Using Deep Learning Neural Networks and Candlestick Chart Representation to Predict Stock Market

심층신경망을 이용하여 캔들차트로 표현한 주식시장 예측

팀 연어유희

Abstract

arXiv:1903.12258v1 [q-fin.GN] 26 Feb 2019

Using Deep Learning Neural Networks and Candlestick Chart Representation to Predict Stock Market

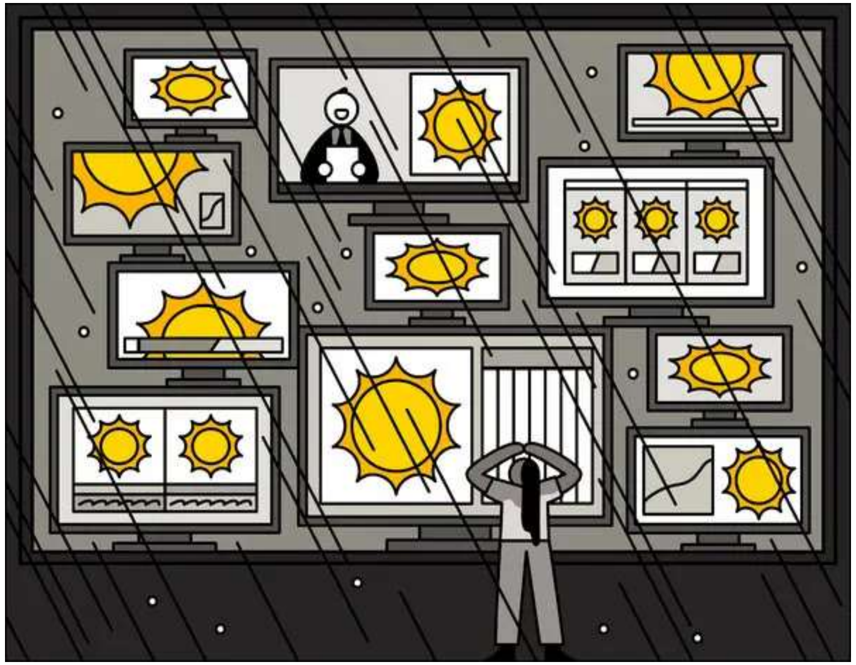
Rosdyana Mangir Irawan Kusuma[1], Trang-Thi Ho[2], Wei-Chun Kao[3], Yu-Yen Ou[1] and Kai-Lung Hua[2]

[1]Department of Computer Science and Engineering, Yuan Ze University, Taiwan Roc
[2]Department of Computer Science and Engineering, National Taiwan University of Science and Technology, Taiwan Roc
[3]Omniscient Cloud Technology

Abstract

Stock market prediction is still a challenging problem because there are many factors effect to the stock market price such as company news and performance, industry performance, investor sentiment, social media sentiment and economic factors. This work explores the predictability in the stock market using Deep Convolutional Network and candlestick charts. The outcome is utilized to design a decision support framework that can be used by traders to provide suggested indications of future stock price direction. We perform this work using various types of neural networks like convolutional neural network, residual network and visual geometry group network. From stock market historical data, we converted it to candlestick charts. Finally, these candlestick charts will be feed as input for training a Convolutional Neural Network model. This Convolutional Neural Network model will help us to analyze the patterns inside the candlestick chart and predict the future movements of stock market. The effectiveness of our method is evaluated in stock market prediction with a promising results 92.2 % and 92.1 % accuracy for Taiwan and Indonesian stock market dataset respectively. The constructed model have been implemented as a web-based system freely available at http://140.138.155.216/deepcandle/ for predicting stock market using candlestick chart and deep learning neural networks.

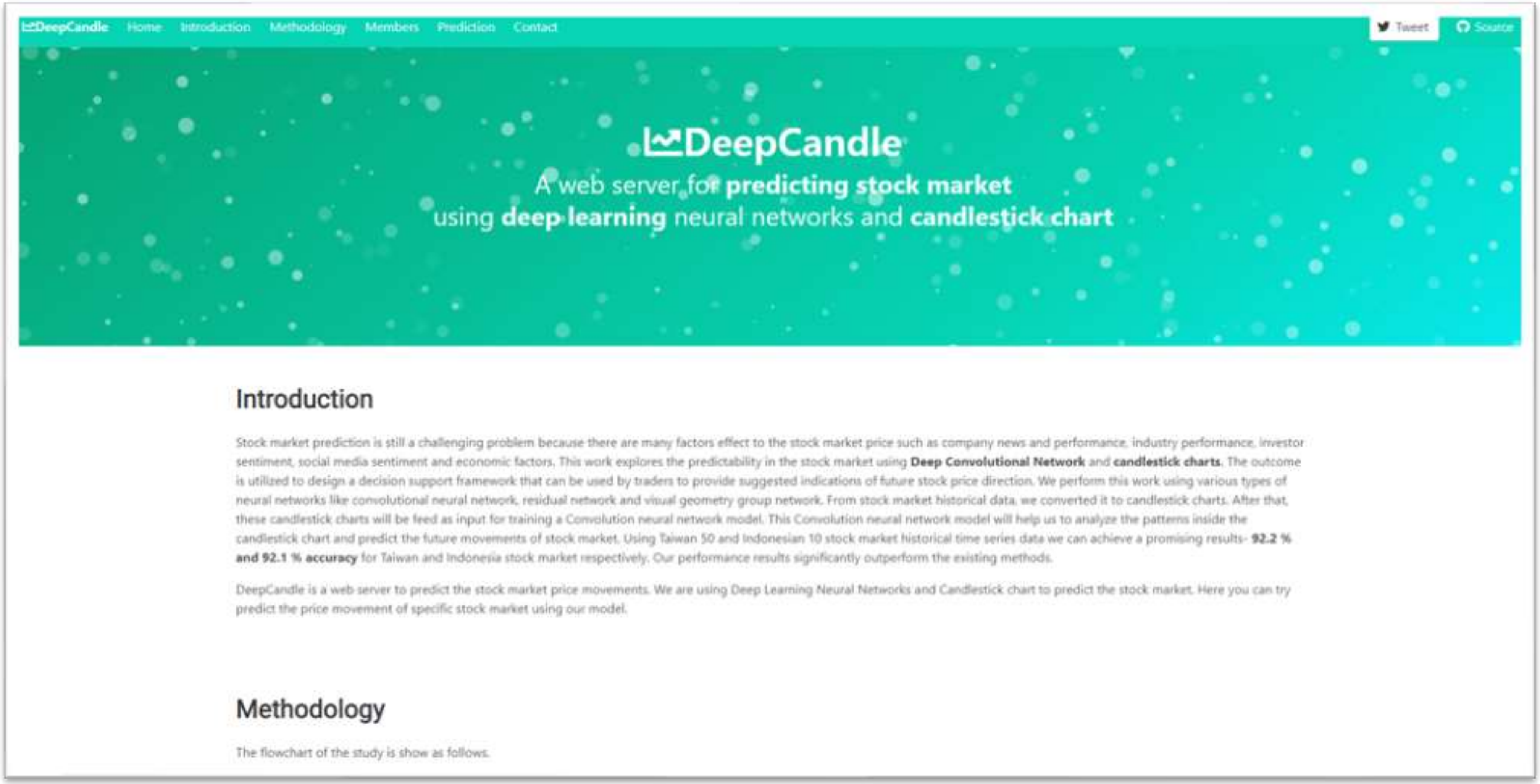
Keywords: Stock Market Prediction, Convolutional Neural Network, Residual Network, Candlestick Chart.



의존하지 말아야 할 한 가지가 있다면,
그것은 바로 예측이다.

#StockMarket_Prediction #Convolutional_Neural_Network #Residual_Network #Candlestick_Chart

Abstract



http://140.138.155.216/deepcandle/

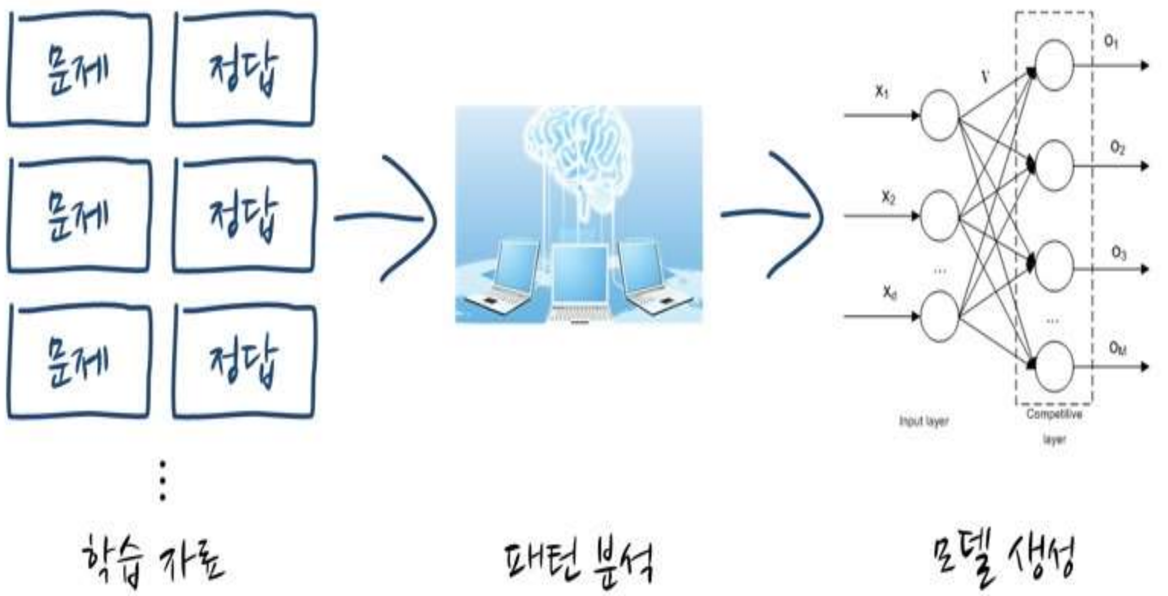
Stock Market Prediction

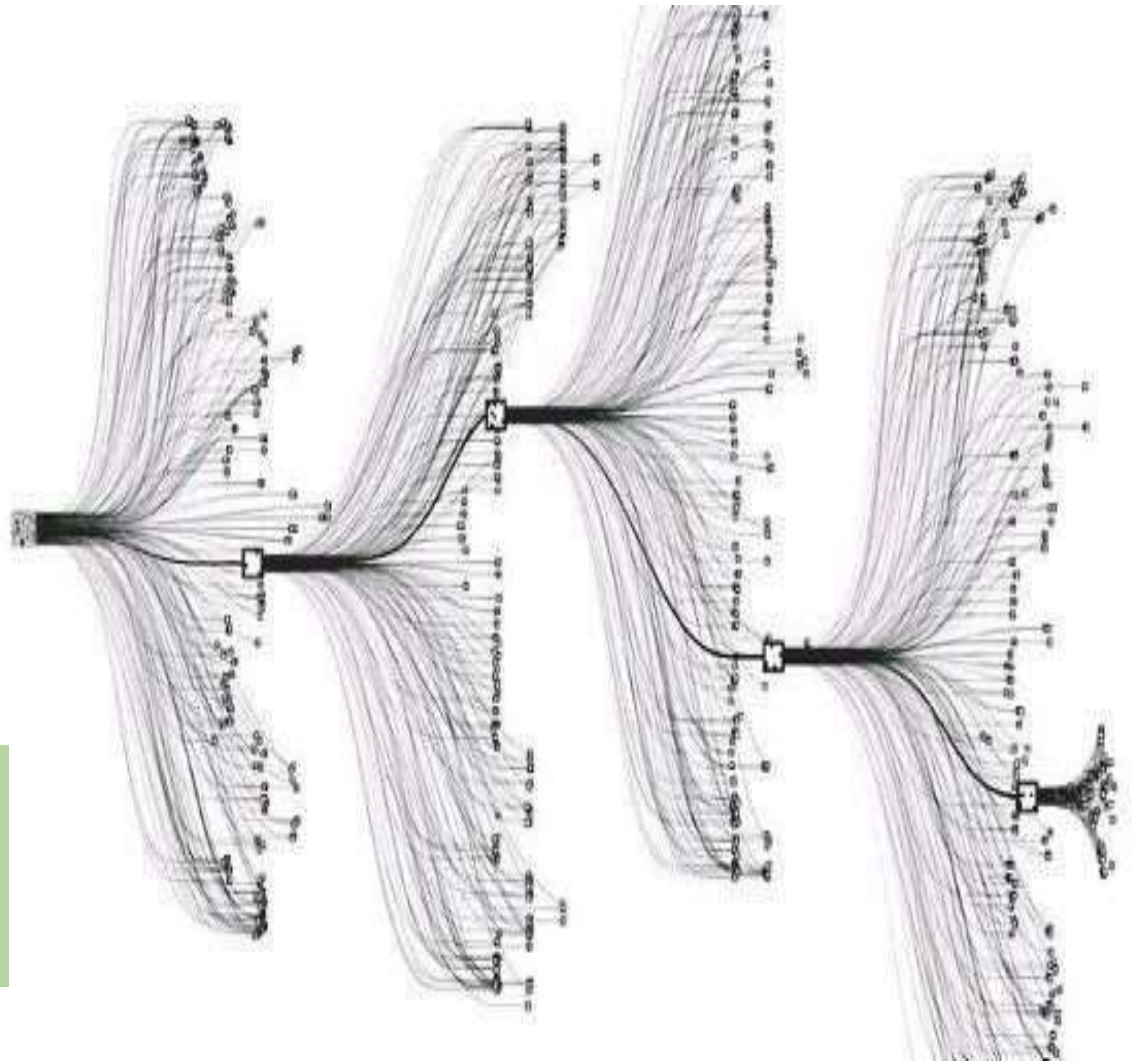
Introduction 1

유진 파마의 3가지 효율적 시장 가설(1965)

★ 약형 효율적 시장가설	현재의 주가에 **과거의 모든 시장정보**가 반영되어 있다 ▶ **과거의 정보**를 통해 초과수익을 얻을 수 없다
★★ 준강형 효율적 시장가설	현재의 주가에는 과거의 시장정보 뿐만 아니라 **공개적으로 이용 가능한 모든 정보**가 반영되어 있다 ▶ **증권분석가들의 정보**로 초과수익을 얻을 수 없다
★★★ 강형 효율적 시장가설	주가는 공개적인 정보 뿐만 아니라 **모든 사적인 정보**까지 완전하게 반영한다 ▶ **내부자만 아는 정보**로도 초과수익을 얻을 수 없다

Introduction 2





On the other occasion, from historical data of stock market converted into audio wavelength using deep convolutional wave net architecture can be applied to forecast the stock market movement.

Introduction 3

TAIWAN
STOCK EXCHANGE

IDX
Indonesia Stock Exchange
Bursa Efek Indonesia

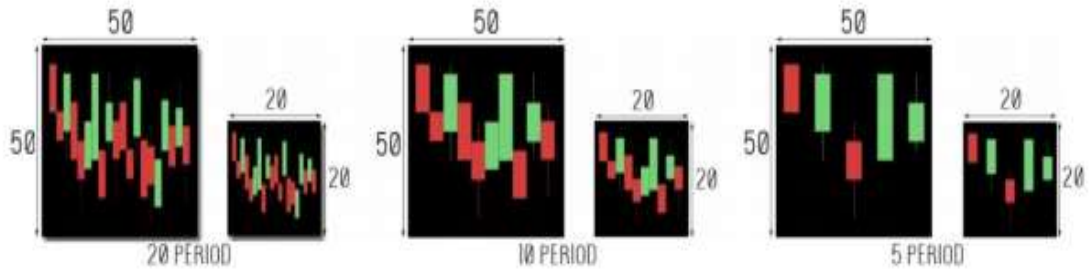


Figure 2: Proposed candlestick chart without volume indicator in different period time and size.

The goal is to analyze the correlation of some parameters such as period time, image size, feature set with the movement of stock market to check whether it will be going up or going down in the next day.

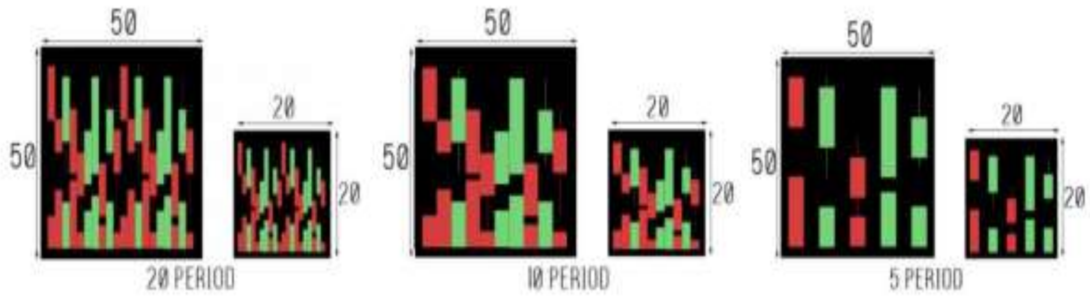
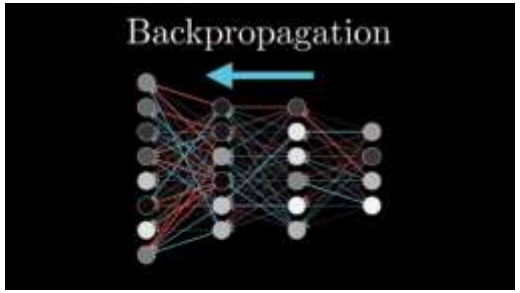
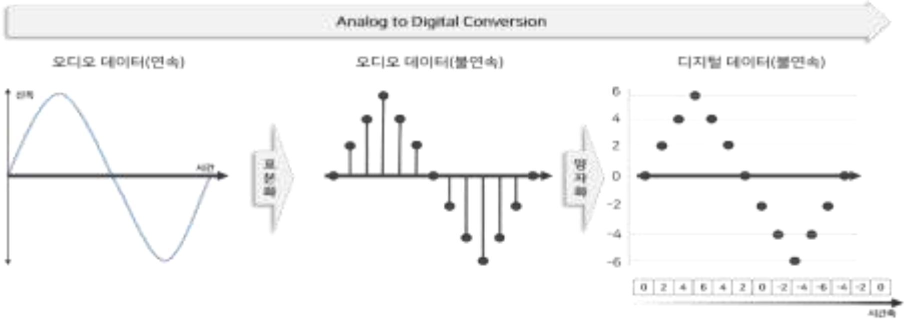


Figure 3: Proposed candlestick chart with volume indicator in different period time and size.

Related Work 1

	In 1990, (Schneburg) conducted a study using data from a randomly selected German stock market, then using the **back-propagation method** for their machine learning architecture
	(J. Bollen) reported the sentiment analysis method by taking data from one of the famous microblogging site **Twitter** to predict the **Dow Jones Industrial Average** (DJIA) stock market movements
	(Borovykh, Bohte et al.) tried to use the deep convolutional **wave net architecture** method to perform analysis and prediction using data from S&P500 and CBOE

Related Work 2

Related works using candlestick charts in their research
(Zhang, Zhang et al. 2018) input data is not only from historical stock trading data, a financial **news** and users sentiments from **social media** can be correlated to predict the movement in stock market.
(do Prado, Femeda et al. 2013) used the candlestick chart to learn the pattern contained in Brazilian stock market by using **sixteen candlestick patterns**.
(Tsai and Quan 2014) utilized the candlestick chart to combine with seven different wavelet-based textures to analyze the candlestick chart.
(Hu, Hu et al. 2017) used the candlestick chart to build a decision-making system in stock market investment. They used the convolutional encoder to **learn the patterns contained in the candlestick chart**.
(Patel, Shah et al. 2015) used **ten technical parameters** from stock trading data for their input data and compare **four prediction models**, Artificial Neural Network (ANN), Support Vector Machine (SVM), random forest and nave-Bayes.
(Khaidem, Saha et al. 2016) combine the Random Forest with technical indicator such as Relative Strength Index (RSI) shown a good performance.

Dataset - Data Collection

yahoo! finance

TAIWAN STOCK EXCHANGE

IDX
Indonesia Stock Exchange
Bursa Efek Indonesia

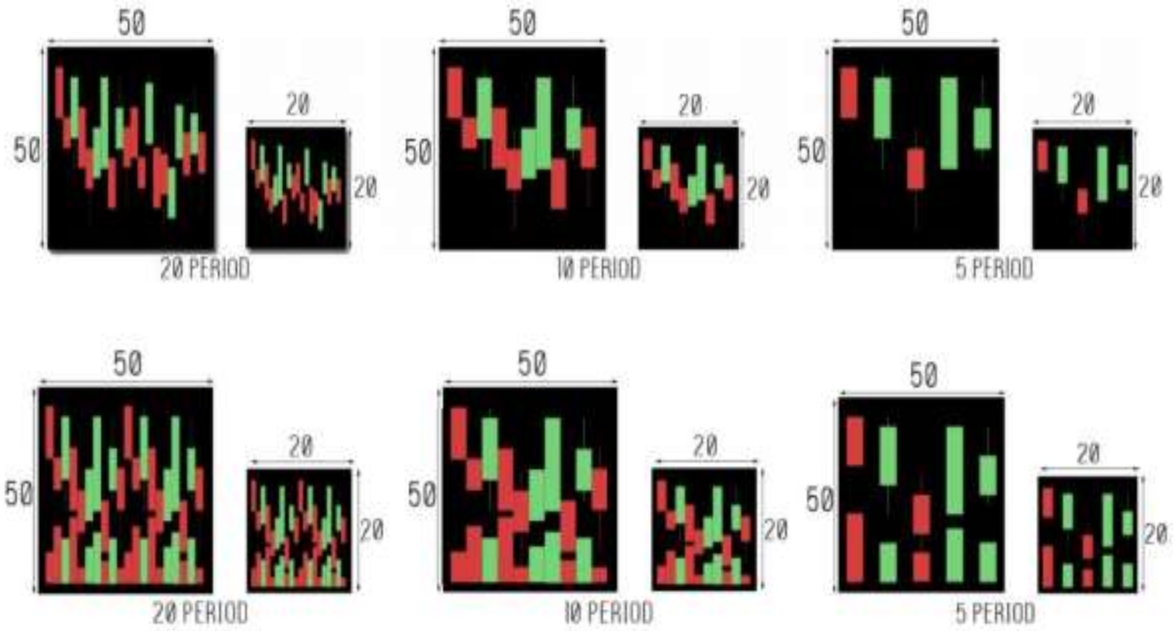
Table 1: The period time of our dataset, separated between the training, testing and independent data.

Stock Data	Training Data		Testing Data		Independent Data	
	Start	End	Start	End	Start	End
TW50	2000/01/01	2016/12/31	2017/01/01	2018/06/14	2017/01/01	2018/06/14
ID10	2000/01/01	2016/12/31	2017/01/01	2018/06/14	2017/01/01	2018/06/14

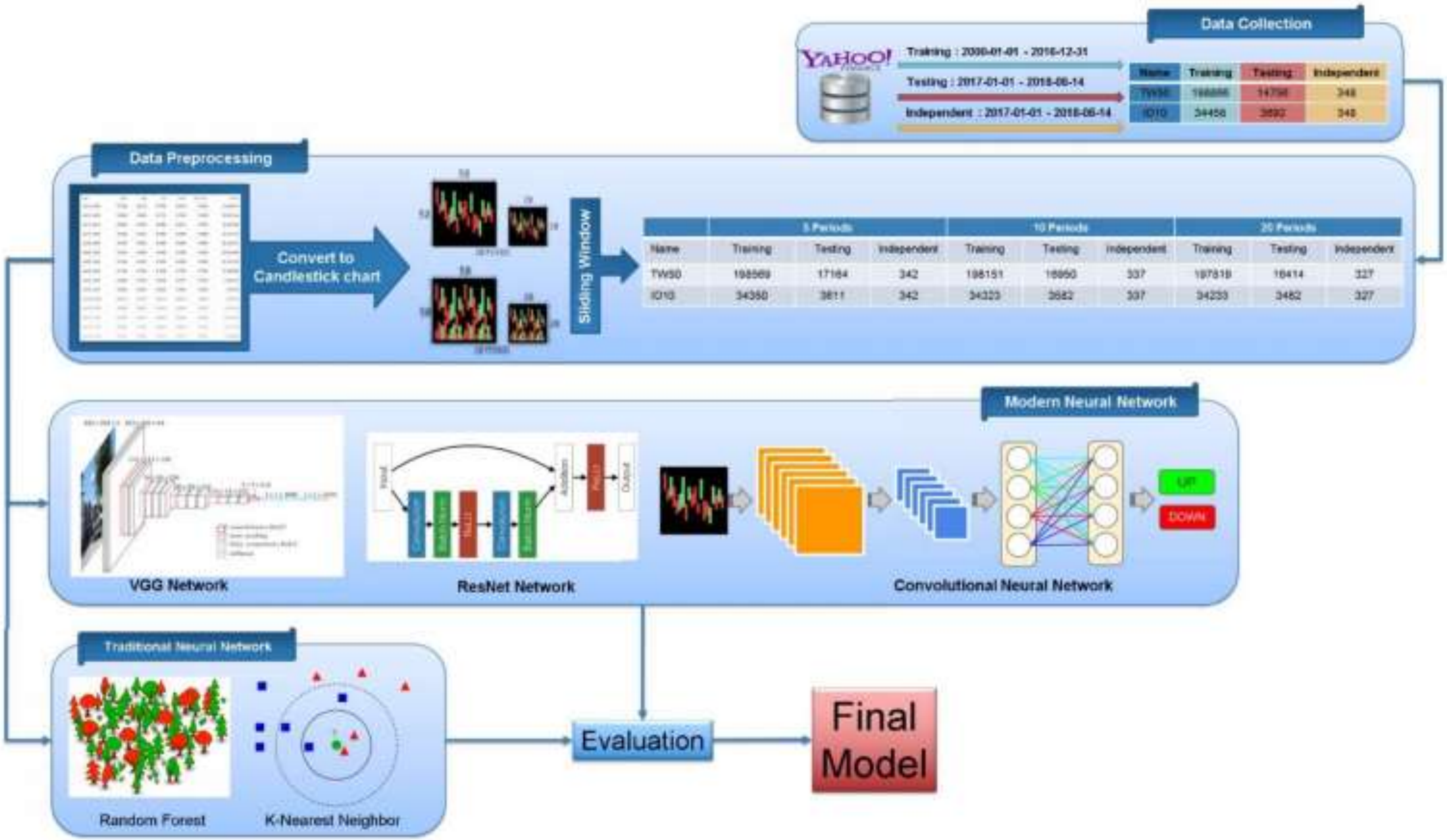
Dataset - Data Preprocessing

348 lines (348 sloc) | 22.3 KB

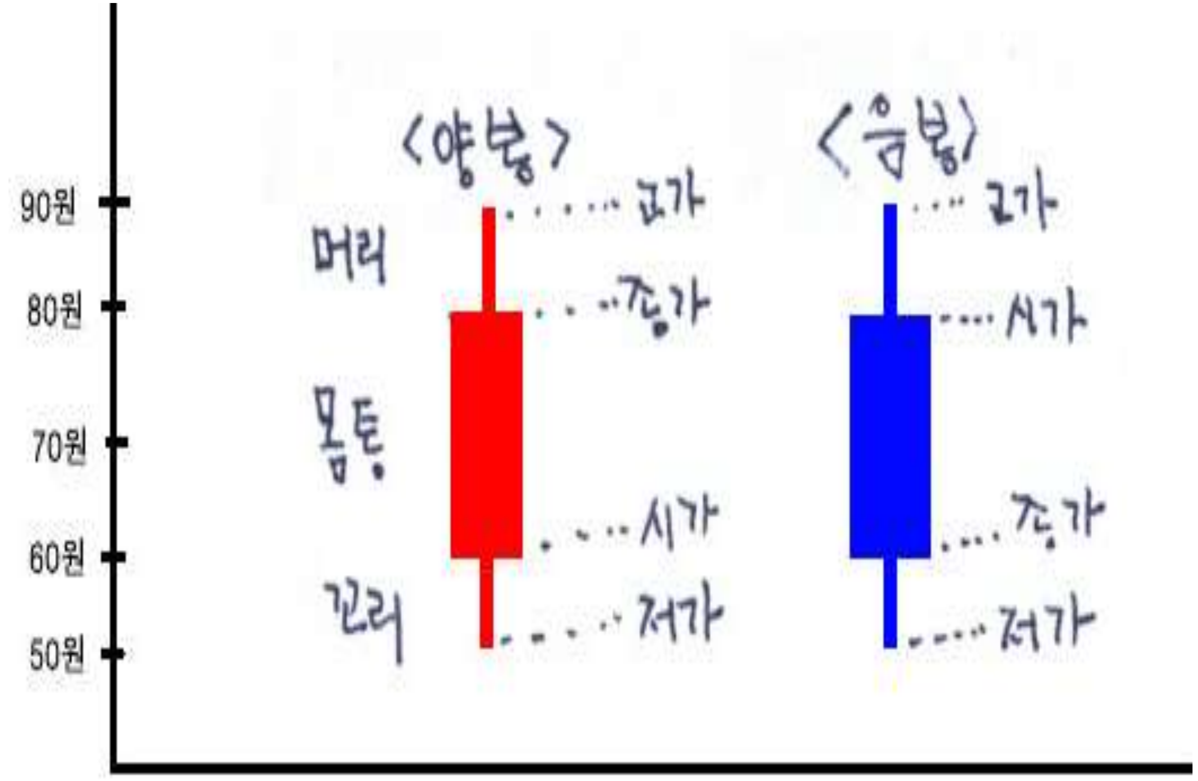
Date	Open	High	Low	Close	Adj Close	Volume
2017-01-03	71.699997	71.900002	71.5	71.900002	68.466629	2325000
2017-01-04	72.0	72.099998	71.75	71.849998	68.419014	4278000
2017-01-05	71.849998	72.5	71.800003	72.5	69.037979	4566000
2017-01-06	72.449997	72.75	72.449997	72.650002	69.180817	3746000
2017-01-09	72.75	72.949997	72.300003	72.449997	68.990364	5032000
2017-01-10	72.449997	72.599998	72.400002	72.400002	68.942757	1834000
2017-01-11	72.800003	72.849998	72.400002	72.400002	68.942757	1894000
2017-01-12	72.599998	73.300003	72.599998	73.150002	69.656937	4850000
2017-01-13	72.800003	72.800003	72.599998	72.599998	69.133202	3903000
2017-01-16	72.400002	72.400002	71.800003	72.0	68.561852	2412000
2017-01-17	72.0	72.400002	72.0	72.300003	68.847534	1569000
2017-01-18	72.199997	72.199997	72.050003	72.150002	68.704689	3831000
2017-01-19	71.849998	72.099998	71.599998	71.949997	68.514236	4061000
2017-01-20	71.949997	72.25	71.949997	72.150002	68.704689	2415000
2017-01-23	72.550003	73.0	72.550003	72.75	69.276039	5959000
2017-01-24	73.0	73.300003	72.900002	73.300003	69.799782	4565000
2017-02-02	73.5	73.650002	72.75	72.900002	69.418877	6984000
2017-02-03	72.949997	73.150002	72.650002	72.849998	69.371262	2799000
2017-02-06	73.0	73.400002	72.949997	73.25	69.75215899999999	2936000

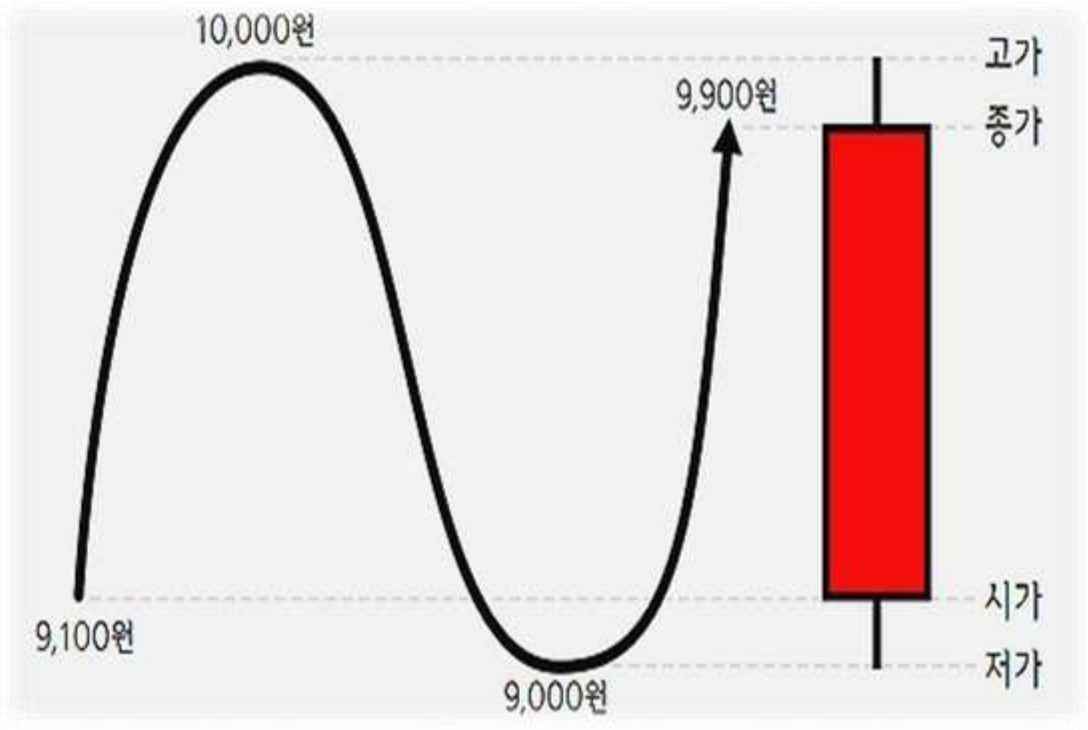


Methodology

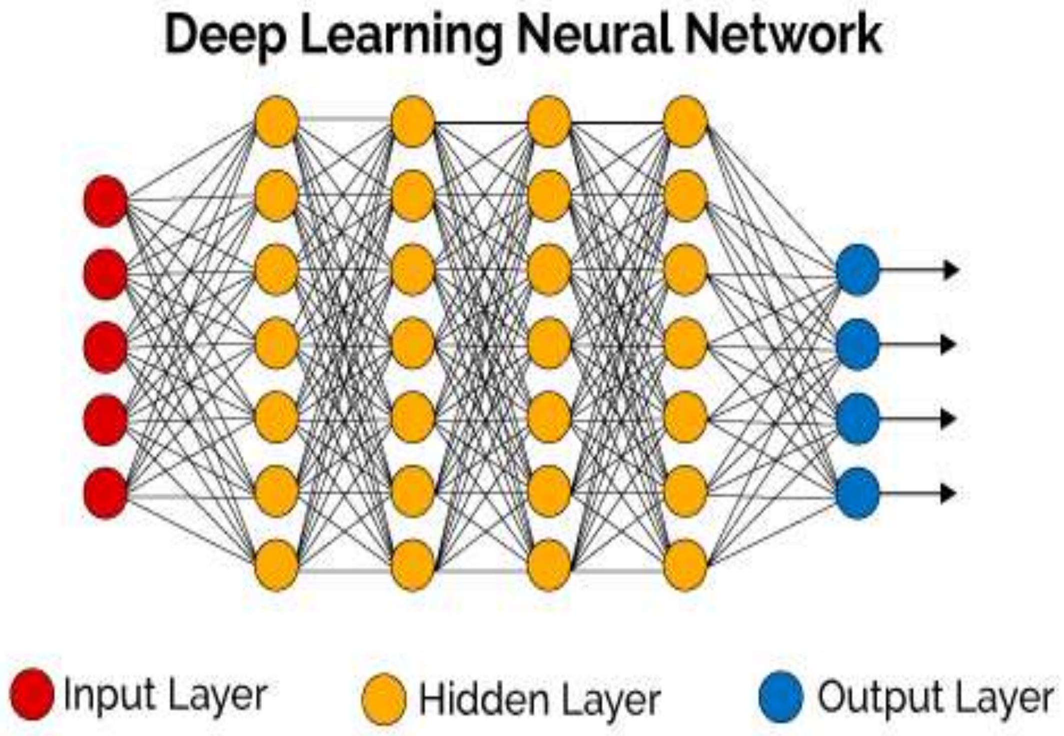


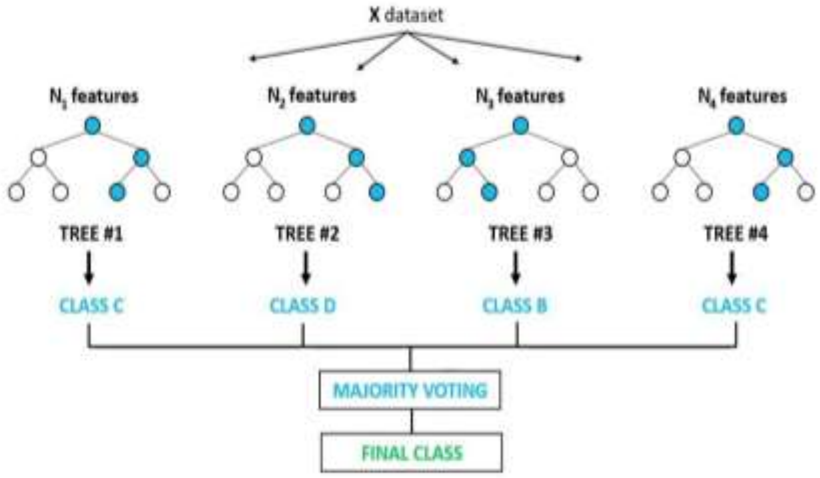
Methodology - Candlestick Chart

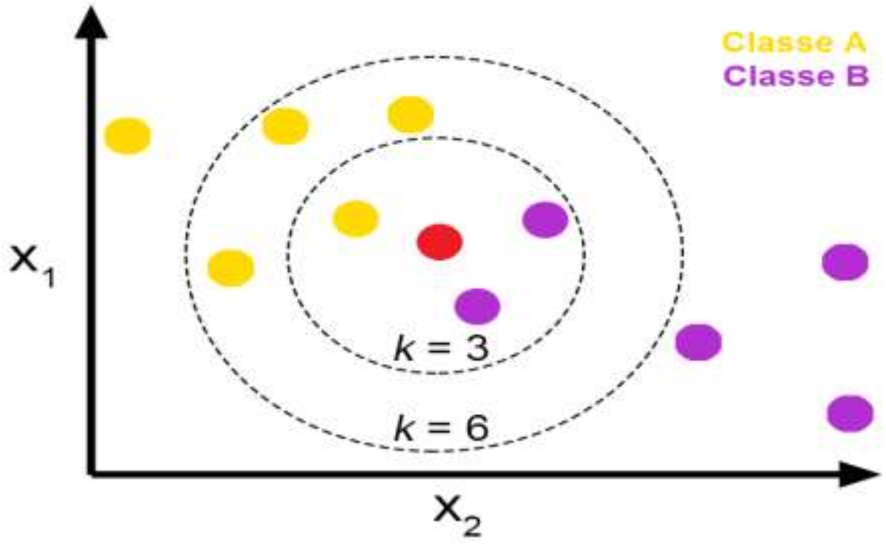




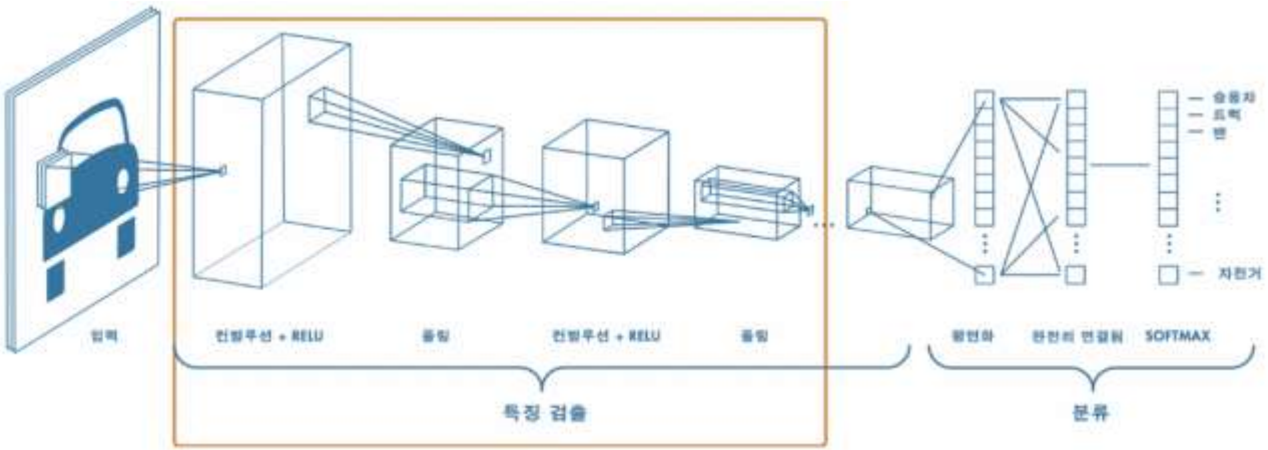
Methodology - Learning Algorithm

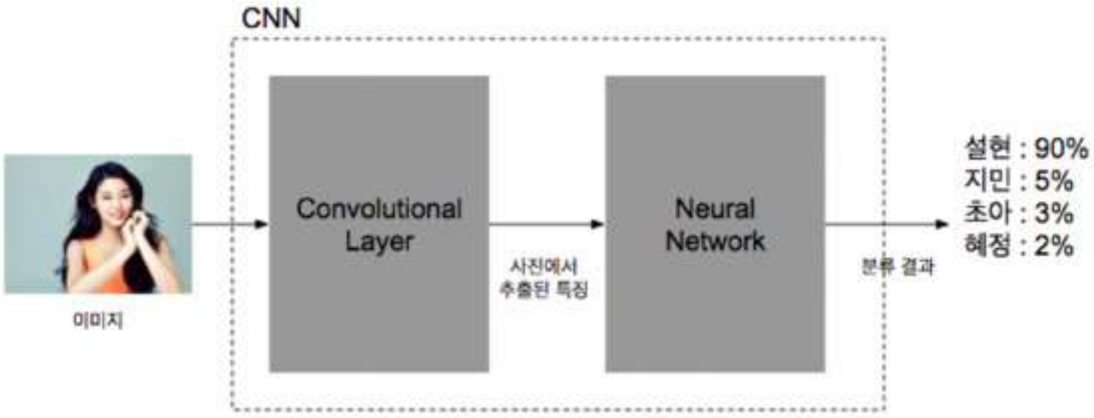


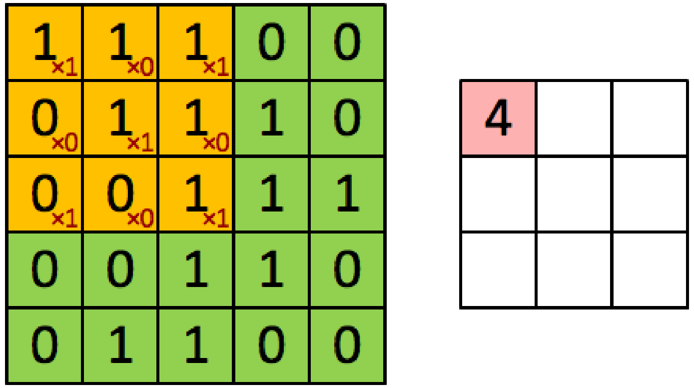




Methodology - Convolutional Neural Network 1

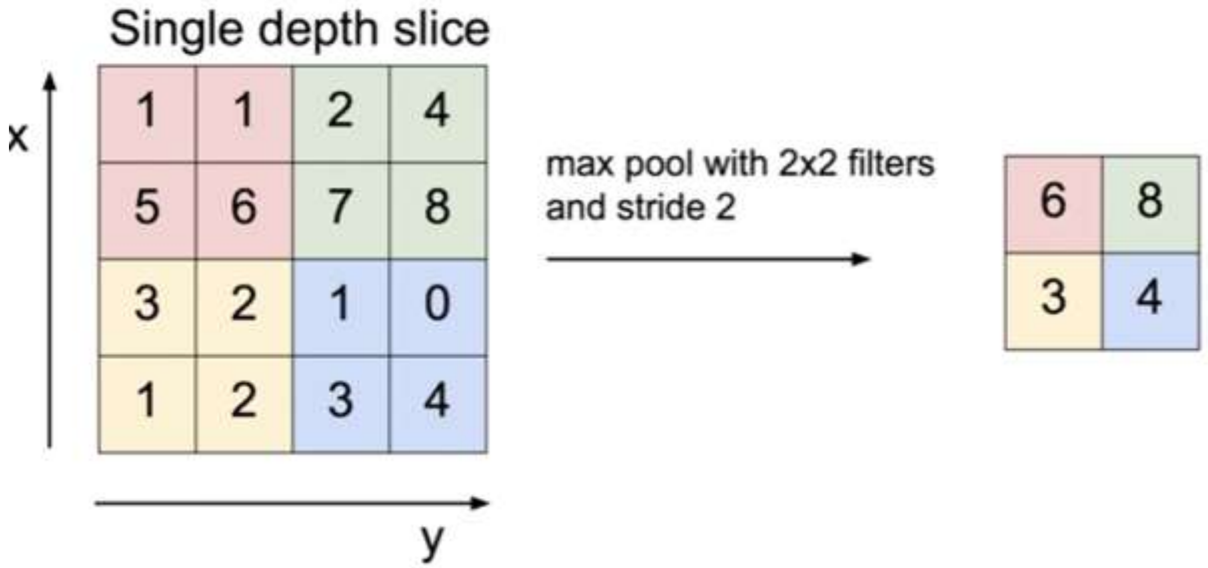






Image

Convolved Feature



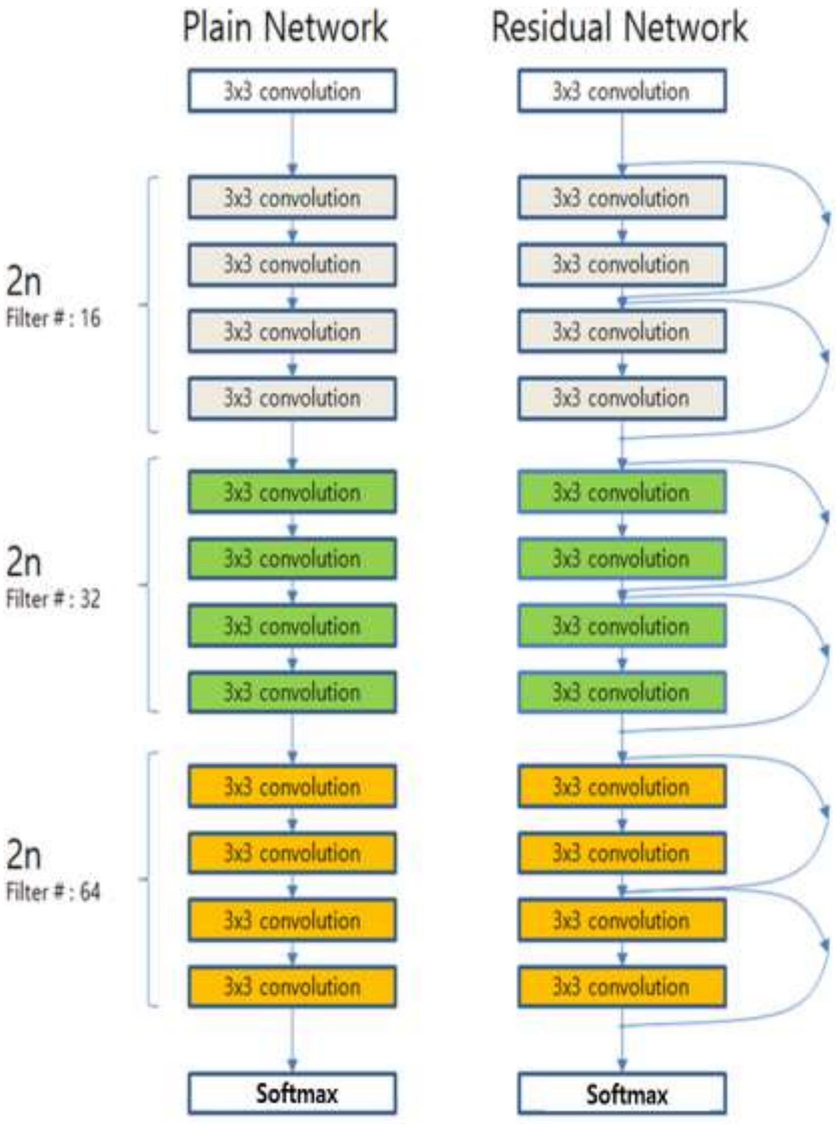
Methodology - Convolutional Neural Network 2

Input
Conv2D-32 ReLU
max-pooling
Conv2D-48 ReLU
max-pooling
Dropout
Conv2D-64 ReLU
max-pooling
Conv2D-96 ReLU
max-pooling
Dropout
Flatten
Dense-256
Dropout
Dense-2

Table 2: Our proposed CNN architecture.

```
def build_model(SHAPE, nb_classes, bn_axis, seed=None):
    if seed:
        np.random.seed(seed)
    input_layer = Input(shape=SHAPE)
    # Step 1
    x = Conv2D(32, 3, 3, init='glorot_uniform',
               border_mode='same', activation='relu')(input_layer)
    # Step 2 - Pooling
    x = MaxPooling2D(pool_size=(2, 2))(x)
    # Step 1
    x = Conv2D(48, 3, 3, init='glorot_uniform', border_mode='same',
               activation='relu')(x)
    # Step 2 - Pooling
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Dropout(0.25)(x)
    # Step 1
    x = Conv2D(64, 3, 3, init='glorot_uniform', border_mode='same',
               activation='relu')(x)
    # Step 2 - Pooling
    x = MaxPooling2D(pool_size=(2, 2))(x)
    # Step 1
    x = Conv2D(96, 3, 3, init='glorot_uniform', border_mode='same',
               activation='relu')(x)
    # Step 2 - Pooling
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Dropout(0.25)(x)
    # Step 3 - Flattening
    x = Flatten()(x)
    # Step 4 - Full connection
    x = Dense(output_dim=256, activation='relu')(x)
    # Dropout
    x = Dropout(0.5)(x)
    x = Dense(output_dim=2, activation='softmax')(x)
    model = Model(input_layer, x)
    return model
```

Methodology - Residual Network



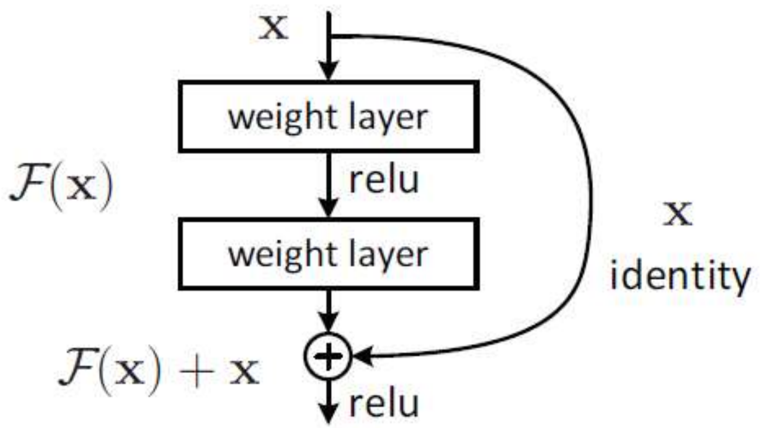


Figure 2. Residual learning: a building block.

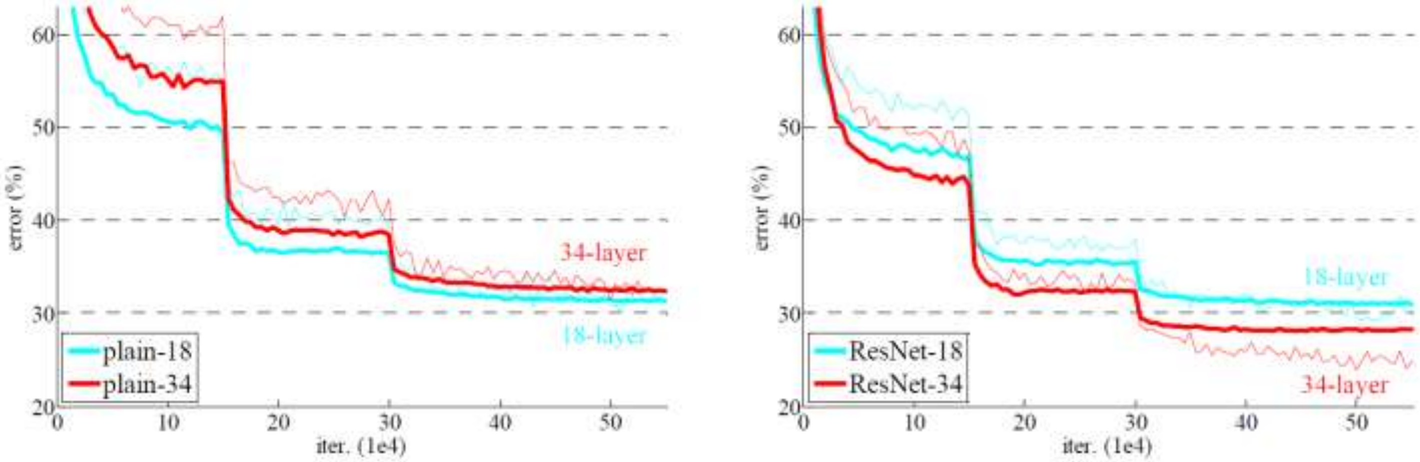
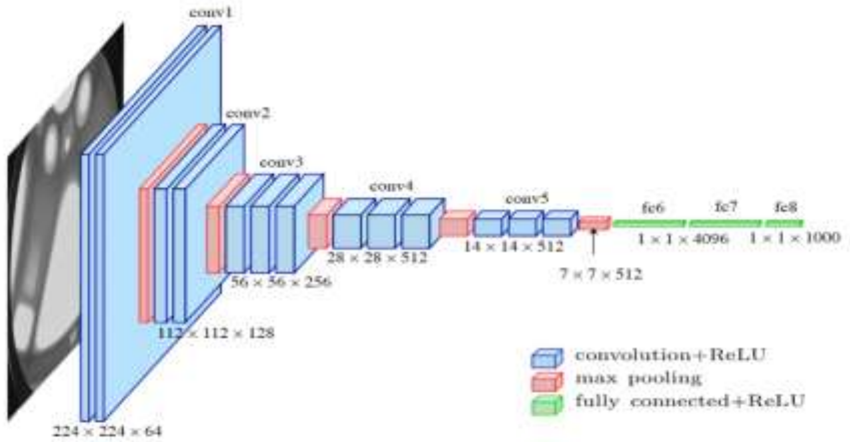
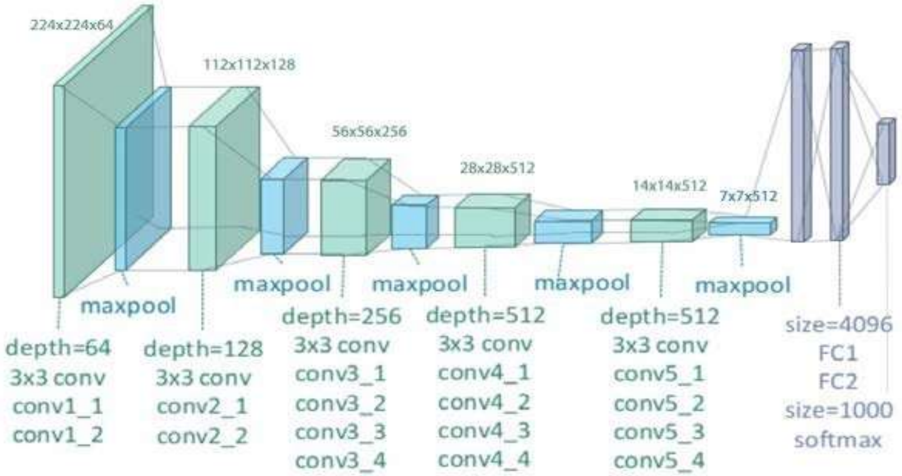


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

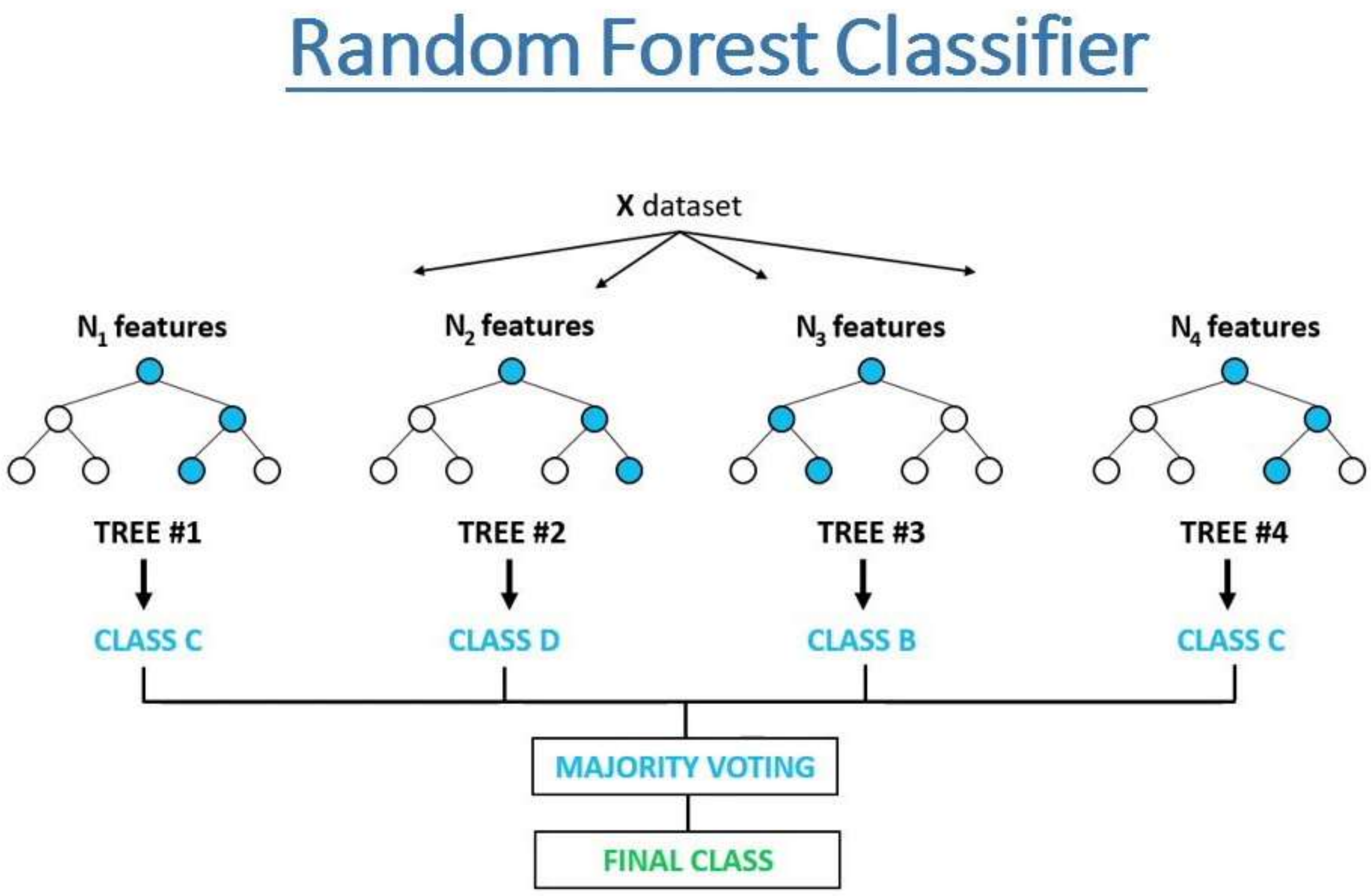
Methodology - VGG Network

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					



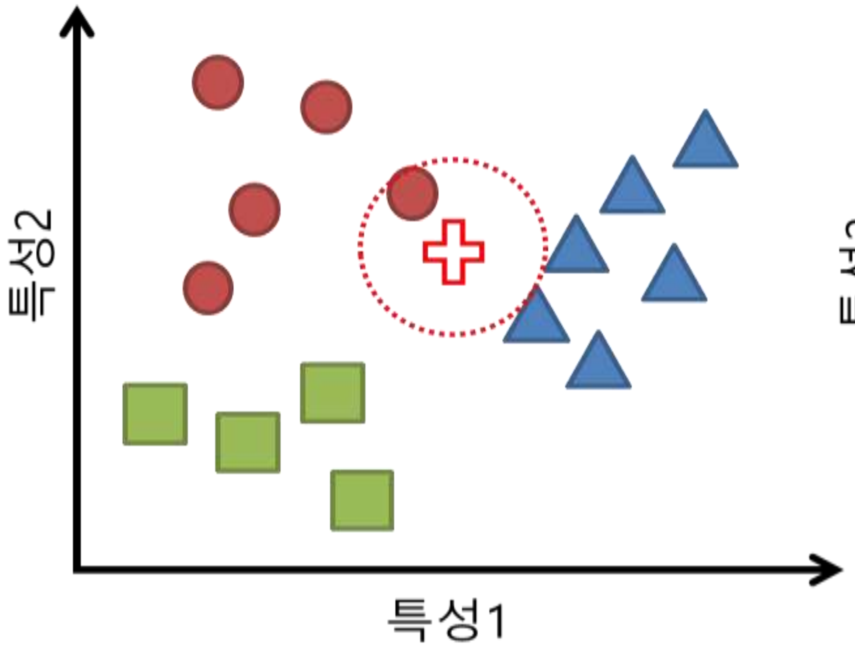


Methodology – Random Forest

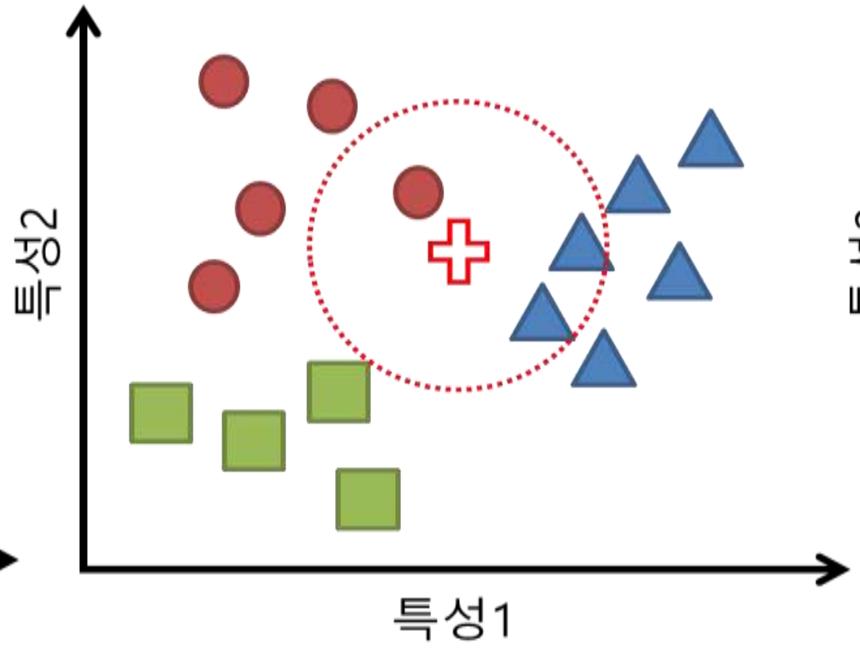


- 월등히 높은 정확성
- 간편하고 빠른 학습 및 테스트
- 변수 소거 없이 수천 개의 입력 변수가능
- 임의화를 통한 좋은 일반화 성능
- 다중 클래스 알고리즘 특성
- Bagging 방지

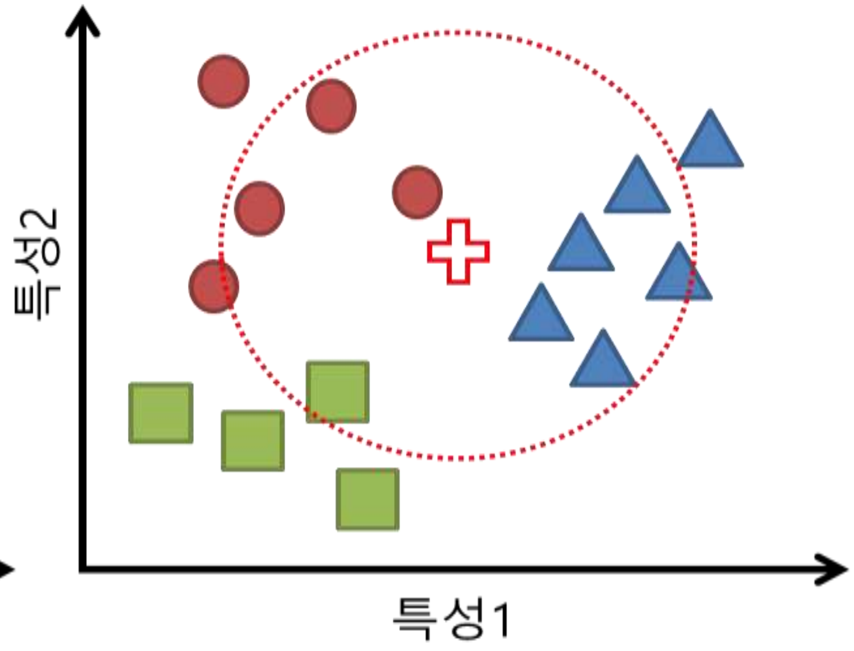
Methodology – K-Nearest Neighbors



k=1일 때



k=3일 때



k=9일 때

- 단순하고 효율적이다
- 기저 데이터 분포에 대한 가정을 하지 않는다.
- 훈련 단계가 빠르다
- 수치 기반 데이터 분류 작업에서 성능이 우수하다

최소-최대 정규화(min-max normalization)

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

변수 X의 범위를
0%에서 100%까지로 나타냄

z-점수 표준화(z-score standardization)

$$X_{new} = \frac{X - \mu}{\sigma} = \frac{X - \mathrm{Mean}(X)}{StdDev(X)}$$

변수 X의 범위를
평균의 위 또는 아래로 몇 표준 편차만큼
떨어져 있는지 관점으로 확대/축소

Methodology – Performance Evaluation

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specitivity = \frac{TN}{TN + FP}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

(1) 민감도(Recall): 진짜를 진짜로 예측될 확률

(2) 정밀도(precision): 예측값이 얼마나 맞았는지 확률

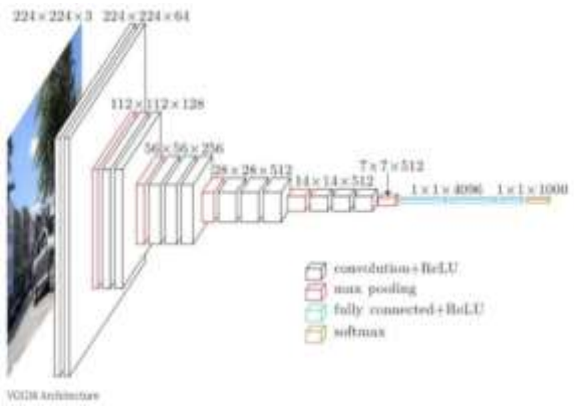
(3) 정확도 (accuracy):
예측값과와 실제값이 맞은 건수 / 전체 데이터 수

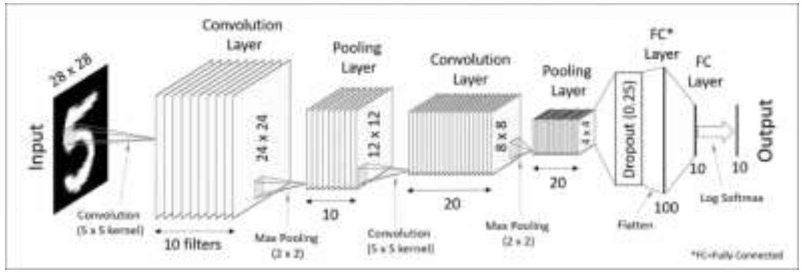
(4) Matthew 상관 계수 (MCC):

phi coefficient라고도 불리는데, 옳고 그름을 판별하는 이진분류(binary classification)에 사용되는 metric이다.
-1과 1 사이의 값
1에 가까울수록 두 관측치가 비슷하다고 본다.

Experimental Results and Discussion

In this section, we perform classification based on some traditional and modem machine learning algorithms (random forest, kNN, residual network, VGG, CNN) and then evaluate the performance of our best classification algorithm compared to **three state-of-the-art methods.**





Classification for Taiwan 50 Dataset

Table 3: Summary result of Taiwan 50 with their best classifier for each trading days and image dimension.

	Classifier	Period	Dimension	Sensitivity	Specitivity	Accuracy	MCC
with volume	CNN	5	50	83.2	83.8	83.5	0.67
	CNN	10	50	88.6	87.3	88.0	0.758
	CNN	20	50	**91.6**	**91.3**	**91.5**	**0.827**
	CNN	5	20	83.9	82.7	83.3	0.666
	Random Forest	10	20	87.0	88.3	87.6	0.751
	CNN	20	20	90.8	90.2	90.6	0.808
without volume	CNN	5	50	83.6	85.1	84.4	0.687
	CNN	10	50	89.2	88.1	88.7	0.773
	CNN	20	50	**93.3**	90.7	**92.2**	**0.84**
	CNN	5	20	84.8	83.0	83.9	0.678
	CNN	10	20	88.0	88.2	88.1	0.761
	CNN	20	20	81.7	91.4	91.0	0.817

Classification for Indonesia 10 Dataset

Table 4: Summary result of Indonesia 10 with their best classifier for each trading days and image dimension.

	Classifier	Period	Dimension	Sensitivity	Specitivity	Accuracy	MCC
with volume	ResNet50	5	50	80.7	85.4	83.1	0.661
	ResNet50	10	50	88.6	88.4	88.5	0.77
	CNN	20	50	**90.0**	**90.1**	**90.0**	**0.798**
	ResNet50	5	20	78.8	82.3	80.6	0.612
	CNN	10	20	83.3	85.4	84.3	0.686
	CNN	20	20	89.1	84.6	87.1	0.738
without volume	ResNet50	5	50	79.1	87.9	83.3	0.671
	CNN	10	50	87.5	86.6	87.1	0.74
	CNN	20	50	**92.1**	92.1	**92.1**	**0.837**
	CNN	5	20	83.4	82.4	82.9	0.658
	CNN	10	20	85.4	85.6	85.5	0.708
	VGG16	20	20	91.5	89.7	90.7	0.808

Independent Testing Result

Table 5: Summary result of Taiwan 50 with their best classifier for each trading days and image dimension.

	Classifier	Period	Dimension	Sensitivity	Specitivity	Accuracy	MCC
with volume	CNN	5	50	83.2	83.8	83.5	0.67
	CNN	10	50	88.6	87.3	88.0	0.758
	CNN	20	50	**91.6**	**91.3**	**91.5**	**0.827**
	CNN	5	20	83.9	82.7	83.3	0.666
	Random Forest	10	20	87.0	88.3	87.6	0.751
	CNN	20	20	90.8	90.2	90.6	0.808
without volume	CNN	5	50	83.6	85.1	84.4	0.687
	CNN	10	50	89.2	88.1	88.7	0.773
	CNN	20	50	**93.3**	90.7	**92.2**	**0.84**
	CNN	5	20	84.8	83.0	83.9	0.678
	CNN	10	20	88.0	88.2	88.1	0.761
	CNN	20	20	81.7	91.4	91.0	0.817

Comparison 1

Table 7: Comparison result with Khaidem.

Khaidem, Saha et al. Samsung					
Name	Trading Period	ACC	Precision	Recall	Specificity
Khaidem	1 Month	86.8	88.1	87.0	0.865
Our	1 Month	**87.5**	88.0	**87.0**	**0.891**
Khaidem	2 Month	90.6	91.0	92.5	0.88
Our	2 Month	**94.2**	**94.0**	**94.0**	0.862
Khaidem	3 Month	93.9	92.4	95.0	0.926
Our	3 Month	**94.5**	**94.0**	**95.0**	0.882
Khaidem, Saha et al. Apple					
Khaidem	1 Month	88.2	89.2	90.7	0.848
Our	1 Month	**89.6**	**90.0**	90.0	**0.863**
Khaidem	2 Month	93.0	94.1	93.8	0.919
Our	2 Month	**93.6**	94.0	**94.0**	0.877
Khaidem	3 Month	94.5	94.5	96.1	0.923
Our	3 Month	**95.6**	**96.0**	96.1	0.885
Khaidem, Saha et al. GE					
Khaidem	1 Month	84.7	85.5	87.6	0.809
Our	1 Month	**90.2**	**90.0**	90.0	0.86
Khaidem	2 Month	90.8	91.3	93.0	0.876
Our	2 Month	**97.8**	**98.0**	**98.0**	**0.993**
Khaidem	3 Month	92.5	93.1	94.5	0.895
Our	3 Month	**97.4**	**98.0**	**98.0**	**0.983**

(Khaidem, Saha et al. 2016) combine the Random Forest with technical indicator such as Relative Strength Index (RSI)

Comparison 2

Table 8: Comparison result with Patel.

	S&P BSE SENSEX		NIFTY 50	
	ACC	F-Measure	ACC	F-Measure
Patel	89.84	0.9026	89.52	0.8935
Our	**97.2**	**0.97**	**93.4**	**0.93**
	Reliance Industry		Infosys	
Patel	92.22	0.9234	90.01	0.9017
Our	**93.9**	**0.94**	**93.9**	**0.94**

(Patel, Shah et al. 2015) used **ten technical parameters** from stock trading data for their input data and compare **four prediction models**, Artificial Neural Network (ANN), Support Vector Machine (SVM), random forest and nave-Bayes.

F1 score = 2 * (Precision * Recall) / (Precision + Recall)

Comparison 3

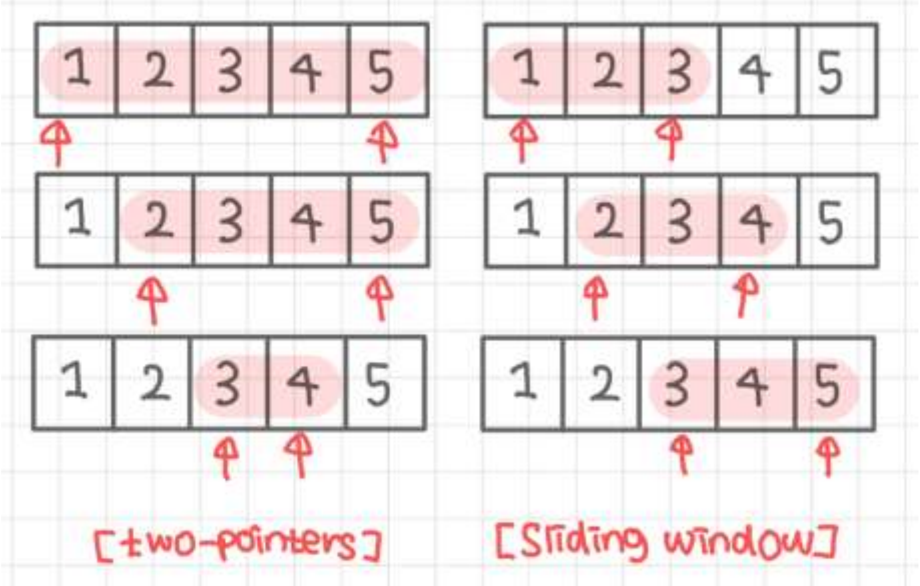
Table 9: Comparison result with Zhang.

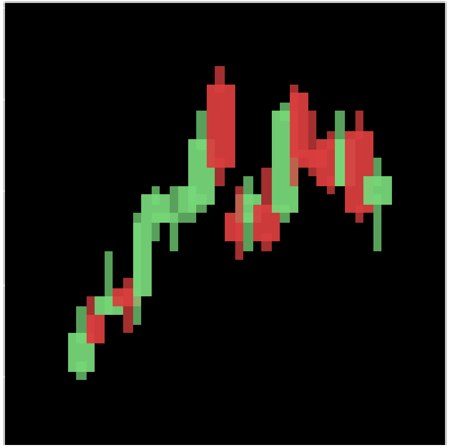
Hong Kong - Zhang		
	Accuracy	MCC
Zhang	61.7	0.331
Our	**92.6**	**0.846**

(Zhang, Zhang et al. 2018) input data is not only from historical stock trading data, a financial **news** and users sentiments from **social media** can be correlated to predict the movement in stock market.

Conclusions and Future Works 1

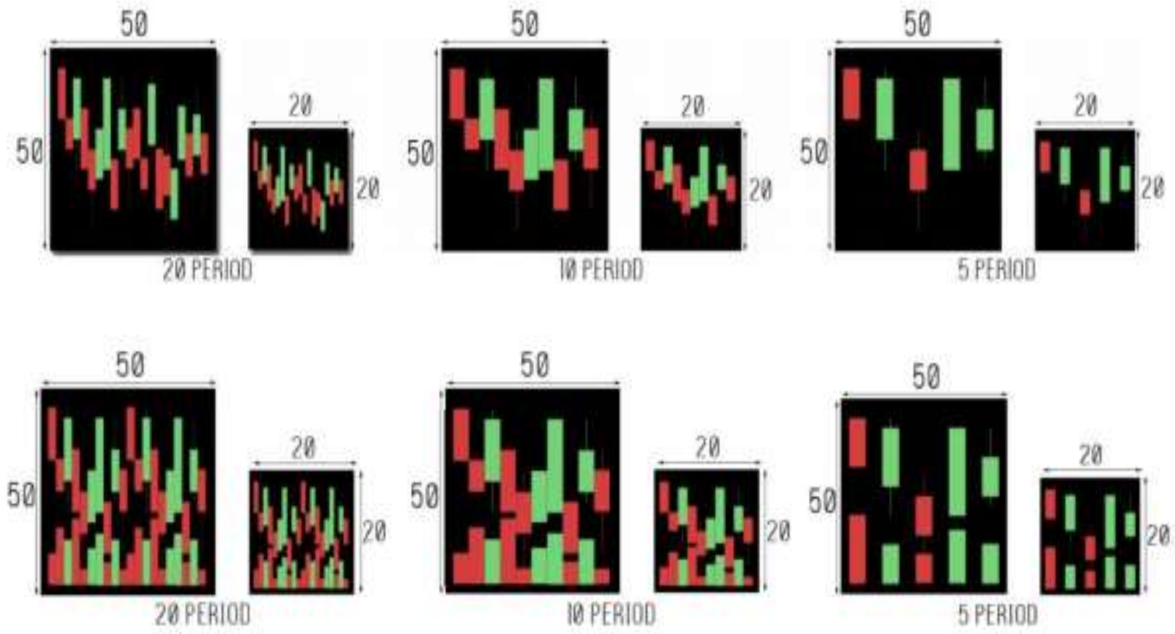
In this study, we present a new method for stock market prediction using 2 stock market datasets including 50 company stock markets for Taiwan50 datasets and 10 company stock market for Indonesian datasets. The first, we employ the sliding window technique to generate the period data. To find out correlation between enrich candlestick chart information and stock market prediction performance, we utilized the computer graphic technique to generate the candlestick chart images for stock market data. Finally, an CNN learning algorithm is employed to build our prediction for stock market.





Conclusions and Future Works 2

We found that the model using long-term trading days period with CNN learning algorithm achieves the highest performance of sensitivity, specificity, accuracy, and MCC. It is proved that Convolutional neural network **can find the hidden pattern inside the candlestick chart images** to forecast the movement of specific stock market in the future. Adding the indicator such as volume in candlestick chart not really help the algorithms increase finding the hidden pattern.



Conclusions and Future Works 3

The comparison experiments indicated that our proposed method provide highly accurate forecast for other datasets compare to the other existing methods. Patel used trading data from Reliance Industries, Infosys Ltd., CNX Nifty and S&P Bombay Stock Exchange BSE Sensex during 10 years with accuracy in the range of 89% - 92% while we achieved accuracy in the range of 93% - 97%. Khaidem method achieved the accuracy in the range of 86% - 94% using three trading data from Samsung, GE and Apple while we achieved in the range of 87% - 97%. Zhang utilized 13 different companies in Hong Kong stock exchange with accuracy 61%. Meanwhile, our method achieved 92% for accuracy.

Model Implementation

1. 캔들스틱 차트 그리기

```
# 캔들스틱 차트를 그리기 위해서 mpl_finance 이용합니다.
pip install mpl_finance
```

```
[ ] import os
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from pathlib import Path

    from shutil import copyfile, move
    from mpl_finance import candlestick2_ochl
```

```
[ ] def ohlc2cs(fname, seq_len, dimension):
        # python preprocess.py -m ohlc2cs -l 20 -i stockdatas/EWT_testing.csv -t testing
        print("Converting olhc to candlestick")
        symbol = fname.split('_')[0]
        print(symbol)
        df = pd.read_csv(fname, names=['Open','High','Low','Close','Volume','Change'])
        df.fillna(0)
        plt.style.use('dark_background')
        df.reset_index(inplace=True)

        figs = np.zeros((len(df)-1, dimension, dimension, 3))
        labels = []
        for i in range(0, len(df)-1):
            # ohlc+volume
            c = df.ix[i:i + int(seq_len) - 1, :]
            c_ = df.ix[i:i + int(seq_len), :]
            if len(c) == int(seq_len):
                my_dpi = 96
                fig = plt.figure(figsize=(dimension / my_dpi,
                                          dimension / my_dpi), dpi=my_dpi)
                ax1 = fig.add_subplot(1, 1, 1)
                candlestick2_ochl(ax1, c['Open'], c['Close'], c['High'],
                                  c['Low'], width=1,
                                  colorup='#77d879', colordown='#db3f3f')
                ax1.grid(False)
                ax1.set_xticklabels([])
```

```
                candlestick2_ochl(ax1, c['Open'], c['Close'], c['High'],
                                  c['Low'], width=1,
                                  colorup='#77d879', colordown='#db3f3f')
                ax1.grid(False)
                ax1.set_xticklabels([])
                ax1.set_yticklabels([])
                ax1.xaxis.set_visible(False)
                ax1.yaxis.set_visible(False)
                ax1.axis('off')

                # create the second axis for the volume bar-plot
                # Add a seconds axis for the volume overlay

            starting = c_["Close"].iloc[-2]
            endvalue = c_["Close"].iloc[-1]
            if endvalue > starting :
                label = 1
            else :
                label = 0
            labels.append(label)

            fig.canvas.draw()
            fig_np = np.array(fig.canvas.renderer._renderer)
            figs[i] = fig_np[:,:,:3]

            plt.close(fig)
            # normal length - end

        print("Converting olhc to candlestik finished.")
        return figs, labels
```

```
[ ] # 036570 : 엔씨소프트
    inputs = '036570_from_2010.csv'
    seq_len = 20
    dimension = 48

    figures, labels = ohlc2cs(inputs, seq_len, dimension)
```

```
[ ] #위 함수로 생성된 figures는 값의 범위가 0~255 이기 때문에 0~1로 맞춰주기 위해 255로 나눕니다.
    figures = figures/255.0
    print(np.shape(labels), np.shape(figures))
```

Model Implementation

2. 데이터 Generator 생성

```
def single_stock_generator(chart, labels, batch_size) :
    #output [chart, labels]
    while True :
        stock_batch = np.zeros(shape=(batch_size, dimension, dimension, 3))
        label_batch = np.zeros(shape=(batch_size, ))
        for i in range(batch_size) :
            idx = np.random.randint(len(labels))
            stock_batch[i] = chart[idx]
            label_batch[i] = labels[idx]

        yield stock_batch, label_batch
```

```
train_len = 1500
batch_size = 16
train_gen = single_stock_generator(figures[:train_len], labels[:train_len], batch_size)
test_gen = single_stock_generator(figures[train_len:], labels[train_len:], batch_size)
```

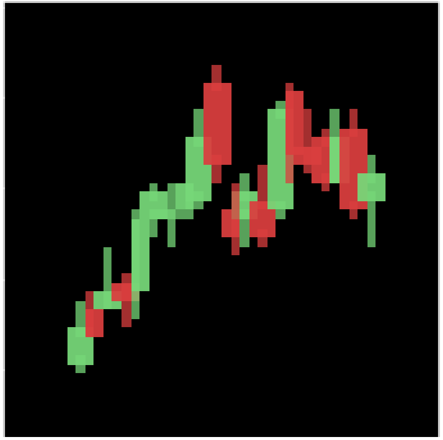
```
tmp_data = next(train_gen)
print("Chart image shape : ",np.shape(tmp_data[0]))
print("Label shape :",np.shape(tmp_data[1]))
```

```
Chart image shape :  (16, 48, 48, 3)
Label shape : (16,)
```

```
# 만들어진 차트 이미지 중 하나를 예시로 그려보겠습니다.
import matplotlib as mpl
import matplotlib.pylab as plt
%matplotlib inline
```

```
plt.figure()
plt.imshow(tmp_data[0][0][:,:,:])
plt.show()
```

```
plt.figure()
plt.imshow(tmp_data[0][0][:,:,:])
plt.show()
```



Model Implementation

3. 모델 작성

```
!pip install -q tensorflow-gpu==2.0.0-rc1
import tensorflow as tf
```

```
# Keras의 Functional APi
from tensorflow import keras
from tensorflow.keras import layers
```

```
# 논문에서 제시한 CNN 구조
# CNN의 filter size, dropout rate, padding 등은 임의로 지정
```

```
inputs = keras.Input(shape=(48, 48, 3))
x = inputs
x = layers.Conv2D(32, 3, activation='relu', padding="same")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(48, 3, activation='relu', padding="same")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Dropout(rate=0.5)(x)
x = layers.Conv2D(64, 3, activation='relu', padding="same")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(96, 3, activation='relu', padding="same")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Dropout(rate=0.5)(x)
x = layers.Flatten()(x)
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(rate=0.5)(x)
x = layers.Dense(1, activation='sigmoid')(x)
outputs = x

model = keras.Model(inputs, outputs)
model.summary()
```

```
Model: "model_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_16 (InputLayer)        [(None, 48, 48, 3)]       0
_________________________________________________________________
conv2d_66 (Conv2D)           (None, 48, 48, 32)        896
_________________________________________________________________
max_pooling2d_59 (MaxPooling (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_67 (Conv2D)           (None, 24, 24, 48)        13872
_________________________________________________________________
max_pooling2d_60 (MaxPooling (None, 12, 12, 48)        0
_________________________________________________________________
dropout_33 (Dropout)         (None, 12, 12, 48)        0
_________________________________________________________________
conv2d_68 (Conv2D)           (None, 12, 12, 64)        27712
_________________________________________________________________
max_pooling2d_61 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_69 (Conv2D)           (None, 6, 6, 96)          55392
_________________________________________________________________
max_pooling2d_62 (MaxPooling (None, 3, 3, 96)          0
_________________________________________________________________
dropout_34 (Dropout)         (None, 3, 3, 96)          0
_________________________________________________________________
flatten_27 (Flatten)         (None, 864)               0
_________________________________________________________________
dense_30 (Dense)             (None, 256)               221440
_________________________________________________________________
dropout_35 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_31 (Dense)             (None, 1)                 257
=================================================================
Total params: 319,569
Trainable params: 319,569
Non-trainable params: 0
_________________________________________________________________
```

Model Implementation

4. 훈련

```
[ ] num_iters = train_len // batch_size
    num_epochs = 10
```

```
def train_step(train_data_gen, test_data_gen, model) :
    optimizer = tf.keras.optimizers.Adam(0.0001)
    model = model
    loss_fn = tf.keras.losses.BinaryCrossentropy()
    num_test_iters = num_iters // 4
    for epoch in range(num_epochs) :
        epoch_loss_avg = tf.keras.metrics.Mean()
        val_loss_avg = tf.keras.metrics.Mean()

        for iter in range(num_iters) :
            x_batch, y_batch = next(train_data_gen)
            with tf.GradientTape() as tape :
                y_ = model(x_batch)
                loss_value = loss_fn(y_batch, y_)
                grads = tape.gradient(loss_value, model.trainable_variables)
            optimizer.apply_gradients(zip(grads, model.trainable_variables))
            epoch_loss_avg(loss_value)

        for iter in range(num_test_iters) :
            x_batch, y_batch = next(test_data_gen)
            y_ = model(x_batch)
            loss_value = loss_fn(y_batch, y_)
            val_loss_avg(loss_value)



        print("Epoch {:03d}: , Train Loss: {:.5f}".format(epoch, epoch_loss_avg.result()))
        print("Val_Loss: {:.3f}".format(val_loss_avg.result()))
```

```
[ ] train_step(train_gen, test_gen, model)
```

```
train_step(train_gen, test_gen, model)
```

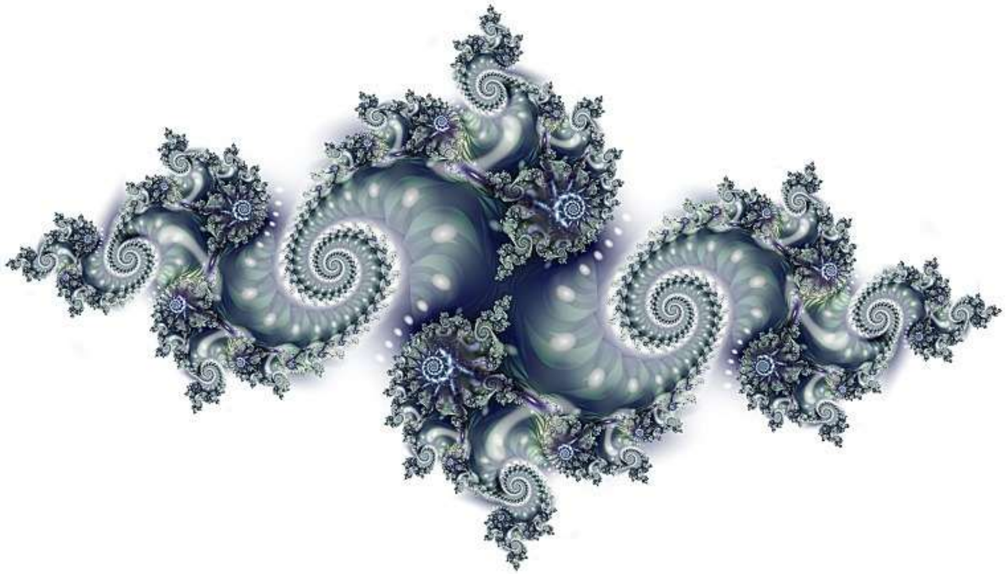
```
Epoch 000: , Train Loss: 0.69078
Val_Loss: 0.697
Epoch 001: , Train Loss: 0.69190
Val_Loss: 0.701
Epoch 002: , Train Loss: 0.69189
Val_Loss: 0.693
Epoch 003: , Train Loss: 0.69279
Val_Loss: 0.693
Epoch 004: , Train Loss: 0.69222
Val_Loss: 0.696
Epoch 005: , Train Loss: 0.69333
Val_Loss: 0.695
Epoch 006: , Train Loss: 0.69185
Val_Loss: 0.699
Epoch 007: , Train Loss: 0.69222
Val_Loss: 0.693
Epoch 008: , Train Loss: 0.69112
Val_Loss: 0.696
Epoch 009: , Train Loss: 0.69273
Val_Loss: 0.691
```

```
[1] y_
```

```
<tf.Tensor: id=5683842, shape=(2219, 1), dtype=float32, numpy=
array([[0.4786848 ],
       [0.47747847],
       [0.4801753 ],
       ...,
       [0.47713557],
       [0.47713557],
       [0.47713557]], dtype=float32)>
```

Conclusions

1	2	3	4	5	정확도
하나	둘	셋	넷		?
One	Two	Three	Four		?
i	ii	iii	iv		?



팀 연어유희