functions into consideration. Comparing to those two activation functions, ReLU functions solves the vanishing gradient problem because the gradient of ReLU functions is zero in the interval of larger than zero. In the process of modeling and data training, it is found that the speed of derivation calculation of ReLU functions is faster that those two functions. Program implementation for ReLU is an if-else statement, while sigmoid functions need Floating-point four operations. So, ReLU functions are applied in the modeling and data training in this project.

3.3 Improvement for the accuracy

The accuracy is the nuclear of this project, which means multiple methods should be applied to improve the accuracy. Initially, the accuracy of this convolutional neural network model was unsatisfactory, I increase the accuracy of the model by making the adjustments as follow.

Firstly, the data need appropriate method for standardization. Initially, I used the way to standardize the data using the formula as follow: x=(x-mean)/mean and the accuracy of the neural network was lower than 0.5, which is unsatisfactory. After consulting my supervisor on the methods of data standardization, I changed my method to using the method of zero-mean normalization as mentioned in section 2.2. The accuracy increases by changing the method for standardization.

Secondly, increasing the number of convolutional layers improves the accuracy. As mentioned before, in theory, the accuracy of the neural network model positively correlates with the number of convolutional layers. So, in this project, I also constructed another convolutional neural network with three convolutional layers for comparative reference. Because the process of training data requires time, so I decrease the size of the input data, which contain 11 variables for 30 years for the company with the stock code “BA”. Figure 12 shows the experimental results for this comparative experiment. After 500 times of iteration, the accuracy of the convolutional neural network with three convolutional layers achieves 0.7. So, the model can be adjusted by increasing the number of convolutional layers to increase the accuracy.

Comp Exp	no. of circles	Accuracy	Processing time
	0	0.539657	0 min
	100	0.627551	2 min
	200	0.583333	2 min
	300	0.644036	2 min
	400	0.698348	2 min
	500	0.683528	2 min
	600	0.7022676	2 min
	700	0.7273242	2 min
	800	0.708233	2 min
	900	0.7120884	2 min

Table 6: Experimental results for the neural network with three convolutional layers

4. Result and Conclusion

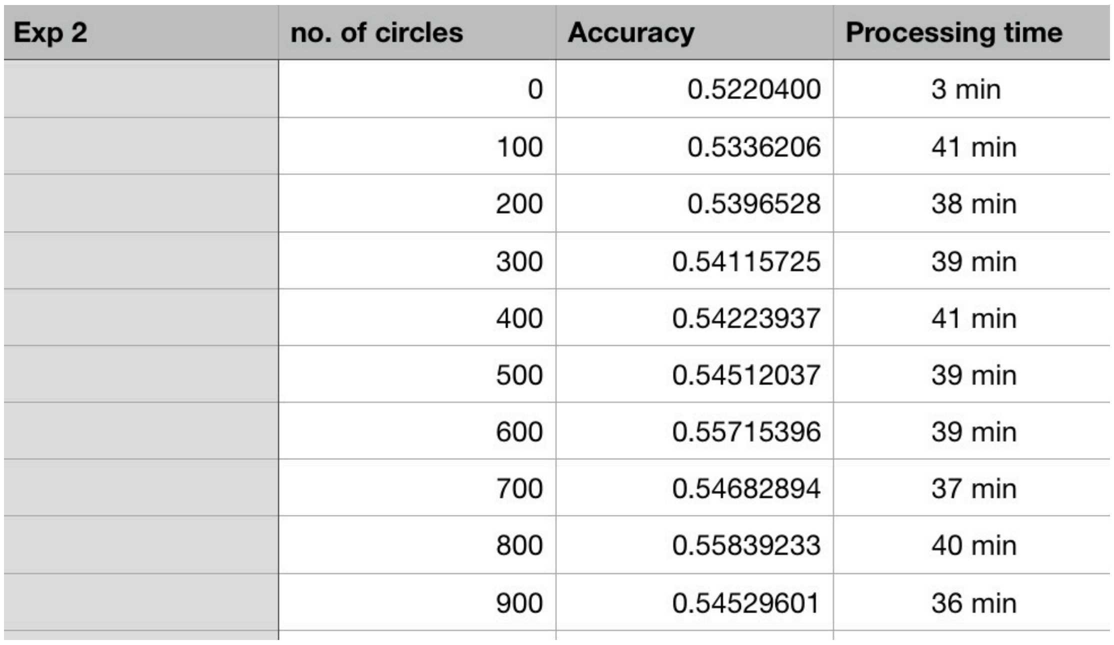
4.1 Results for experiments

Multiple experiments are implemented for more accurate result during the process of data training. I take two sets of experimental results for example, which are showed in Table 7 and Table 8. For each set of experimental results, accuracy and processing time are recorded. Table 7 shows the results when the program calculates and outputs accuracy every 50 circles, while table 8 shows that every 100 circles.

Exp 1	no. of circles	Accuracy	Processing time
	0	0.5220404	1min
	50	0.5436206	19min
	100	0.5493223	19min
	150	0.5536610	20min
	200	0.5411572	19min
	250	0.5412037	18min
	300	0.5579320	19min
	350	0.5423728	20min
	400	0.5593823	17min
	450	0.5536037	18min
	500	0.5468289	18min
	550	0.5579022	18min
	600	0.5522398	19min
	650	0.5482048	20min
	700	0.5563117	18min
	750	0.5498480	18min
	800	0.5568922	18min
	850	0.5481278	19min
	900	0.5532167	18min
	950	0.5527493	18min

Table 7: Experimental results outputted by the program for every 50 circles

Table 8: Experimental results outputted by the program for every 100 circles



Exp 2	no. of circles	Accuracy	Processing time
	0	0.5220400	3 min
	100	0.5336206	41 min
	200	0.5396528	38 min
	300	0.54115725	39 min
	400	0.54223937	41 min
	500	0.54512037	39 min
	600	0.55715396	39 min
	700	0.54682894	37 min
	800	0.55839233	40 min
	900	0.54529601	36 min

According to the results of the experiments, there are three points of discoveries:

Firstly, after enough iterations, the average accuracy is around ***0.55***, which is greater than 0.5. The accuracy of the neural network can be assumed as basically satisfactory, although the improvement of the accuracy still needs to be studied.

Secondly, after around 100 batch size iterations, the accuracy rate is almost approaching the convergence. Batch normalization is effective to reach this result, which increases the efficiency of the process. In practice, it does not require large amount of time to run the iterations many times, which is significant in many applications.

Thirdly, the processing time is longer than the expectation. According to the Figure 10 and Figure 11, to complete 1000 times of iteration, around 6 hours is needed. The ***average processing time for one iteration is 0.4 minute***. Long processing time for data training constrains the adjustments for the neural network model in this project. Besides, it is difficult to achieve an ideal efficiency in the practical applications. In view of this problem, an GPU may be used to reduce the processing time.

4.2 Summary

This report aims to explain the process of how to forecast a particular financial variable by using the idea of deep leaning and CNN to generate the features of the data. After introducing the background knowledge and literature review, this report states the objective and scope of this project.

Additionally, the process of the project is reported in this report. Methodology part introduces each step of the establishment of the model and data training. Difficulties I met in the process of the project are also reported, with corresponding solutions. Last but not least, the experimental results of the project are included. I also evaluate the degree of satisfaction of the experimental results.

According to the experimental results, the average accuracy of the neural network is around 0.55, which shows the objective of the project has been achieved. The accuracy rate is approaching the convergence after around 100 times of iteration. The processing time is relatively long, which needs to be improved in the future.

4.3 Significance

Convolutional neural networks have satisfactory performance in multiple fields such as object recognition, scene annotation, especially image recognition. This project explores the application of CNN in financial market prediction, which unleash the potential of CNN in a new field.

Because the performance of the model is satisfactory, the CNN model can be applied to make predictions of the future trend of financial variables, such as stocks and futures. This application can reduce the economic loss because of accidents such as financial crisis and inflation. Moreover, it can be used to make profit by predicting the future value of financial indices.

4.4 Future prospect

Apart from building the model for financial prediction, convolutional neural networks also provide possibility to combine with time series, such as ARIMA model. It shows a future prospect which is meaningful to the further study.

Reference

Taghi S. & Hoseinzade S. (2013). Forecasting S&P 500 index using artificial neural networks and design of experiments. Journal of industrial Engineering International.

Leuing MT, Daouk H & Chen A. (2000). Forecasting stock indices: a comparison of classification and level estimation models. Int J Forecast 2000, 16: 173-190. 10.1016/S0169-2070(99)00048-5.

Ketkar, N. (2017). Deep learning with Python: a hands-on introduction. Berkeley, CA: Apress. doi:10.1007/978-1-4842-2766-4

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research, 259(2), 689-702.

Krizhevsky, A., Sutskever, I., Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Part of: Advances in Neural Information Processing Systems 25 (NIPS 2012).

Figures

Figure1: Applied go. A neural network with three layers, [Chart]. In Applied go: Perceptrons – the most basic form of a neural network. Retrieved October 29, 2017, from https://appliedgo.net/perceptron/ .

Figure 2: Convolutional Neural Network---- Learning and Practicing. Retrieved June 1st, 2016, from https://www.zhihu.com/question/35222570

Figure3: PERPETUAL ENIGMA. Interactions between convolution layers, [Chart]. In PERPETUAL ENIGMA: Understanding Locally Connected Layers In Convolutional Neural Networks. Retrieved October 29, 2017, from https://prateekvjoshi.com/2016/04/12/understanding-locally-connected-layers-in-convolutional-neural-networks/ .

Figure 6: Revolutions. Pooling, [Chart]. In Revolutions: Deep Learning Part 2: Transfer Learning and Fine-tuning Deep Convolutional Neural Networks. Retrieved October 29, 2017, from https://appliedgo.net/perceptron/ .

Figure 8: Dow Sectors. Market index. Retrieved April, 2016, from https://www.marketindex.com.au/dow-jones .

Figure 10: Artificial Neural Network. Retrieved October, 2015, from http://

www.saedsayad.com/artificial_neural_network.htm

Figure 12: A Quick Introduction to Neural Networks. The data science blog. Retrieved August 9, 2016, from https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

Figure 19: How to Compute the Derivative of a Sigmoid Function (fully worked example). kawahara.ca. Retrieved June 20th, 2014, from http://kawahara.ca/how-to-compute-the-derivative-of-a-sigmoid-function-fully-worked-example/