Determining stock price direction by using CNN on 1-D time series data encoded as 2-D Images

Dikshant Dwivedi, Kshitij Kumar, Pranjali Jain, CSE, Bennett University, Greater Noida, India

Abstract— Analyzing and predicting the trends in the financial exchange has generally been a difficult undertaking. Over the past few decades, artificial intelligence has played a key role in the financial sector. There are many machine learning and deep learning algorithms utilized to tackle the problems of this sector. Deep learning algorithms, especially CNNs work extremely well with images. In this research, we have encoded time-series data to three types of 2-D images namely, normal, GAF, and MTF using different pre-processing techniques and then labeled them as buy, sell or hold. 2-D CNN model has been employed for the proposed research work. We have also used close-price as well as mid-price data for obtaining more data for analysis and observations. The results in terms of accuracy (highest – 59.9% for mid prices and 50.1% for close prices) and F1-score have been tabulated and compared with respect to different architectures, parameters, and image labeling strategies.

Keywords: Convolutional Neural Network, Gramian Angular Field, Markov Transition Field, Stock Market, Technical Indicators

1. Introduction

The stock market gives abundant opportunities to monetary establishments to make tremendous benefits with efficient investments. Stock price conduct is unpredictable and unexpected, resulting in a chaotic system and because of this, investing resources into financial exchanges can be very risky. It is a well-known fact that more than 90% of traders, whether novice or experienced, fail to make money in the stock market. Some of the reasons behind such a grim statistic are that the stock market is unpredictable, traders often lack knowledge of investment market cycles, letting emotions guide investment decisions and impatience to get rich quick. While some of these behaviors can be worked upon, there's still a lot of uncertainty surrounding the stock market. There's a lot of analysis required beforehand to make the right investment decision and one has to look for factors such as a stock's intrinsic value, price history, the current political climate, market sentiment, credit rating, reputation, etc. There are a lot of mathematical models devised for this task and people build their careers around mastering the intricacies of the market.

However, as automation takes over more and more industries, the appeal and benefits of machine-run algorithm trading are no longer restricted to financial markets. As a result, stock market data from 2021 show that machines are already performing over 80% of US trades. To the dismay of trading analysts, Artificial intelligence uses advanced statistical models to make rapid online trading resolutions, resulting in a trade that is centered more around sell-offs and short-term movements than on long-term outlook. When talking about Artificial intelligence, Deep learning based prediction/classification models have emerged as one of the best performers in a variety of applications in recent years, outperforming traditional unsupervised machine learning algorithms such as SVM. Their applications, however, in stock price prediction models have been extremely limited. This is a research area with the potential to yield favorable results. By far the most popular deep learning model has been CNNs because of their ability to extract relevant features out of data without any human intervention. CNNs are widely used in the image data arena because they perform exceptionally well on computer vision tasks such as image classification, object detection and so on.

In this paper, we have made an attempt to predict the direction of the close and mid-price of a stock using 2D-CNNs by converting a 1-D time series regression problem into a 2-D image classification problem. We have explored and compared various techniques that can be utilized to achieve the task along with all the challenges that we faced in the way.

2. Background Research

For a better understanding of this research paper, one needs to be aware of the following concepts and terminologies used.

2.1 Opening Price

The first price at which the stock trades at the beginning of a trading day on an exchange.

2.2 Closing Price

The last price at which the stock trades at the end of a trading day on an exchange. The closing price of a security is the standard benchmark used by investors to track its performance over time.

2.3 Highest/Lowest Price

The highest/lowest price at which a stock is traded on a given trading day.

2.4 Mid Price

The price that is halfway (average) between the bid (the highest price) and the ask (the lowest price) rates of a stock.

2.5 Convolutional Neural Network (CNN or ConvNet)

It is a variety of ANN used for analyzing visuals in DL. CNNs eliminate the need for heavy pre-processing of images when contrasted with other image classification methods. This implies that the network has the ability to learn and enhance filters automatically in contrast to traditional algorithms which use hand-engineered filters. The fact that feature extraction does not rely on prior knowledge or manual intervention is a massive benefit. They depend on the common-weight architecture of convolution channels, which slide along the input image to generate functions known as feature maps. A portion of applications includes classification, image/video recognition, segmentation, edge detection and generating brain-computer interfaces. A CNN unlike any other feedforward neural network has some hidden layers that perform the convolutions with a ReLU activation function. 3D volumes of neurons, Local connectivity, shared weights, and mechanism for pooling are some of the properties that, when combined, enable these networks to accomplish better generalization for larger visual datasets with lowered memory requirements.

2.6 Max Pooling

Max Pooling is a type of non-linear sampling rate reduction that is commonly utilized as a CNN layer to produce a feature map with the most prominent features. These features are then fed to the following layers leading to a progressive pruning in the number of parameters. This in turn leads to a lowered memory footprint and computation, thus controlling the problem of overfitting.

$$f_{X,Y}(S) = \max_{a,b=0}^{1} S_{2X+a,2Y+b} \qquad (1)$$

2.7 Batch Normalization

During the training of very deep neural networks, the distribution of inputs can be affected leading to destabilization of the network and slower learning. Batch normalization address this problem by normalizing (rescaling and recentering) the input data fed to subsequent layers.

2.8 Principle Component Analysis

When the dataset being used has a lot of features, the chances of overfitting become higher leading to a model that is not generalized to new data points. Such a problem can be tackled through principle component analysis that reduces the spatial dimensions. It does the same by projecting all the feature points onto only the first few key components and efficiently preserving distribution as much as possible. For calculating the K-th component in this case, the following equation is followed:

$$\widehat{\mathrm{X}}_k = \mathrm{X} - \sum_{s=1}^{k-1} \mathrm{X}\mathrm{w}_{(s)}\mathrm{w}_{(s)}^{\top}\,\mathrm{w}_{(s)}\mathrm{w}_{(s)}^{\top} \qquad (2)$$

Subsequently, the weight vector that is used to identify maximum variance from the newly generated data matrix is calculated using the below equation:

$$\mathrm{w}_{(k)} = \arg\max_{\|\mathrm{w}\|=1} \left\{ \|\widehat{\mathrm{X}}_k \mathrm{w}\|^2 \right\} = \arg max \left\{ \frac{w^{\top}\hat{x}_k^{\top}\hat{x}_k w}{w^T w} \right\} \qquad (3)$$

A kernel can also be utilized to perform PCA in a non-linear fashion. The technique developed as a result is capable of creating mappings that can maximize data variance and is called kernel PCA.

2.9 Gramian Angular Field (GAF) Images

These are images represented in the form of a Gramian matrix using a polar coordinate system. A time series represented in form of a GAF can demonstrate a correlation between each time point. Each element in the Gramian matrix is the trigonometric(cosine) sum between contrasting time intervals. GAF is represented as follows [25]:

$$G = \begin{bmatrix} \cos(\phi_1+\phi_1) & \cdots & \cos(\phi_1+\phi_n) \\ \cos(\phi_2+\phi_1) & \cdots & \cos(\phi_2+\phi_n) \\ \vdots & \ddots & \vdots \\ \cos(\phi_n+\phi_1) & \cdots & \cos(\phi_n+\phi_n) \end{bmatrix} \qquad (4)$$

where Φ_n represents the angular cosine of xi point of given time series X.

If we have a time series such that X = $\{x_1, x_2, ..., x_n\}$ consisting of 'n' number of observations, using equation 5, X is rescaled to make all the values in the range [-1,1].

$$\bar{x}_i = \frac{(x_i - \max(X) + (x_i - \min(X))}{\max(X) - \min(X)} \qquad (5)$$

After rescaling X, it is transformed into the polar coordinate system using equation 6:

$$\begin{gathered} \phi = \arccos(\tilde{x}_i), -1 \le \tilde{x}_i \le 1, \tilde{x}_i \in \tilde{X} \\ r = \frac{t_i}{N}, t_i \in \mathbb{N} \end{gathered} \qquad (6)$$

Finally, the GAF is calculated using equation 7.

$$\tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}'} \cdot \sqrt{I - \tilde{X}^2} \qquad (7)$$

where is $\tilde{X}$ the rescaled time-series data.

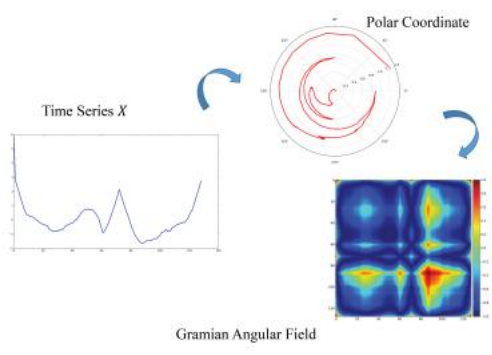

Fig. 1: Summarized process of encoding time series data to GAF images [25].

2.10 Markov Transition Fields (MTFs)

These images are based on the concept of first-order Markov transition probability along the first dimension and along the second dimension, they use temporal dependency.

Given a time series $X = \{x_1, x_2, ..., x_n\}$ of n real-valued observations, it is first discretized into Q quantile bins followed by calculation of Markov Transition Matrix, W and represented as follows [25]:

$$M = \begin{bmatrix} w_{ij} \mid x_1 \in q_i, x_1 \in q_j & \cdots & w_{ij} \mid x_1 \in q_i, x_n \in q_j \\ w_{ij \mid x_2 \in q_i, x_1 \in q_j} & \cdots & w_{ij \mid x_2 \in q_i, x_n \in q_j} \\ \vdots & \ddots & \vdots \\ w_{ij \mid x_n \in q_i, x_1 \in q_j} & \cdots & w_{ij \mid x_n \in q_i, x_n \in q_j} \end{bmatrix} \quad (8)$$

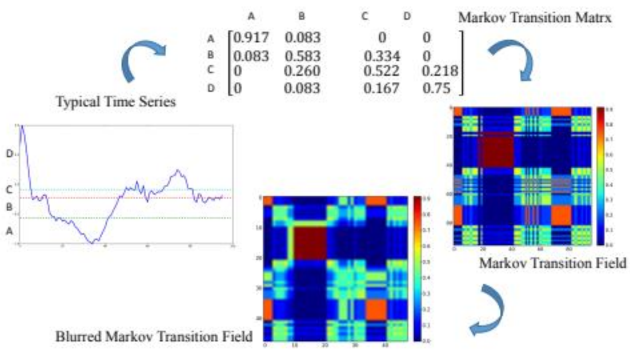

Fig. 2: Summarized process of encoding time series data to MTF images

2.11 The imbalanced class classification problem

This situation arises when the dataset is biased due to the reason that the number of elements in a dataset of one class is significantly higher than the combined number of elements of other classes. This problem is common in buy/sell/hold classification in stocks where the number of hold points is larger than buy and sell points together. Other common examples are fraud detection, spam detection, etc. The imbalanced class problem is still an active research area and some strategies that can be used to mitigate its effects to a very limited extent are resampling, ensemble learning, giving higher weights to underrepresented classes during training or randomly deleting samples with overrepresented classes.

3. Related Work

During the 1920s and 1930s, J. Walker and G. U Yule thought of the first real application of autoregressive models to the time series information [1]. This implementation in the later years served as a backbone for stock price forecasting using artificial intelligence. In the early years, classical regression models were being employed for this task such as linear regression, polynomial regression, etc. One such example is an autoregressive model proposed by Mr Krishna Charlapall, and Dr P. K. Sahoo to anticipate the future cost of stock in 2007 [2]. To improve accuracy, traditional statistical models utilized in technical analysis like Simple Moving Average (SMA), Exponential Moving Average (EMA), Moving Average Convergence Divergence (MACD), etc. are also employed.

Nowadays, non-linear ML algorithms like SVM and ANN are utilized since variance fundamental to the movement of stocks and other resources make linear methods inadequate. This makes sense since stock time series data has a non-stationary character. SVM, proposed by Vapnik [3], is a perfect candidate in this domain because of better-generalised performance, its effectiveness in high dimensional spaces, resistance to overfitting and the tendency to find a global optimum. In another example, R.C. Cavalcante et al. [4] employed SVM, ANN, ensemble methods as well as hybrid mechanisms for financial forecasting. Saahil Madge, in his paper, utilized an SVM model with RBF kernel on cost information through the Great Recession and ensuing recovery period for forecasting and achieved significant notable accuracies with certain criteria in the long haul. [5].

Artificial neural network (ANN) due to its fault tolerance, adaptability, universal function approximation, robustness, parallel data processing, and ability to learn and generalize is favoured over other models for stock market prediction [6], [7]. For predicting prices and dealing in the Taiwan Stock Index, Chen et al. [8] concocted a neural network model in 2003. Similarly, O.B. Sezer et al. [9] used technical indicators along with ANN for predicting turning points of Dow30 stock prices. Researchers have also applied various MLP configurations based on learning rate, the number of neurons, hidden layers for forecasting stock prices [10], [11]. In another study, Zabir Haider Khan, Tasnim Sharmin Alin, and Md. Akter Hussain developed a backpropagation approach for accurately training neural networks and multilayer feedforward networks on Bangladesh stock exchange data. [12]

In recent years, models combining genetic and evolutionary optimization techniques to predict stock prices and index values have also come up [13], [14]. Following these studies, Y.-K. Kwon et al. [15] in 2007, came up with an RNN model combined with GA optimization whereas, in 2017 O.B. Sezer et al. [16] proposed a deep MLP approach with GA enhancement. Mabu et al. [17] proposed a mechanism in which MLP was combined with rule-based evolutionary algorithms to decide buy/sell focuses on stock prices.

Additionally, Significant growth in computational prowess and the availability of large datasets has acted as a catalyst for the growth of Deep Neural Networks, leading to new approaches for financial time series analysis. One of these implementations uses extracting texts from news, the internet, and social media which was proposed by Ding et al. [18]. In 2017, T. Fischer et al. [19] utilized LSTM for predicting the movement direction for stocks of S&P 500. In the same year, on a similar dataset, C. Krauss et al. [20] made comparisons among random forests, deep neural nets, and gradient boosted trees. In another study by O.B. Sezer et al. [21], a feed-forward neural network was trained using technical indicators which were followed by optimization using evolutionary algorithms and it was concluded that deep learning models can learn well and provide satisfactory results for buy/sell points for an individual stock.

In general practice, CNNs deal with image classification and analytical problems. Even though they give remarkable results in the branch of computer vision, they are not preferred in the analysis of time series data directly. Deep learning algorithms like LSTM and RNN models are the ones that are most preferred for this purpose. Also, the dependence of algorithmic dealing frameworks on technical indicators has led to the integration of these indicators with deep NN which is still uncommon in writing. In 2018, O.B. Sezer et al. [22] came with a novel idea of using CNN with a 2-D grid portrayal of the technical indicators. For this purpose, they utilized 15 technical indicators with fifteen distinct parameters and time periods. The image conversion of technical indicators combined with other data implicitly converted the regression problem into a classification problem. On the basis of closing price, they classified images (data) into buy, sell and hold points. For this research, they used the data of Dow 30 stock prices. A graph-embedding layer-based LSTM technique for predicting stock costs was proposed in [23]. To determine stock developments, many STIs were employed to train the LSTM model. The approach was computationally faster; nevertheless, the prediction accuracy for denser graphs needs additional refinement. For stock cost prediction based on standard indicators, a CNN-LSTM algorithm was proposed. This approach used a sequence array of recorded data as the CNN's feedback image and feature matrices taken from the CNN as the LSTM's feedback vector. A previous study discovered that the CNN-LSTM model predicts stock prices with the greatest accuracy when contrasted with other DL approaches such as RNN, LSTM, CNN and MLP [24]. Inspired by the progress of DL in the branch of speech recognition and computer vision, Z. Wang and T. Oates [25] made an attempt to encode time-series information using various kinds of images, namely, MTF, GAF, and a combination of both i.e., MTF + GAF. They applied 12 datasets and tiled CNNs to understand high-level characteristics using individual GAF, MTF, and a combination of GAF+MTF. Upon analysis of these high-level features, they found that their methodology delivered significant results contrasted with the condition of state-of-art methods.

4. Proposed methodology

For the purpose of our study, we acquired the Bajaj Finance daily stock prices dataset from Kaggle. This dataset spanning from the year 2002 to the year 2020 contains open, high, low and close prices of the company's stock as well as its total volume traded. We have tried to predict the direction of close and mid prices of this dataset by encoding its 1D time-series input into 2D images and then classifying those images into "Buy" "Sell" and "Hold" Labels using some labelling strategy. To achieve this task, here is the methodology we proposed: Data preparation, labelling strategy, feature engineering, conversion to image, model creation, training and evaluation.

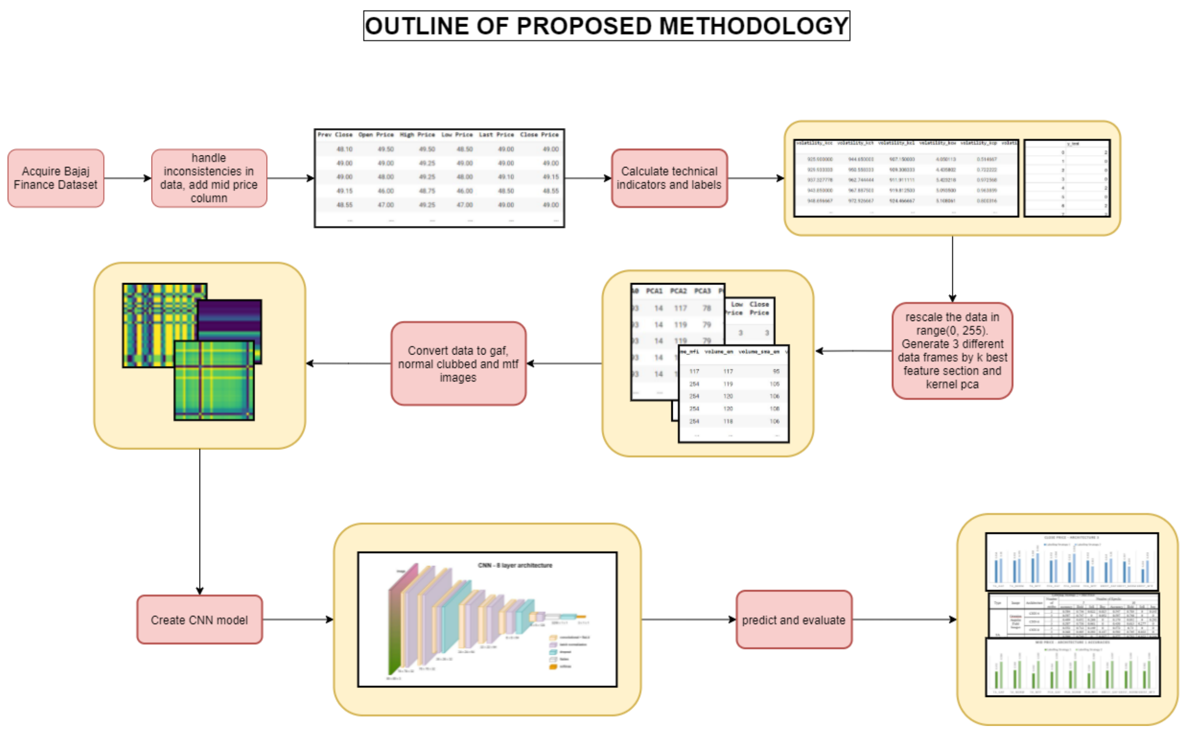

Fig. 3. Outline of proposed methodology

4.1 Data Preparation

The raw dataset acquired is probed for missing values, redundancies, noise, outliers, other inconsistencies, etc. and is dealt with by using appropriate methods like removing rows, replacing with mean/median and imputing values. An additional column for the mid prices is calculated using corresponding High and Low prices columns and is then added to the data frame. Since the dataset spans around 18 years, the variance in the prices was observed to be very high. The prices in the earlier years were too low compared to those in the later years. This posed a problem in calculating labels because a window for the "hold" class had to be decided which would give consistent results for all values. To tackle this problem, all the samples before July 2012 were slashed.

4.2 Labelling Strategies

We trained our models and compared their accuracies using two labelling strategies for both close and mid prices. For labelling strategy 1, we used the following algorithm. The hold window size of 12 was decided by a hit and trial process that gave us an even number of ratios for all three labels.

```
Labels := []
holdWindowSize := 12
for rowCounter := 0 to noOfDays - 2 do:
    priceOnCurrDay := prices[rowCounter]
    priceOnNextDay := prices[rowCounter + 1]
    changeInPrice := priceOnNextDay - priceOnCurrDay
    if changeInPrice > holdWindowSize then:
        Labels[rowCounter] := Buy
        end if
    else if changeInPrice < holdWindowSize then:
        Labels[rowCounter] := Sell
        end else if
    else do:
        Labels[rowCounter] := Hold
        end else
end for
```

Fig. 4. Pseudo code for labelling strategy 1

For labelling strategy 2, we used the same algorithm proposed by O.B. Sezar et al. in their paper [20]. The only modification we made was for a window size of 3. It is to be noted that for any real-world implementation, the window size should be much higher. For larger window sizes, the class imbalance was too high to perform any real tasks. Even for a window size of 3, the number of "Hold" labels was 4 times more than that of "Buy" and "Sell" labels. The pseudo-code for the algorithm is given in the figure below

```
Labels := []
windowSize := 3
while rowCounter < noOfDays do:
    if rowCounter >= windowSize - 1 then:
        windowBegin := rowCounter - (windowSize - 1)
        windowEnd := rowCounter
        windowMiddle := (windowBegin + windowEnd)/2
        for i := windowBegin to windowEnd do:
            currPriceInWindow := prices[i]
            if currPriceInWindow < minPrice then:
                minPrice := currPriceInWindow
                minPriceIndex := i
            end if
            if currPriceInWindow > maxPrice then:
                maxPrice := currPriceInWindow
                maxPriceIndex := i
            end if
        end for
        if maxPriceIndex == windowMiddle then:
            Labels[windowMiddle] := Sell
        end if
        else if minPriceIndex == windowMiddle then:
            Labels[windowMiddle] := Buy
        end if
        else do:
            Labels[windowMiddle] := Hold
        end else
    end if
    rowCounter := rowCounter + 1
end while
```

Fig. 5. Pseudo code for labelling strategy 2

4.3 Feature Engineering

In order to have enough data points while converting to images, more features are engineered from the existing ones. For financial time series data, there are several statistical indicators that are used in traditional technical analysis to get useful insights about a company's shares. So, we used our existing data to come up with three different data frames. They are described down below with the names that we are going to refer them with:

TA_NORMALIZED_DATA: For this data frame, we used the open, high, low, close and volume values from the original dataset and fed them as arguments to a python Technical Analysis(TA) Library. This library covers a wide range of momentum, volume, volatility, trend and other indicators. As a result, from a data frame of 5 dimensions, we ended up with a data frame of 88 dimensions. Then for the purpose of further processing and conversion of this data into images, all the values were normalized and rescaled in the range of unsigned integers from 0 to 255.
However, with the above data frame, there was a concern about the curse of dimensionality. So, in the other data frames, these concerns were addressed using kernel PCA and k best feature selection techniques.

PCA_NORMALIZED_DATA: A linear kernel was employed with PCA on the normalized technical indicators returned by the TA library to get a data frame with reduced dimensionality which was then combined with the original normalized data frame value. This resulted in a data frame of 23 dimensions.

30_K_BEST_TA_NORMALIZED_DATA: The K best feature selection strategy was applied on the calculated technical indicators returned by the TA library to select 30 features that contributed the most towards the target variable. For this purpose, we used SelectKBest(chi2, k=30) function from the scikit-learn library in python, where k is the required number of best features and chi2 is the Chi-Square test used for determining relationships between features.

$$\chi^2_c = \sum \frac{(O_i - E_i)^2}{E_i}$$

where:

$c =$ degrees of freedom

$O =$ observed value(s)

$E =$ expected value(s)

4.4 Conversion to image

We then prepared three sets of images, NORMAL_CLUBBED_IMAGES, GAF and MTF for each combination of three data frames (TA_NORMALIZED_DATA, PCA_NORMALIZED_DATA, 30_K_BEST_TA_NORMALIZED_DATA), two labelling strategies (Labelling Strategy 1, Labelling Strategy 2) and price types (mid-price, close price). Those three sets are mentioned before:

NORMAL_CLUBBED_IMAGES: For this purpose, we simply, clubbed all the values in a row of the sample and converted them into a PIL image.

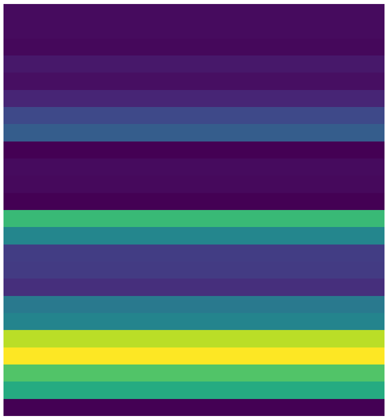

Fig. 6. Normal Image

GAF and MTF: In order to convert to GAF, we used **Gramian Angular Field** class from pyts module in python. Similarly, from the same module, the data was encoded to MTF using **Markov Transition Field** class. Implementation in both the libraries was done by referring to the same methods proposed by [23], which are also already covered briefly above in the background research section.

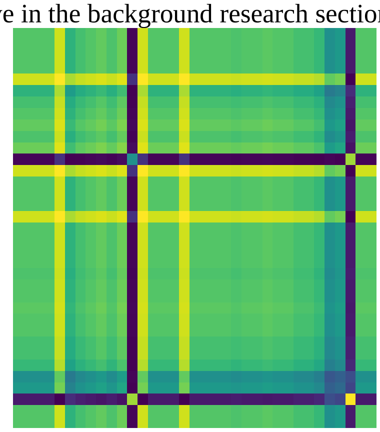

Fig. 7. GAF

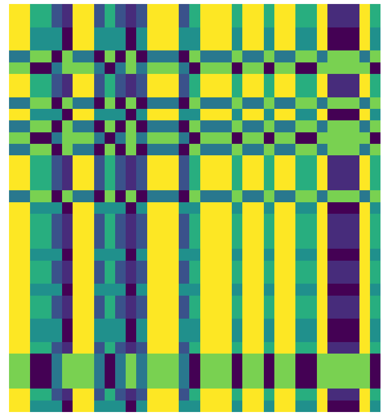

Fig. 8. MTF

All three sets of images were resized to 80 × 80 dimensions before being fed to the models for training.

4.5 Model creation, training and evaluation

The predictor images and target labels were divided into an 80:20 ratio of training/testing sets. The images were then trained and evaluated on 3 different architectures of 2D CNN to accommodate all possible scenarios of overfitting and underfitting. The three architectures here are referred to as CNN_4 (ARCH1), CNN_6(ARCH2) and CNN_8(ARCH3). All the architecture configurations are detailed in the figures below. Apart from the already discussed max pooling and batch normalization layers, there's also a dropout layer employed in CNN_8 and CNN_6 with a dropout ratio of 0.4 to ensure minimal overfitting. All three architectures had a learning rate of 0.001, used adam optimizer which is the industry standard for multi-class classification, sparse categorical cross-entropy as the loss function and ReLU as the activation function.

All the models were trained for 5 and 10 epochs. Each time, the stride values for one of the layers was changed from two to three. For evaluation, we used the accuracy and F1 score metrics.

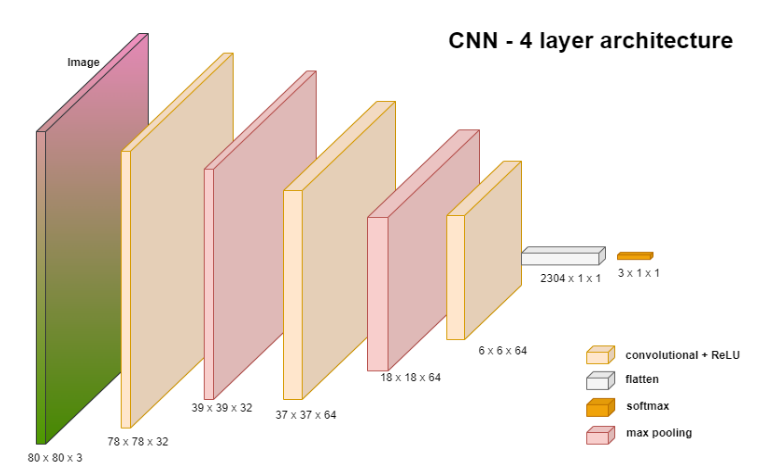

Fig. 9. CNN-4 layered Architecture

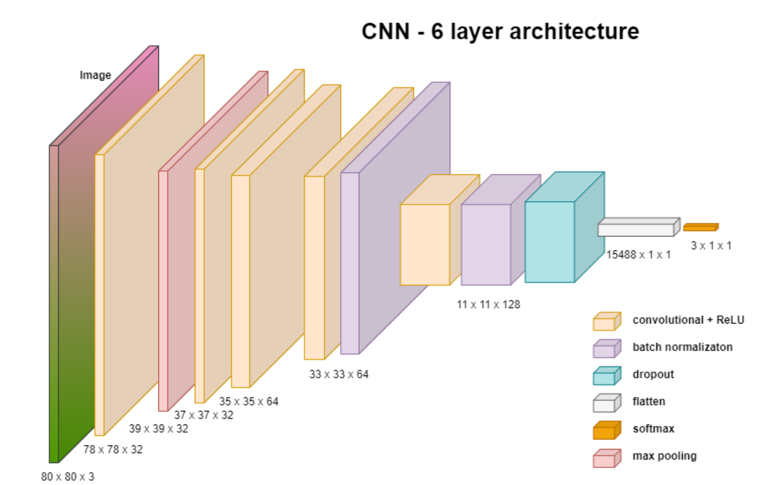

Fig. 10. CNN-6 layered Architecture

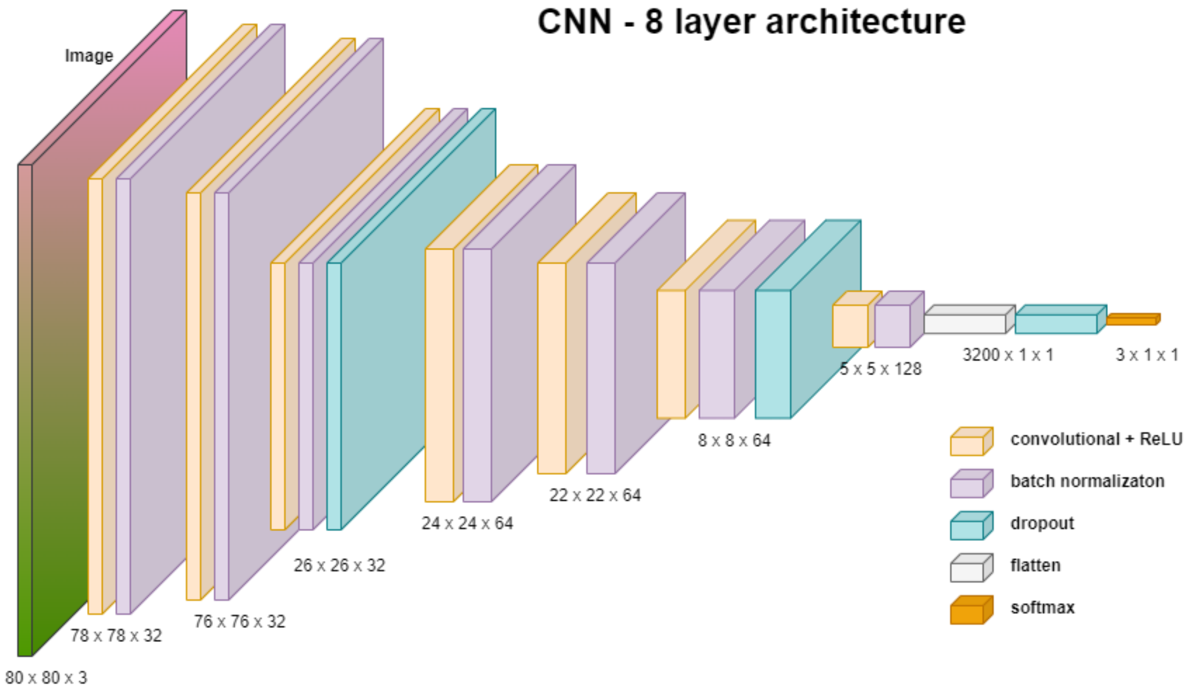

Fig. 11. CNN-8 layered Architecture

5. Observation and Result

Table 1. Accuracies and F1-scores for different data pre-processing techniques, images, number of strides and architectures using Labeling Strategy 1 and Close Price

Labeling Strategy 1 – Close Price											
Type	Image	Architecture	Number of strides	Number of Epochs							
				5				10			
				Accuracy	Hold	Sell	Buy	Accuracy	Hold	Sell	buy
TA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.392	0.131	0.2	0.539	0.377	0	0	0.549
			3	0.302	0.308	0.209	0.358	0.379	0	0.19	0.533
		CNN-6	2	0.334	0.47	0.015	0.252	0.374	0.017	0	0.545
			3	0.229	0.288	0.239	0.099	0.334	0	0.501	0
		CNN-8	2	0.354	0.197	0.156	0.497	0.294	0.164	0.336	0.344
			3	0.392	0.078	0.165	0.541	0.357	0.049	0.506	0.171
	Markov Transition Field Images	CNN-4	2	0.359	0.017	0.346	0.47	0.394	0.078	0.335	0.528
			3	0.357	0.145	0.262	0.48	0.349	0.201	0.236	0.467
		CNN-6	2	0.344	0	0.405	0.399	0.379	0.306	0.014	0.533
			3	0.384	0.017	0.029	0.556	0.379	0	0	0.55
		CNN-8	2	0.384	0.341	0.385	0.418	0.374	0.255	0.42	0.406
			3	0.387	0.17	0.164	0.529	0.377	0.219	0.428	0.4
	Normal Images	CNN-4	2	0.374	0	0.361	0.492	0.374	0	0.054	0.542
			3	0.374	0	0.146	0.534	0.369	0.017	0.149	0.527
		CNN-6	2	0.352	0	0.449	0.353	0.359	0.112	0.467	0.287
			3	0.334	0	0.501	0	0.339	0.016	0.503	0.084

		CNN-8	2	0.352	0.017	0.503	0.136	0.372	0	0.042	0.54
			3	0.392	0	0.167	0.553	0.374	0	0.014	0.545
PCA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.374	0	0.211	0.521	0.377	0.017	0.249	0.517
			3	0.337	0.094	0.287	0.462	0.379	0	0.015	0.548
		CNN-6	2	0.302	0.437	0.252	0.058	0.334	0	0.502	0
			3	0.314	0.342	0.419	0	0.367	0.045	0.251	0.514
		CNN-8	2	0.339	0.339	0.113	0.443	0.367	0.451	0.232	0.354
			3	0.347	0.441	0.262	0.283	0.354	0.017	0.342	0.465
	Markov Transition Field Images	CNN-4	2	0.367	0.09	0.212	0.513	0.342	0.15	0.329	0.442
			3	0.342	0.099	0.265	0.463	0.372	0.095	0.237	0.514
		CNN-6	2	0.384	0.017	0.015	0.552	0.357	0.245	0.496	0.051
			3	0.302	0.384	0.307	0.124	0.296	0.421	0.19	0.082
		CNN-8	2	0.352	0.157	0.043	0.404	0.327	0.216	0.349	0.379
			3	0.344	0.159	0.224	0.484	0.362	0.157	0.31	0.467
	Normal Images	CNN-4	2	0.379	0.017	0.015	0.55	0.379	0.017	0.043	0.564
			3	0.382	0.083	0.46	0.429	0.364	0	0.209	0.509
		CNN-6	2	0.349	0.08	0	0.521	0.354	0.233	0	0.512
			3	0.337	0.459	0.247	0.108	0.317	0.014	0.12	0.477
		CNN-8	2	0.322	0.295	0.407	0.199	0.352	0.306	0.407	0.335
			3	0.327	0.335	0.21	0.391	0.374	0	0.348	0.487
30 Best TA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.379	0.141	0.32	0.486	0.347	0	0.275	0.494
			3	0.359	0.214	0.211	0.486	0.362	0	0.122	0.519
		CNN-6	2	0.344	0.084	0.422	0.371	0.387	0.349	0	0.513
			3	0.312	0.395	0	0.337	0.334	0.078	0.279	0.452
		CNN-8	2	0.324	0.058	0.446	0.25	0.344	0.397	0.354	0.268
			3	0.359	0.169	0.2	0.485	0.349	0.124	0.222	0.482
	Markov Transition Field Images	CNN-4	2	0.349	0.205	0.265	0.466	0.344	0.207	0.346	0.413
			3	0.392	0.128	0.431	0.462	0.339	0.134	0.267	0.465
		CNN-6	2	0.377	0.089	0	0.545	0.379	0	0	0.55
			3	0.284	0.445	0	0.024	0.314	0.437	0.25	0.084
		CNN-8	2	0.307	0.39	0.263	0.214	0.349	0.328	0.234	0.434
			3	0.384	0.243	0.402	0.44	0.334	0.321	0.335	0.346
	Normal Images	CNN-4	2	0.342	0.097	0.15	0.484	0.352	0	0.398	0.425
			3	0.294	0.077	0.285	0.371	0.379	0	0	0.55
		CNN-6	2	0.334	0	0.501	0	0.374	0.033	0	0.543
			3	0.334	0	0.501	0	0.357	0	0.494	0.235
		CNN-8	2	0.337	0.174	0.417	0.32	0.324	0.147	0.288	0.432
			3	0.394	0.17	0.404	0.491	0.369	0	0.458	0.393

Table 2. Accuracies and F1-scores for different data pre-processing techniques, images, number of strides and architectures using Labeling Strategy 2 and Close Price

Labeling Strategy 2 – Close Price											
Type	Image	Architecture	Number of strides	Number of Epochs							
				5				10			
				accuracy	Hold	Sell	Buy	Accuracy	Hold	Sell	buy
TA Normalized	Gramian Angular	CNN-4	2	0.491	0.66	0.038	0	0.499	0.667	0	0.054
			3	0.469	0.635	0.053	0.084	0.476	0.65	0.04	0.016

Data	Field Images	CNN-6	2	0.259	0	0	0.412	0.259	0	0.323	0.368
			3	0.496	0.663	0	0	0.275	0.049	0.192	0.397
		CNN-8	2	0.38	0.539	0.155	0.211	0.385	0.545	0	0.2
			3	0.426	0.588	0.108	0.183	0.426	0.599	0	0.166
	Markov Transition Field Images	CNN-4	2	0.499	0.666	0	0	0.491	0.662	0	0
			3	0.446	0.615	0.167	0.146	0.458	0.625	0.179	0.152
		CNN-6	2	0.436	0.601	0.246	0	0.292	0.374	0.301	0.034
			3	0.395	0.539	0.225	0.22	0.365	0.513	0.268	0
		CNN-8	2	0.466	0.635	0.056	0.049	0.406	0.552	0.083	0.291
			3	0.461	0.633	0.145	0.034	0.368	0.498	0.179	0.267
	Normal Images	CNN-4	2	0.501	0.667	0.04	0	0.501	0.668	0	0
			3	0.501	0.668	0	0	0.481	0.655	0.051	0
		CNN-6	2	0.499	0.656	0.205	0	0.438	0.614	0.056	0.092
			3	0.368	0.51	0.283	0.019	0.496	0.659	0.123	0
		CNN-8	2	0.378	0.515	0.304	0	0.421	0.596	0.036	0.151
			3	0.277	0.208	0.384	0.018	0.433	0.61	0.039	0.214
PCA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.499	0.662	0.106	0.037	0.494	0.661	0.02	0
			3	0.491	0.66	0.02	0.035	0.476	0.646	0.038	0
		CNN-6	2	0.239	0	0.386	0	0.418	0.562	0.201	0.171
			3	0.297	0.194	0.076	0.408	0.259	0.02	0.019	0.411
		CNN-8	2	0.368	0.518	0.185	0.168	0.393	0.545	0.156	0.269
			3	0.431	0.599	0.05	0.184	0.484	0.651	0.074	0.12
	Markov Transition Field Images	CNN-4	2	0.479	0.647	0.071	0.052	0.491	0.655	0	0.071
			3	0.489	0.648	0.079	0.122	0.474	0.64	0.083	0.15
		CNN-6	2	0.262	0.039	0.122	0.403	0.499	0.666	0	0
			3	0.37	0.525	0.283	0	0.335	0.428	0.309	0.174
		CNN-8	2	0.416	0.548	0.269	0.271	0.421	0.564	0.177	0.263
			3	0.388	0.533	0.224	0.214	0.413	0.581	0.109	0.128
	Normal Images	CNN-4	2	0.501	0.668	0	0	0.496	0.663	0	0
			3	0.486	0.656	0	0.036	0.501	0.668	0	0
		CNN-6	2	0.446	0.614	0	0.223	0.418	0.567	0.333	0.09
			3	0.272	0.095	0.297	0.369	0.413	0.585	0.079	0.154
		CNN-8	2	0.458	0.622	0.116	0.129	0.388	0.534	0.257	0.129
			3	0.438	0.579	0	0.373	0.438	0.579	0	0.373
30 Best TA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.486	0.655	0	0	0.499	0.666	0	0
			3	0.471	0.632	0.088	0.206	0.494	0.657	0.021	0.068
		CNN-6	2	0.234	0.01	0.383	0.033	0.373	0.502	0.263	0.184
			3	0.428	0.6	0.229	0	0.264	0.01	0.102	0.413
		CNN-8	2	0.38	0.511	0.275	0.214	0.463	0.636	0.139	0.037
			3	0.474	0.638	0.075	0	0.484	0.659	0.052	0
	Markov Transition Field Images	CNN-4	2	0.501	0.668	0	0	0.388	0.561	0.147	0.161
			3	0.494	0.657	0.02	0.071	0.479	0.651	0	0.018
		CNN-6	2	0.277	0.197	0.385	0	0.239	0	0.38	0.094
			3	0.234	0.065	0.146	0.352	0.29	0.253	0.378	0.036
		CNN-8	2	0.353	0.4	0.375	0.251	0.398	0.54	0.232	0.191
			3	0.443	0.608	0	0.22	0.428	0.553	0.368	0.135
		CNN-4	2	0.501	0.668	0	0	0.504	0.67	0.021	0

	Normal Images		3	0.423	0.57	0	0.313	0.501	0.668	0	0
		CNN-6	2	0.501	0.557	0.018	0.297	0.438	0.582	0.309	0.106
			3	0.375	0.544	0.209	0.152	0.232	0	0.379	0
		CNN-8	2	0.259	0.066	0.351	0.301	0.3	0.445	0.016	0.236
			3	0.486	0.664	0.118	0	0.494	0.664	0.038	0.036

Table 3. Accuracies and F1-scores for different data pre-processing techniques, images, number of strides and architectures using Labeling Strategy 1 and Mid Price

Labeling Strategy 1 – Mid Price											
Type	Image	Architecture	Number of strides	Number of Epochs							
				5				10			
				accuracy	Hold	Sell	Buy	Accuracy	Hold	Sell	buy
TA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.372	0.097	0.171	0.528	0.399	0.251	0.074	0.54
			3	0.364	0.076	0.166	0.523	0.359	0.201	0.197	0.493
		CNN-6	2	0.304	0.455	0	0.08	0.329	0	0.27	0.469
			3	0.349	0	0.409	0.423	0.322	0.455	0.033	0.167
		CNN-8	2	0.389	0.256	0.436	0.444	0.425	0.323	0.438	0.479
			3	0.377	0.214	0.083	0.522	0.354	0.339	0.426	0.23
	Markov Transition Field Images	CNN-4	2	0.332	0.208	0.303	0.425	0.349	0.401	0.245	0.359
			3	0.387	0.201	0.305	0.517	0.327	0.203	0.248	0.44
		CNN-6	2	0.307	0	0.463	0.038	0.389	0	0	0.562
			3	0.319	0.413	0	0.317	0.367	0.244	0.35	0.448
		CNN-8	2	0.324	0.35	0.386	0.173	0.372	0.408	0.099	0.45
			3	0.369	0.363	0.34	0.394	0.357	0.379	0.269	0.396
	Normal Images	CNN-4	2	0.399	0.179	0.078	0.564	0.382	0.015	0.033	0.555
			3	0.342	0.421	0.047	0.365	0.322	0	0.176	0.47
		CNN-6	2	0.317	0.016	0.425	0.275	0.322	0.014	0.468	0.213
			3	0.384	0.095	0.016	0.551	0.299	0.423	0	0.232
		CNN-8	2	0.384	0	0	0.559	0.281	0.014	0.442	0.013
			3	0.362	0.042	0.016	0.529	0.392	0.386	0.265	0.466
PCA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.364	0.182	0.236	0.487	0.349	0.015	0.18	0.5
			3	0.369	0.056	0.06	0.537	0.332	0.102	0.335	0.434
		CNN-6	2	0.334	0	0.441	0.295	0.389	0	0.062	0.556
			3	0.384	0	0.347	0.507	0.364	0	0.293	0.488
		CNN-8	2	0.364	0.07	0.075	0.52	0.319	0.016	0.46	0.156
			3	0.367	0.136	0.335	0.493	0.359	0.015	0.413	0.439
	Markov Transition Field Images	CNN-4	2	0.334	0.19	0.169	0.471	0.367	0.176	0.352	0.478
			3	0.309	0.251	0.176	0.418	0.367	0.03	0.19	0.518
		CNN-6	2	0.296	0.416	0	0.217	0.299	0.439	0	0.16
			3	0.327	0.48	0.163	0	0.327	0.48	0.163	0
		CNN-8	2	0.322	0.14	0.288	0.439	0.389	0.335	0.359	0.455
			3	0.299	0.436	0.171	0.113	0.342	0.252	0.234	0.451
	Normal Images	CNN-4	2	0.389	0	0.17	0.547	0.387	0.015	0.118	0.547
			3	0.344	0.173	0.369	0.406	0.389	0	0.116	0.553
		CNN-6	2	0.364	0.451	0.36	0.222	0.332	0.499	0.142	0
			3	0.281	0.117	0.406	0.176	0.364	0.015	0.193	0.515
		CNN-8	2	0.312	0.4	0.327	0.135	0.309	0.015	0.369	0.406
			3	0.332	0.015	0.433	0.336	0.312	0.276	0.414	0.128
30 Best TA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.394	0.096	0.313	0.536	0.399	0.147	0.231	0.541
			3	0.382	0.015	0	0.554	0.394	0.328	0.284	0.499
		CNN-6	2	0.299	0	0.46	0	0.372	0	0.253	0.509
			3	0.296	0	0.328	0.373	0.337	0	0.456	0.256
		CNN-8	2	0.324	0.439	0.145	0.222	0.384	0.454	0.146	0.452

			3	0.334	0.082	0.42	0.365	0.387	0.374	0.256	0.489
	Markov Transition Field Images	CNN-4	2	0.382	0.068	0.134	0.537	0.392	0.03	0.049	0.56
			3	0.347	0.175	0.156	0.489	0.344	0.289	0.339	0.388
		CNN-6	2	0.299	0	0.46	0	0.392	0	0	0.563
			3	0.407	0.11	0.383	0.525	0.374	0.078	0.09	0.538
		CNN-8	2	0.354	0.32	0.338	0.399	0.377	0.184	0.36	0.478
			3	0.344	0.136	0.115	0.502	0.349	0.114	0.332	0.468
	Normal Images	CNN-4	2	0.382	0	0.255	0.532	0.352	0.054	0.331	0.471
			3	0.334	0.039	0.029	0.506	0.425	0.365	0.269	0.518
		CNN-6	2	0.296	0.395	0.016	0.259	0.387	0.298	0.441	0.397
			3	0.374	0.099	0.19	0.538	0.312	0.476	0	0.013
		CNN-8	2	0.279	0.161	0.327	0.341	0.369	0.358	0	0.476
			3	0.329	0.054	0.363	0.409	0.354	0.107	0.117	0.521

Table 4. Accuracies and F1-scores for different data pre-processing techniques, images, number of strides and architectures using Labeling Strategy 2 and Mid Price

Labeling Strategy 2 – Mid Price											
Type	Image	Architecture	Number of strides	Number of Epochs							
				5				10			
				accuracy	Hold	Sell	Buy	Accuracy	Hold	Sell	buy
TA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.584	0.736	0.022	0.025	0.547	0.703	0	0.145
			3	0.597	0.747	0	0.052	0.597	0.748	0	0
		CNN-6	2	0.499	0.652	0.268	0	0.179	0.032	0	0.293
			3	0.587	0.738	0.061	0	0.456	0.613	0.277	0
		CNN-8	2	0.552	0.712	0.149	0	0.572	0.73	0	0
			3	0.368	0.485	0.292	0.187	0.592	0.745	0.022	0
	Markov Transition Field Images	CNN-4	2	0.599	0.749	0	0.027	0.577	0.732	0.022	0.144
			3	0.531	0.704	0.13	0.075	0.537	0.694	0.121	0.083
		CNN-6	2	0.219	0	0.36	0	0.363	0.507	0.022	0.268
			3	0.592	0.743	0.022	0	0.224	0.04	0.352	0.073
		CNN-8	2	0.441	0.6	0.273	0.118	0.431	0.586	0.139	0.237
			3	0.534	0.688	0.12	0.207	0.537	0.692	0.1	0.146
	Normal Images	CNN-4	2	0.599	0.75	0	0	0.599	0.75	0	0
			3	0.599	0.75	0	0	0.599	0.75	0	0
		CNN-6	2	0.259	0.249	0.187	0.298	0.529	0.694	0.022	0.102
			3	0.338	0.483	0	0.236	0.567	0.725	0	0.132
		CNN-8	2	0.486	0.653	0.173	0.164	0.587	0.733	0.096	0.143
			3	0.579	0.733	0.022	0	0.594	0.747	0	0
PCA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.599	0.75	0	0	0.589	0.744	0	0
			3	0.599	0.75	0	0	0.599	0.75	0	0
		CNN-6	2	0.577	0.732	0.091	0	0.599	0.75	0	0
			3	0.388	0.557	0	0.238	0.186	0	0.215	0.296
		CNN-8	2	0.436	0.598	0.171	0.197	0.554	0.716	0.155	0
			3	0.38	0.508	0.187	0.269	0.491	0.656	0.075	0.233
	Markov Transition Field Images	CNN-4	2	0.589	0.744	0.061	0.051	0.597	0.748	0	0
			3	0.564	0.729	0.06	0	0.554	0.718	0.075	0.
		CNN-6	2	0.209	0.087	0.188	0.294	0.242	0.106	0.355	0.13
			3	0.35	0.496	0.237	0.119	0.599	0.75	0	0
		CNN-8	2	0.554	0.715	0.038	0.089	0.466	0.626	0.196	0.165
			3	0.531	0.703	0.08	0.023	0.521	0.695	0.083	0.077
	Normal Images	CNN-4	2	0.599	0.75	0	0	0.579	0.741	0	0.022
			3	0.572	0.726	0.075	0.051	0.592	0.744	0	0
		CNN-6	2	0.499	0.665	0.162	0.062	0.559	0.721	0.073	0

			3	0.224	0	0.361	0.054	0.549	0.708	0.238	0
		CNN-8	2	0.315	0.399	0.108	0.29	0.222	0.149	0.057	0.31
			3	0.574	0.731	0	0.025	0.569	0.727	0.038	0
30 Best TA Normalized Data	Gramian Angular Field Images	CNN-4	2	0.597	0.747	0	0.053	0.579	0.737	0	0.046
			3	0.577	0.731	0	0.048	0.587	0.743	0	0
		CNN-6	2	0.224	0	0.36	0.161	0.506	0.687	0.063	0.093
			3	0.471	0.662	0	0.144	0.438	0.611	0.09	0.19
		CNN-8	2	0.353	0.499	0.181	0.197	0.222	0.033	0.263	0.322
			3	0.539	0.698	0.132	0.023	0.597	0.747	0	0.053
	Markov Transition Field Images	CNN-4	2	0.599	0.75	0	0	0.463	0.638	0.107	0.194
			3	0.592	0.746	0	0	0.474	0.637	0.11	0.215
		CNN-6	2	0.559	0.718	0.022	0.062	0.393	0.53	0.211	0.189
			3	0.547	0.703	0.103	0.143	0.496	0.656	0.115	0.15
		CNN-8	2	0.448	0.587	0.291	0.175	0.395	0.552	0.178	0.165
			3	0.559	0.727	0.022	0.076	0.451	0.627	0.068	0.164
	Normal Images	CNN-4	2	0.599	0.75	0	0	0.599	0.75	0	0
			3	0.597	0.748	0	0	0.554	0.713	0	0.143
		CNN-6	2	0.463	0.635	0.203	0.168	0.433	0.584	0.251	0.026
			3	0.524	0.674	0.247	0.163	0.584	0.741	0	0.048
		CNN-8	2	0.312	0.443	0.019	0.248	0.547	0.703	0.258	0
			3	0.539	0.705	0.073	0.022	0.295	0.335	0.282	0.25

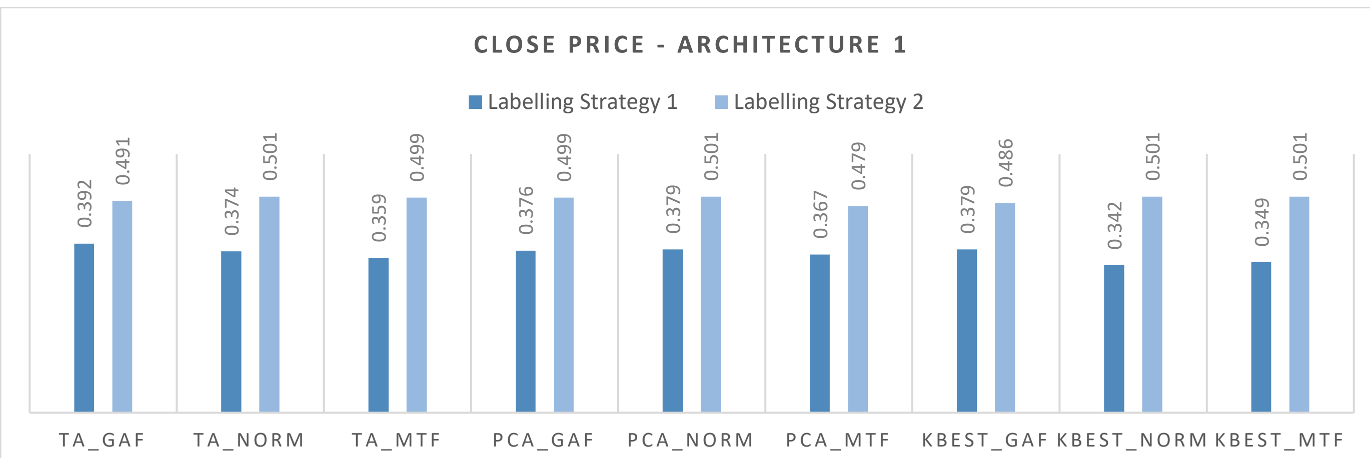

Fig. 12. Comparison among different pre-processing techniques cum images based on labelling strategy 1 and 2 for CNN-4 layered architecture and Close Price

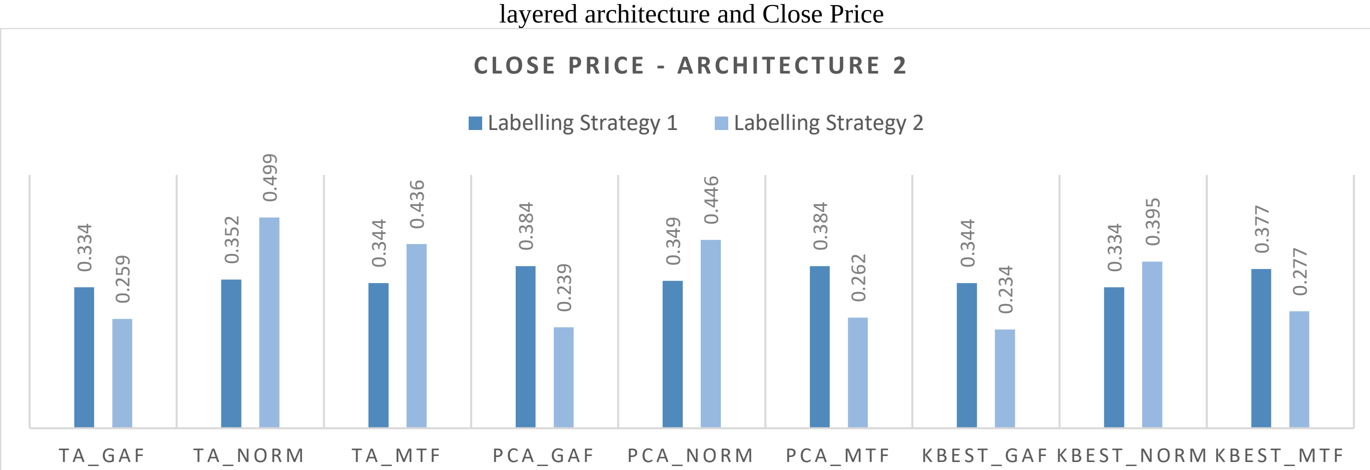

Fig. 13. Comparison among different pre-processing techniques cum images based on labelling strategy 1 and 2 for CNN-6 layered architecture and Close Price

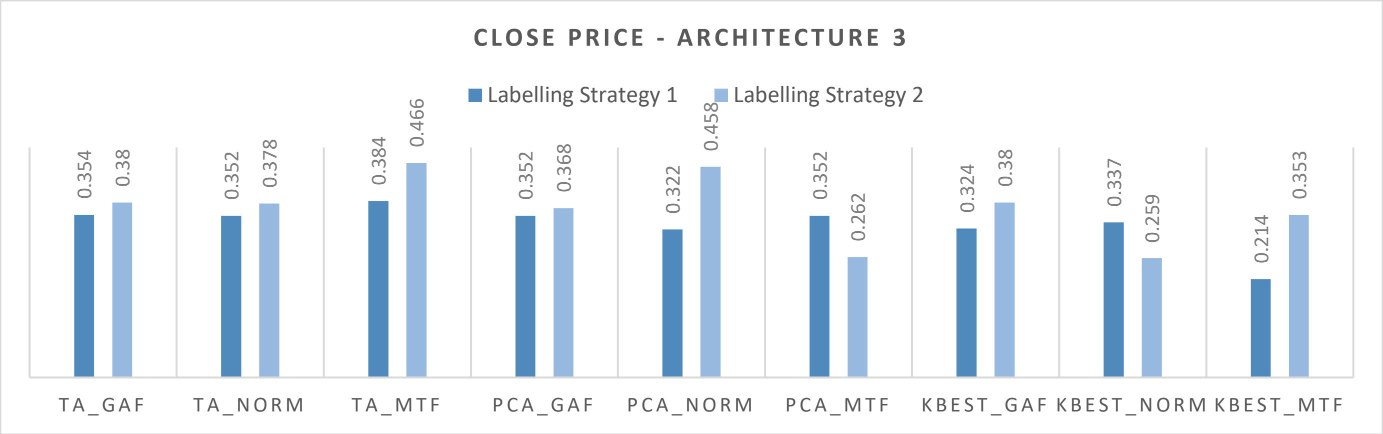

Fig. 14. Comparison among different pre-processing techniques cum images based on labelling strategy 1 and 2 for CNN-8 layered architecture and Close Price

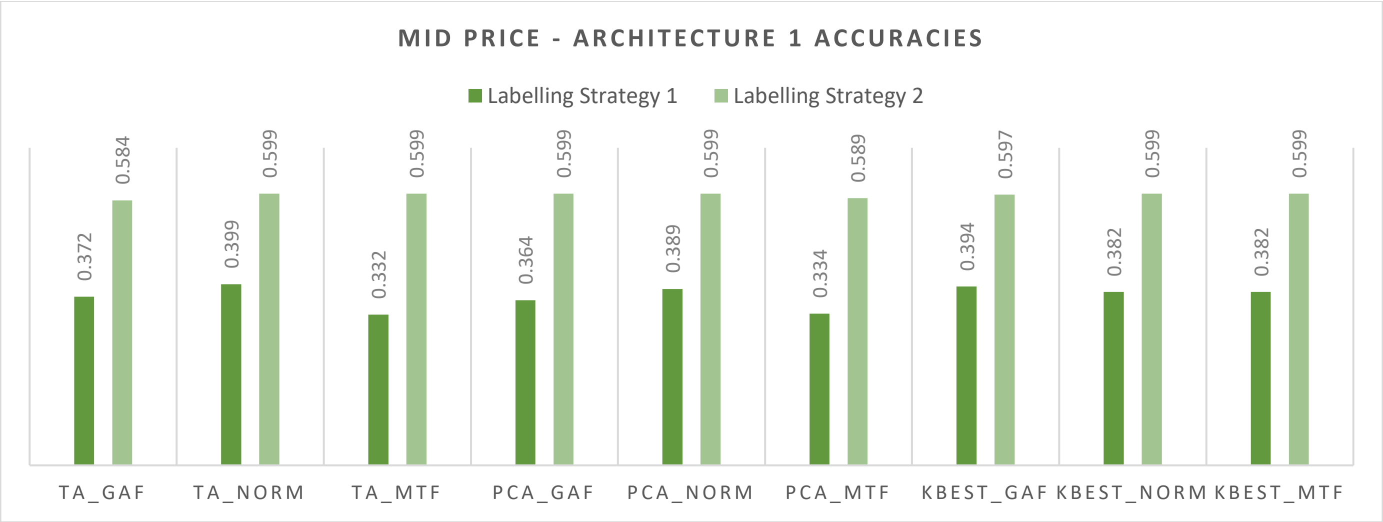

Fig. 15. Comparison among different pre-processing techniques cum images based on labelling strategy 1 and 2 for CNN-4 layered architecture and Mid Price

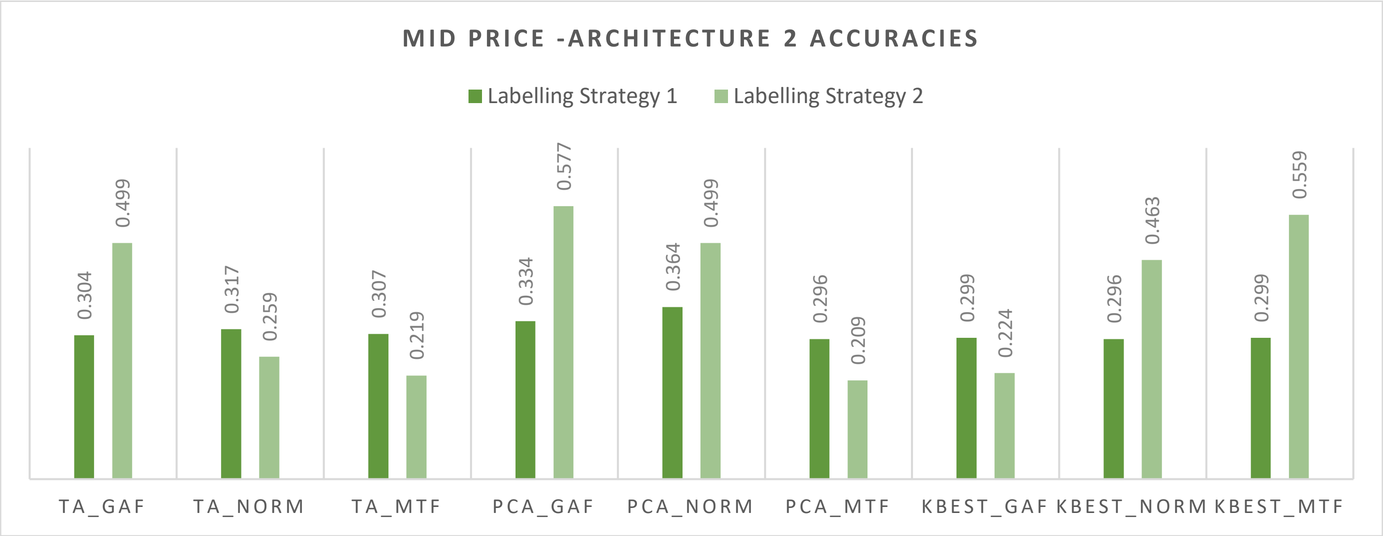

Fig. 16. Comparison among different pre-processing techniques cum images based on labelling strategy 1 and 2 for CNN-6 layered architecture and Mid Price

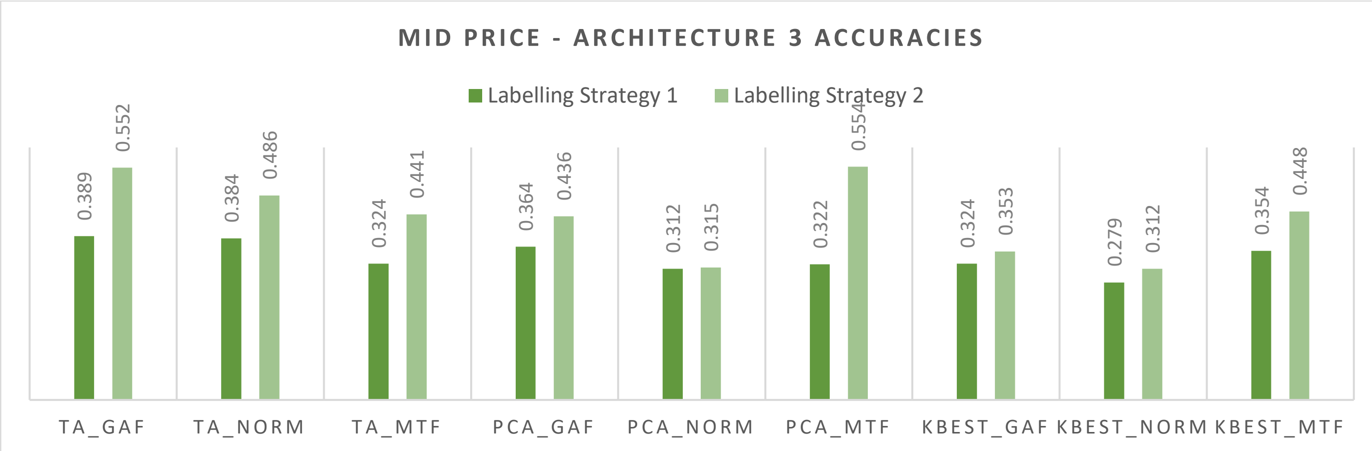

Fig. 17. Comparison among different pre-processing techniques cum images based on labelling strategy 1 and 2 for CNN-8 layered architecture and Mid Price

We have summarized the results from tables and the graphs above in context of each parameter used below:

5.1 In terms of price type predicted

Our initial assumption of mid prices giving much better accuracies compared to close prices in general, stands true from our observations. The accuracies for close prices were mostly in the range of [0.2 to 0.5] while majority of the accuracies for mid prices were in the range of [0.3 to 0.59]

5.2 In terms of Labelling Strategy

We can see that both close and mid prices are giving higher accuracies for labelling strategy 2.

But at the same time, it is to be noted that this labelling strategy suffered from imbalance class classification problem. This led to a much poorer F1 score in case of labelling strategy 2. Labelling strategy 1 on the other hand, had more uniform F1 distributions but accuracy in general was worse than that in case of labelling strategy 2.

5.3 In terms of Architecture

In general, Architecture 2 performed poorest compared to the other architectures. Architecture 1 suffered from the problem of overfitting and gave very high training accuracies, but in many scenarios gave better results in comparison to Architecture 3. Architecture 3 accuracies were in many cases less than that of Architecture 1, but the F1 scores were well defined. Overfitting was minimal in architecture 3 and in general, the results were most reliable.

5.4 In terms of data frames

The results were very ambiguous in case of type of data used. All the three data frames gave very similar results.

5.5 In terms of strides and epochs

Having stride as 2 and epochs as 5 gave optimal results with minimal training time in most cases.

6. Conclusion and Future Work

In this study, we used 2-D CNN on 3 different sets of pre-processed data of Bajaj Finance, encoded them into images and fed them to 4, 6 and 8 layered 2-D CNN models to analyze accuracies and F1-scores of predicted classes. For future work, the accuracies can be improved by applying hybrid models like CNN-LSTM that retain long term changes. Additionally, we could come up with a mechanism for selecting hold window sizes that are specialized for the neighborhood and to remove the non-uniform F1-score distribution which arose due to the imbalance class classification problem, we can give higher weights to ‘sell’ and ‘buy’ classes during training, delete some ‘hold’ values randomly or use techniques like ensemble learning and resampling. Also, there is space for more hyperparameter tuning like changing activation functions and optimizers

7. Acknowledgement

We would like to express our gratitude towards Dr. Vipul Kumar Mishra for encouraging us to do research work which helped us to learn myriad of things and strengthened our analytical skills. A special thanks to Dr. Tanveer Ahmed, Ms. Shambhavi Mishra and Dr. Vimal Baghel who guided us throughout the research work. Their suggestions helped us to gain insights regarding our idea. We are thankful for all the constructive criticism we got throughout this period, this really helped to write and present the paper in a much better way than it could have been. Also, we would like to express our gratitude towards the readers, who took out time to read and understand our work.

8. References

1. Department of Statistics, Stockholm University, A brief history of time series analysis, 2017

2. Dr P. K. Sahoo, Mr Krishna Charlapally, Stock Price Prediction Using Regression Analysis, International Journal of Scientific & Engineering Research, Volume 6, Issue 3, March-2015, ISSN 2229-5518

3. Statistical Learning Theory, Vladimir N. Vapnik(1998, Wiley)

4. R. C. Cavalcante, R. C. Brasileiro, V. L. Souza, J. P. Nobrega, A. L. Oliveira, Computational intelligence and financial markets: A survey and future directions, Expert Systems with Applications 55 (2016) 194–211.

5. Saahil Madge, Predicting Stock Price Direction using Support Vector Machines, Independent Work Report Spring 2015, Princeton

6. J.-Z. Wang, J.-J. Wang, Z.-G. Zhang, S.-P. Guo, Forecasting stock indices with backpropagation neural network, Expert Systems with Applications 38 (11) (2011) 14346–14355.

7. Z. Liao, J. Wang, Forecasting model of the global stock index by stochastic time effective neural network, Expert Systems with Applications 37 (1) (2010) 834–841.

8. A.-S. Chen, M. T. Leung, H. Daouk, Application of neural networks to an emerging financial market: forecasting and trading the Taiwan stock index, Computers & Operations Research 30 (6) (2003) 901– 923.

9. O. B. Sezer, A. M. Ozbayoglu, E. Dogdu, An artificial neural network-based stock trading system using technical analysis and big data framework, in: Proceedings of the SouthEast Conference, ACM, 2017, pp. 223–226

10. S. Dhar, T. Mukherjee, Performance evaluation of Neural Network approach in financial prediction: Evidence from Indian Market, Communication and.

11. B. Vanstone, G. Finnie, T. Hahn, Creating trading systems with fundamental variables and neural networks: The Aby case study, Mathematics and Computers in Simulation 86 (2012) 78–91.

12. Khan, Zabir & Alin, Tasnim & Hussain, Md. Akter. (2011). Price Prediction of Share Market Using Artificial Neural Network 'ANN'. International Journal of Computer Applications. 22. 42–47. 10.5120/2552-3497.

13. R. Aguilar-Rivera, M. Valenzuela-Rend´on, J. Rodr´ıguez-Ortiz, Genetic algorithms and Darwinian approaches in financial applications: A survey, Expert Systems with Applications 42 (21) (2015) 7684– 7697.

14. M. Ozbayoglu, U. Erkut, Stock market technical indicator optimization by genetic algorithms, in Intelligent Engineering Systems through Artificial Neural Networks, Volume 20, ASME Press, 2010.

15. Y.-K. Kwon, B.-R. Moon, A hybrid neurogenetic approach for stock forecasting, IEEE Transactions on Neural Networks 18 (3) (2007) 851–864

16. O. B. Sezer, M. Ozbayoglu, E. Dogdu, A deep neural-network-based stock trading system based on evolutionary optimized technical analysis parameters, Procedia Computer Science 114 (2017) 473–480.

17. S. Mabu, M. Obayashi, T. Kuramoto, Ensemble learning of rule-based evolutionary algorithm using multi-layer perceptron for supporting decisions in stock trading problems, Applied Soft Computing 36 (2015) 357–367.

18. Ding, Y. Zhang, T. Liu, J. Duan, Deep learning for event-driven stock prediction., in Ijcai, 2015, pp. 2327–2333.
19. T. Fischer, C. Krauß, Deep learning with long short-term memory networks for financial market predictions, Tech. rep., FAU Discussion Papers in Economics (2017).

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

21. O. B. Sezer, M. Ozbayoglu, E. Dogdu, A deep neural-network-based stock trading system based on evolutionary optimized technical analysis parameters, Procedia Computer Science 114 (2017) 473–480.

22. O. B. Sezer, A. M. Ozbayoglu, Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach, Applied Soft Computing, 2018

23. Wu, J.MT., Li, Z., Herencsar, N. et al. A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. Multimedia Systems (2021). https://doi.org/10.1007/s00530-021-00758-w

24. Lu, Wenjie & Li, Jiazheng & Li, Yifan & Sun, Aijun & Wang, Jingyang. (2020). A CNN-LSTM-based model to forecast stock prices. Complexity. 2020. 1-10. 10.1155/2020/6622927.

25. Z. Wang and T. Oates, Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks, AAAI Workshop, 2015