Summer Project
SnT Council
-IIT Kanpur-

Finance & Analytics Club

IMAGE PROCESSING IN FINANCE

Final Documentation

Saturday, 18 July 2020

Mentor :
Shaurya Jain

GenSec (SnT) :
Mudit Agarwal

Coodinators (FAC) :
Shubham Gupta
Jatin Khandelwal
Shaurya Jain
Sarim Haroon

By:
Gyanendra Kumar
Aman Yadav
Abhishek Verma
Riya Bhalla
Dhanya Agarwal
Divyanshi Agarwal
Sampada Sinha
Vaishnavi Gupta
Nitika Singhvi
Saurabh Gupta
Sarthak Gupta
Surabhi Anand
Apoorv Bansal
Aaryan Mehar
Aakash Rathi
Shakshi Chauhan
Alok Sharma

Contents

Acknowledgement

We are grateful to the **Science and Technology Council IIT Kanpur**, and **Finance And Analytics Club** for providing us with an opportunity and motivation to carry out this summer project. We would also like to express our gratitude to the coordinator of Finance And Analytics Club : **Shaurya Jain**, for guiding us throughout this project.

Thanks also to those numerous researchers whose work we've referred to in the citations. It is because of them that the field of **Image Processing** is getting noticed for its applications, and its scope to get rid of many human drudgery

Abstract

Computational intelligence techniques for financial trading systems have always been quite popular. In the last decade, deep learning models start getting more attention, especially within the image processing community.

In this study, we propose a novel algorithmic trading model CNN-TA using a 2-D Convolutional Neural Network based on image processing properties. In order to convert financial time series into 2-D images, 12 different technical indicators each with different parameter selections are utilized. Each indicator instance generates data for a 15 day period. As a result, 12x15 sized 2-D images are constructed. Each image is then labelled as Buy, Sell or Hold depending on the hills and valleys of the original time series.

The results indicate that when compared with the Buy Hold Strategy and other common trading systems over a long out-of-sample period, the trained model provides better results for stocks and ETFs.

1 Introduction

"Finance is not merely about making money. It's about achieving our deep goals and protecting the fruits of our labor. It's about stewardship and, therefore, about achieving a good society. "

– Robert J. Shiller

1.1 Overview

In this study, we proposed a novel approach that converts 1-D financial time series into a 2-D image-like data representation in order to be able to utilize the power of deep convolutional neural network for an algorithmic trading system. In recent years, deep learning based prediction/classification models started emerging as the best performance achievers in various applications, outperforming classical computational intelligence.However, image processing and vision based problems dominate the type of applications that these deep learning models outperform the other techniques.

Nowadays, deep learning methods have started appearing on financial studies. There are some implementations of deep learning techniques such as Recurrent neural network (RNN), convolutional neural network (CNN), and long short term memory (LSTM).CNNs have been by far, the most commonly adapted deep learning model. Meanwhile, majority of the CNN implementations in the literature were chosen for addressing computer vision and image analysis challenges

.

1.2 Problem Statement

Take the data of any company And then use machine learning models to predict the future returns using present time-series data. Basically the idea is we convert the times series data into a 2D image and then process it in our CNN model and analyse the profit.

2 Understanding from the Project

2.1 Algorithm Trading

Algorithm Trading is a process for executing orders utilizing automated and pre-programmed trading instructions to account for variables such as price, timing and volume. An algorithm is a set of directions for solving a problem. Computer algorithms send small portions of the full order to the market over time. Algorithmic trading makes use of complex formulas, combined with mathematical models and human oversight, to make decisions to buy or sell financial securities on an exchange. Algorithmic traders often make use of high-frequency trading technology, which can enable a firm to make tens of thousands of trades per second. Algorithmic trading can be used in a wide variety of situations including order execution, arbitrage, and trend trading strategies.

2.2 Risk and Return

Periodic Return is the percentage change in the value of an asset or investment, including reinvestment of income, from the beginning to the end of a period, assuming no contributions or disbursements

In finance, **risk** refers to the degree of uncertainty or potential financial loss inherent in an investment decision.It measures the volatility or standard deviation of returns.

2.3 Financial forecasting

Financial forecasting is the process by which a company thinks about and prepares for the future. Forecasting involves determining the expectations of future results.Forecasts can also help a company identify the assets or debt needed to achieve its goals and priorities.Forecasting helps a company's executive management determine where the company is headed. Calculating the financial impact of those forecasts is where financial modeling comes into play.

2.4 The stock market

The stock market refers to the collection of markets and exchanges where regular activities of buying, selling, and issuance of shares of publicly-held companies take place. Such financial activities are conducted through institutionalized formal exchanges or over-the-counter (OTC) marketplaces which operate under a defined set of regulations. There can be multiple stock trading venues in a country or a region which allow transactions in stocks and other forms of securities.

2.5 Technical analysis

Technical analysis is a trading discipline employed to evaluate investments and identify trading opportunities by analyzing statistical trends gathered from trading activity, such as price movement and volume. Technical analysis tools are used to scrutinize the ways supply and demand for a security will affect changes in price, volume and implied volatility. Technical analysis is often used to generate short-term trading signals from various charting tools.

2.6 Deep Learning

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

In deep learning, a **convolutional neural network (CNN, or ConvNet)** is a class of deep neural networks, most commonly applied to analyzing visual imagery.They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.

2.7 Python Libraries : NumPy and Pandas

This project also provided us a handy knowledge about the usage of various python libraries such as numpy , pandas , matplotlib ,tensorflow ,keras,PIL and many more .

NumPy stands for 'Numerical Python' or 'Numeric Python'. It is an open source module of Python which provides fast mathematical computation on arrays and matrices. Since, arrays and matrices are an essential part of the Machine Learning ecosystem, NumPy along with Machine Learning modules like Scikit-learn, Pandas, Matplotlib, TensorFlow, etc. complete the Python Machine Learning Ecosystem.

Similar to NumPy, **Pandas** is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Dataframe. It is like a spreadsheet with column names and row labels.

2.8 Data Cleaning and Pre-processing

The first problem faced while working with any time series data is cleaning and pre-processing .Hence , data cleaning is very essential before applying any mathematical model in order to get maximum output .We used following methods to clean our data :

- Dropping unnecessary columns in a DataFrame .
- Changing the index of a DataFrame as per our need.
- Using inbuilt Pandas and NumPy methods to clean dataset and replace NaN values with high negative values.
- Renaming columns to a more recognizable set of labels.

3 Method - Image Processing

3.1 Data Extraction

In our study, the daily stock prices of various firms or company are obtained from finance.yahoo.com for training,validation and testing purposes.Stock/ ETF prices between 01/01/2010 to 31/07/2019 are used for training,validaton and testing purposes.
We took stock prices between 01/01/2010 to 31/12/2016 for training our model and stock prices between 01/01/2017 to 31/03/2018 for validating our model.At last we took stock prices between 01/04/2018 to 31/07/2019 for predicting the results with the help of our model.

3.2 Labelling

Once the extraction of data is done , we move on to the labelling part where each stock is manually marked as Buy,Sell or Hold depending on the top and bottom point in the sliding window approach. In this approach the bottom point is marked as Buy since it is the least price encountered in the sliding window,the top point is marked as Sell to maximise the profit, whereas the rest are marked as Hold.Once the labelling is done we move on to the image creation part.The structure of the labelling process is given in Algorithm 1.

ALGORITHM: 1

```
def labelling(data):
    windowsize = 14
    counterrow = 0
    numberofdays = data.shape[0]
    result = np.array(data['result'])
    while(counterrow < numberofdays):
        counterrow = counterrow + 1
        if(counterrow > windowsize):
            windowbeginindex = counterrow - windowsize
            windowendindex = windowbeginindex + windowsize - 1
            windowmiddleindex = (windowbeginindex + windowendindex)/2
            minimum = max(data['Adj Close'])
            maximum = 0
            for i in range(windowbeginindex-1, windowendindex):
                number = data['Adj Close'].iloc[i]
                if(number < minimum):
                    minimum = number
                    minindex = i
                if(number > maximum):
                    maximum = number
                    maxindex = i

            result[minindex] = 1    ## 1 for buy  ,  0 for hold
            result[maxindex] = 2    ## 2 for sell
    return result
```

3.3 Image Creation

For each day a (12×15) image is generated by using 15 technical indicators and 15 different intervals of technical indicators. Meanwhile, each image uses the associated label ("Hold","Buy", or "Sell") with the sliding window logic provided in Algorithm 1. The order of the indicators is important,

since different orderings will result in different image formations.To provide a consistent and meaningful image representation, we clustered indicator groups (oscillator or trend) and similar behaving indicators together or in close proximity.We took the normalized value of Technical indicators and them convert them into array so that using PIL library we can create images and saved them in a respective folder of classes.

3.3.1 Technical Indicator

In image creation phase, for each day, RSI, Williams %R, WMA, EMA, ,Triple EMA, CCI, CMO, MACD, PPO, ROC, and PSI values for different intervals (6 to 20 days) are calculated using TA-lib library.Since 6 to 20 days of indicator ranges are used in our study, swing trades for 1 week to 1 month periods are focused. Different indicator choices and longer ranges can be chosen for models aiming for less trades.

3.3.2 Normalization

Since the values of each indicator varies significantly from each other.We normalized the value of all indicators so that there was no big difference in pixels of images. We normalized all the values of Technical Indicators between 0 to 1 using the percentile method.So for that we first extract the all technical indicators column from dataset and convert them into numpy array and sort them such that all distinct values are arranged in ascending order.Repeating Values are used only 1 time in constructing numpy array as there values remain same in all cells.The structure how we normalized values are listed in Algorithm 2

ALGORITHM: 2

```
def normalization(data,sort_array):
 unique = np.unique(sort_array, axis=0) # To create array of distinct values
 unique =unique.tolist()
 a=len(unique)
 for n in range(6,21):
  for i in range(0,1732):
    data.at[i,'rsi'+str(n)]=unique.index(data['rsi'+str(n)][i])/a
    data.at[i,'roc'+str(n)]=unique.index(data['roc'+str(n)][i])/a
    data.at[i,'sma'+str(n)]=unique.index(data['sma'+str(n)][i])/a
    data.at[i,'ema'+str(n)]=unique.index(data['ema'+str(n)][i])/a
    data.at[i,'wma' +str(n)]=unique.index(data['wma'+str(n)][i])/a
    data.at[i,'tema'+str(n)]=unique.index(data['tema'+str(n)][i])/a
    data.at[i,'william'+str(n)]=unique.index(data['william'+str(n)][i])/a
    data.at[i,'cci'+str(n)]=unique.index(data['cci'+str(n)][i])/a
    data.at[i,'cmo'+str(n)]=unique.index(data['cmo'+str(n)][i])/a
    data.at[i,'macd'+str(n)]=unique.index(data['macd'+str(n)][i])/a
    data.at[i,'ppo'+str(n)]=unique.index(data['ppo'+str(n)][i])/a
    data.at[i,'parabolicsar'+str(n)]=unique.index(data['parabolicsar'+str(n)][i])/a
 return data

df =normalization(df,sort_array)
```

As the value of all indicators are normalized between 0 to 1. Now we are in stage to create the images using PIL Library. Figure 1 illustrates sample 12×15 Pixel images that are created during

the image creation phase.

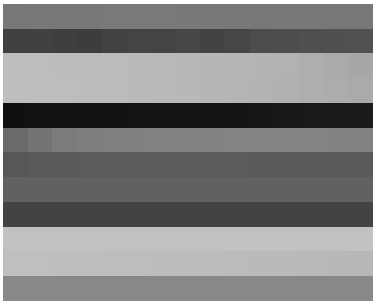


Figure 1: Buy

Figure 2: Sell

Figure 3: Hold

3.4 Data Augmentation

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. **Data augmentation** techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks.

In our Project ,We have a sample size of 2200 datasets only in which we have to train the model as well as Test the model.So using Data Augmentation is somewhat necessary for our model. The parameters we used are standard parameters for Data Augmentation. Using the more efficient value for parameters we can increase the efficiency of model significantly.The parameters we have used in our project are listed below with there values–

$$
\begin{aligned}
Rescale \ &= 1./255, \\
Featurewise\ Center &= True, \\
Width\ Shift\ Range &= 0.1, \\
Shear\ Range &= 0.15, \\
Zoom\ Range &= 0.15, \\
Horizontal\ Flip &= True, \\
Fill\ Mode &=' reflect',
\end{aligned}
$$

3.5 CNN

In the proposed algorithm the CNN model used for analysis phase consists of 9 layers namely the input layer (12×15), two convolutional layers ($10 \times 13 \times 32, 8 \times 11 \times 64$), a max pooling ($4 \times 5 \times 64$), two dropout (0.25, 0.50), fully connected layers (128), and an output layer (3).The convolutional layers is where the filters are applied.Using a 3×3 filter enables us to capture most details in our image.The max pooling layer reduces the number of parameters within the model thereby decreasing the time complexity of the model. Since the model has lots of parameters we use 2 dropout layers that helps prevent over-fitting. Then the model uses a fully connected layer (containing 128 neuron that use a non linear activation function) which takes the results from convolution/pooling layers and uses them to classify the images as Buy,Sell or Hold.

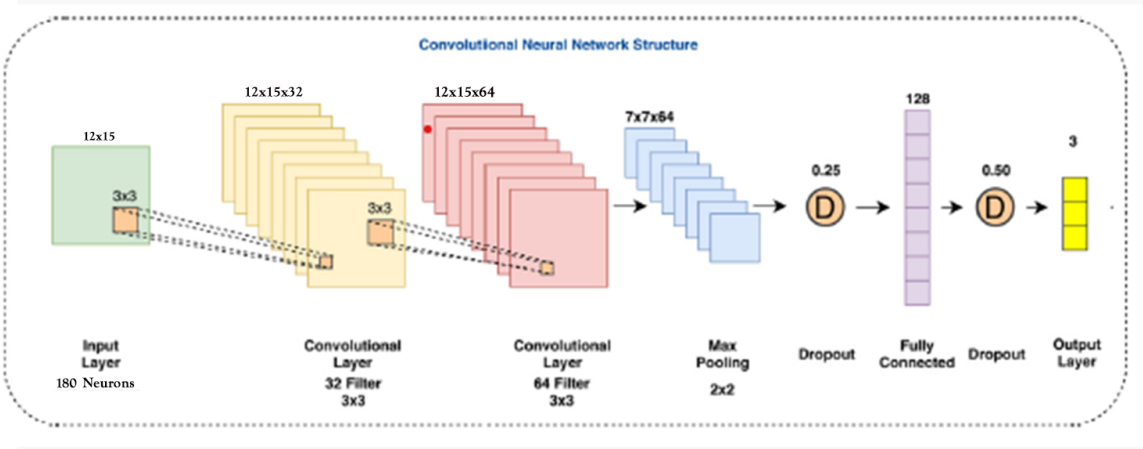


Figure 4: CNN Process

3.6 Performance Evaluation

The overall performance of our CNN model is evaluated using two different approaches i.e **Computational Model Performance and Financial Evaluation**.Computational model performance evaluation presents the convolutional neural network performance, i.e. how well the classifier distinguishes between Buy, Hold and Sell classes. Financial evaluation shows the performance of the whole proposed model by implementing the real world financial scenario. Stocks are bought, sold or held according to the predicted label with the actual stock prices.

3.6.1 Computational model performance

We then analyze the prediction performance of our proposed model. The data used for the testing purpose is of Reliance industrial limited. How well our model classifies our input images as actual labels can be accounted using Cost function. The cost function is a measure of "how good" a neural network did with respect to it's given training sample and the expected output.
Minimizing the Cost Function - A Cost minimization algorithm runs on the training data-set and adjusts the parameters of the model. Once the cost function is minimized for the training data-set, it should also be minimized for an arbitrary data-set if the relation is universal .i.e to say that the model should not have a bias for a particular data-set or the model should not show huge variance for an arbitrary data-set.

RMSprop algorithm is a good choice for minimizing the cost function in case of CNN model. Testing the model - The model prepared is then tested over the test set to check its correctness and efficiency.

$$Mean\ Square\ Error - 1.15$$

$$Accuracy - 0.42$$

$$F1\ Score - 0.31$$

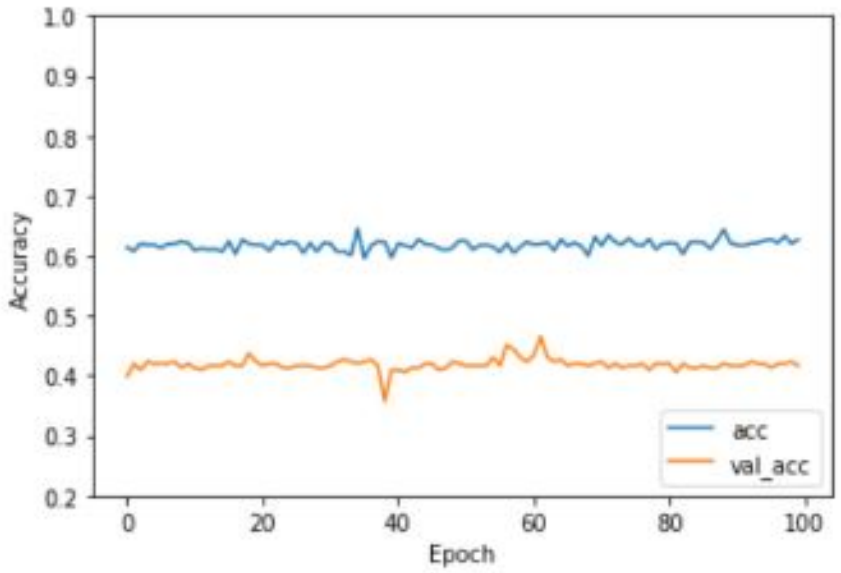


Figure 5: Accuracy Vs Epochs

3.6.2 Financial evaluation

This is the last step in our proposed algorithm where we analyze our model using financial evaluation method.The following proposed method used for financial evaluation maximizes the returns. In our model each stock is bought, sold or held according to the predicted label.

If the predicted label is "Buy" we buy the stock using all the current available capital (only if not bought before). If the predicted label is "Sell" we sell the stock (only if bought before). If the predicted label is "Hold" no action is taken. During a financial trade, if the same label comes consecutively, only the first label is activated and the corresponding transaction is performed. The next transaction is not performed until the label changes. The structure how we calculate return are listed in algorithm 3.

ALGORITHM: 3

```
df2['prediction']=labels_pred
icost=[],buy_l=[],cost=0,buy_c=0,buy=0,sell=0,win=0,loss=0

for i in df2.index:
    if(df2['prediction'][i]==0):
        if(buy_c<=0):
            buy_c=buy_c+1
            buy=buy+1
            cost=cost - df2['Adj Close'][i]
            x=df2['Adj Close'][i]
            buy_l.append(x)
            print("date : {}, cost : {},buy_c:{},price:{}".format(i, cost,buy_c,x),'Buy')
    if(df2['prediction'][i]==2):
        y=df2['Adj Close'][i]
        if(buy_c>=1):
            if(y-x>0):
                buy_c=buy_c-1
                sell=sell+1
                cost=cost+df2['Adj Close'][i]
                print("date : {}, cost :{},buy_c:{},price:{}".format(i, cost,buy_c,y),"Sell")
                if((buy>0) & (sell>0)):
                    if(y-x<0):
                        loss=loss+1
                    if(y-x>0):
                        win=win+1
                    icost.append(y-x)
    if(df2['prediction'][i]==1):
        continue
```

4 Future Aspects

Even though the performance of the model is promising, more improvements can still be achieved. The model might perform better-

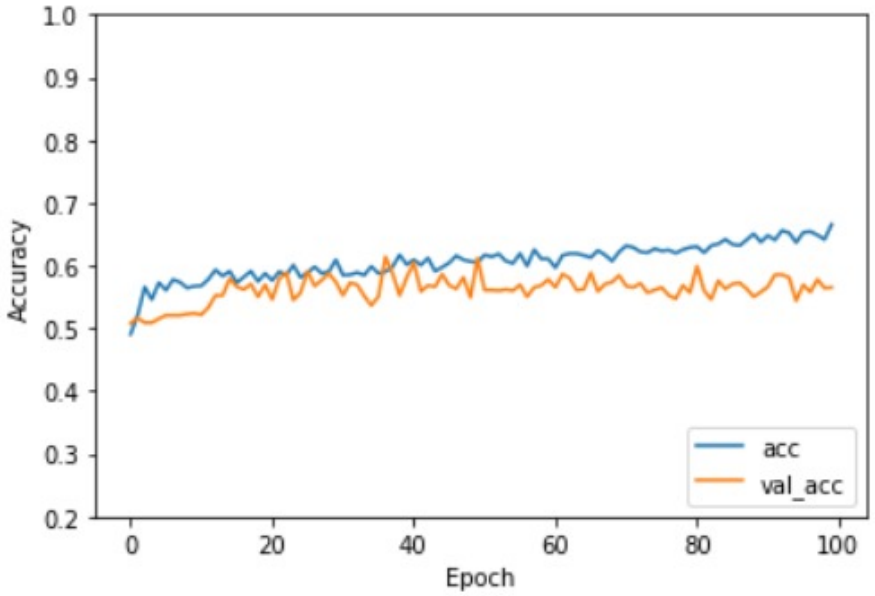
1. If CNN structural or hyper parameters are optimized more efficiently or effectively.
2. Evolutionary algorithms for model optimization may enhance the network performance.
3. Similarly, selecting the proper image size, window size, technical analysis optimization can improve the overall performance considerably.
4. Also, data representation for "Buy", "Sell", "Hold" points can be optimized for better trade signal generation performance.

In this study, we analyzed a long-only strategy for our algorithmic trading model. How-ever, adapting a long-short strategy might increase the profit significantly, since there are a lot idle times where the model is sitting on cash while waiting for a trigger Buy signal.

From a general perspective, more applications might start adapting 2-D CNN for non-image data in the future. There are already some indications for this trend for time series forecasting, gait recognition through radar signals and malware classification. Following these recent developments, in the near future, we might expect similar implementations in other fields to start utilizing image-based CNN

5 Problems Faced

- **Lack of Data** One of the major problem we faced is the lack of data, having more and more data is needed to prevent our CNN model from having bais towards a particular data-set and thus finding appropriate accuracy.
- **Difficulty** faced while finding correct parameters] While doing our research we came though a term Data augmentation but its very difficult to find the correct parameters.
- **Hypertuning** When we are trying to do hypertuning in our model, as it is the best technique to increase the validation or testing accuracy but it doesn't goes well in our's case. Although acuuracy increases almost 8-10% but model is not predicting the buy label, it only predicts the sell and hold pairs which results in 0 buy-sell pairs.



6 Conclusion

In this study, we utilized a 2-D Deep Convolutional Neural Network model to be used with financial stock market data and technical analysis indicators for developing an algorithmic trading system. In our proposed solution, we analyzed financial time series data and converted this data into 2-D images. In our study, we attempted to predict entry and exit points of the time series values as "Buy","Sell" and "Hold" marks for profitable trades. We used stock prices and ETFs as our Financial time series data. The results indicate this novel approach performs very well against Buy & Hold and other models over long periods of out-of-sample test periods. For future work, we will use more ETFs and stocks in order to create more data for the deep learning models. We will also analyze the correlations between selected indicators in order to create more meaningful images so that the learning models can better associate the Buy-Sell-Hold signals and come up with more profitable trading models.

7 Appendix

Relative Strength Index (RSI):

Relative Strength Index (RSI) is an oscillator type technical analysis indicator that shows the historical strength and weakness of stock prices. As stock prices change, RSI values oscilate between 0 and 100 which indicates whether the stock prices are in the overbought or oversold region.

Exponential Moving Average (EMA):

Exponential Moving Average (EMA) is a type of moving average indicator that shows moving average of the prices, emphasizing more for latest days. Latest data has more weight when calculating the moving average.

Weighted Moving Average (WMA):

Weighted Moving Average (WMA) is another type of moving average indicator that is the same as exponential moving average. The only difference is the importance of the close price is decreasing linearly when moving back to the past.

Hull Moving Average (HMA) :

Hull Moving Average (HMA) is a type of moving average indicator that reduces the lag associated with SMA. EMA and WMA tries the reduction of lag using more emphasis on the latest data.

Rate of Change (ROC):

Rate of Change is a technical indicator that illustrates the speed of price change over a period of time.

Moving Average Convergence and Divergence (MACD) :

Moving Average Convergence and Divergence (MACD) is a technical indicator that shows the trend of the stock prices. If MACD line crosses signal lines in upward direction, it is predicted that stock prices will increase. In contrast, if MACD line crosses signal lines in downward direction, it is interpreted that stock prices will decrease.

Percentage Price Oscillator (PPO) :

Percentage Price Oscillator (PPO) is similar to MACD.

Triple Exponential Moving Average:

Triple Exponential Moving Average (TEMA) is a type of EMA indicator that provides the reduction of minor price fluctuations and filters out volatility.

Commodity Channel Index (CCI) :

Commodity Channel Index (CCI) is an indicator that compares current prices and the average price over a period of time. It oscillates mostly (%75) between -100 and 100 values. %25 of time period, indicator passes its range values.

Chande Momentum Oscilator Indicator (CMO) :

The Chande Momentum Oscillator (CMO) is a type of momentum indicator that is similar to RSI and Stochastic Oscillator. It oscillates between -100 and 100. If the indicator value is over 50, it is interpreted that stock prices are in the overbought region. If the value is under -50, it is commonly considered that stock prices are in the oversold region.

Parabolic Sar :

Parabolic SAR (SAR) is a technical analysis indicator that is used to determine points of potential stops and reverses. Current SAR is calculated with three elements; Previous SAR (PSAR), Extreme point (EP) and Acceleration Factor (AF). Previous SAR is a SAR value for the previous period. EP is the highest high of the current uptrend or the lowest low of the current downtrend. AF explains the sensitivity of the SAR. AF begins at .02 and increases by .02 every time when EP rises in a Rising SAR. AF decreases by .02 every time when EP falls in a Falling SAR.

Williams %R :

Williams %R is a momentum based technical indicator that also determines overbought and oversold conditions for stock prices. It oscillates between -100 and 0 values. The corresponding logic for Williams %R is exactly the same as RSI. If the value is under -80, it is interpreted that stock prices are in the oversold region. In contrast, if the value is over -20, the stock price is considered to be in the overbought region.

8 References

- *Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach*
- *TA-Lib : Technical Analysis Library*
- *Tensorflow*
- *Keras*
- *Yahoo Finance*