Applied Soft Computing 70 (2018) 525–538

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach

Omer Berat Sezer*, Ahmet Murat Ozbayoglu

TOBB University of Economics and Technology, Ankara 06560, Turkey

ARTICLE INFO

Article history:
Received 1 December 2017
Received in revised form 1 March 2018
Accepted 13 April 2018
Available online 27 April 2018

Keywords:
Algorithmic trading
Deep learning
Convolutional neural networks
Financial forecasting
Stock market
Technical analysis

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, 15 different technical indicators each with different parameter selections are utilized. Each indicator instance generates data for a 15 day period. As a result, 15×15 sized 2-D images are constructed. Each image is then labeled 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

Stock market forecasting based on computational intelligence models have been part of stock trading systems for the last few decades. At the same time, more financial instruments, such as ETFs, options, leveraged systems (like forex) have been introduced for individual investors and traders. As a result, trading systems based on autonomous intelligent decision making models are getting more attention in various different financial markets globally [1].

In recent years, deep learning based prediction/classification models started emerging as the best performance achievers in various applications, outperforming classical computational intelligence methods like SVM. However, image processing and vision based problems dominate the type of applications that these deep learning models outperform the other techniques [2].

In literature, deep learning methods have started appearing on financial studies. There are some implementations of deep learning techniques such as recurrent neural network (RNN) [3], convolutional neural network (CNN) [4], and long short term memory (LSTM) [5]. In particular, the application of deep neural networks on financial forecasting models have been very limited.

CNNs have been by far, the most commonly adapted deep learning model [2]. Meanwhile, majority of the CNN implementations in the literature were chosen for addressing computer vision and image analysis challenges. With successful implementations of CNN models, the model error rates keep dropping over years. Despite being one of the early proposed models, AlexNet achieved ~50–55% success rate. More recently, different versions of Inception (v3, v4) and ResNet (v50, v101, v152) algorithms achieved approximately ~75–80% success rate [2]. Nowadays, almost all computer vision researchers, one way or another, implement CNN in image classification problems.

In this study, we propose 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 order to come up with such a representation, 15 different technical indicator instances with various parameter settings each with a 15 day span are adapted to represent the values in each column. Likewise, x axis consists of the time series of 15 days worth of data for each particular technical indicator at each row. Also the rows are ordered in such a way that similar indicators are clustered together to accomplish the locality requirements along the y-axis. As a result, 15×15 pixels sized images are generated and fed into the deep convolutional neural network. To the best of our knowledge, a 2-D representation of financial technical analysis time series data and feeding it as the input for a 2-D image classification based deep CNN, namely CNN-TA, for a financial trading system is novel; since it has not been used

* Corresponding author.
E-mail addresses: oberatsezer@etu.edu.tr (O.B. Sezer), mozbayoglu@etu.edu.tr (A.M. Ozbayoglu).

https://doi.org/10.1016/j.asoc.2018.04.024
1568-4946/

not only for any trading system, but also in any financial prediction model the way we have proposed here. Performance evaluation indicates, such an approach actually performs remarkably well even over long periods. The proposed model outperformed Buy & Hold, common technical indicator based models, most widely used neural network, namely MLP, and the state of the art deep learning time series forecasting model, namely LSTM, on short and long out-of-sample periods. Even though, this is probably one of the first attempts using such an unconventional technique, we believe, the proposed model is promising. Moreover, parameter optimization and model fine tuning might even boost the performance even further.

The rest of the paper is structured as follows: After this brief introduction, the related work is presented in Section 2 followed by the model features described in Section 3. The implementation methodology is given in Section 4 where the data, model and the algorithm details are explained. Financial evaluation of the proposed model is analyzed in Section 5. Finally we conclude in Section 6.

2. Related work

2.1. Time series data analytics

In literature, there are different adapted methodologies for time series data analysis. These can be listed as follows: statistical and mathematical analysis, signal processing, extracting features, pattern recognition, and machine learning. Statistical and mathematical analysis in time series data can be achieved through determining the mathematical parameters such as maximum, minimum, average, moving average, variance, covariance, standard deviation, autocorrelation, crosscorrelation and convolution in the sliding window [6]. Curve fitting, regression analysis, autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), Bayesian analysis, Kalman filter methods are the mathematical methods that are generally used to analyze and forecast time series data in literature [7]. In addition, signal processing methods such as Fourier and wavelet transforms are used to analyze the time series data. Discrete Fourier transform (DFT), discrete wavelet transform (DWT), piecewise aggregate approximation (PAA) are also used to analyze time series data to extract features and find the similarities within the data [8]. Unlike traditional approaches, machine learning models are also used in analyzing time series data and predictions. Machine learning algorithms that are mostly used in time series data analytics are listed as follows: clustering algorithms [9], hidden Markov models [10], support vector machines (SVM) [11–13], artificial neural networks (ANNs) [14–17] self organizing maps (SOM) [18–20].

Time-series forecasting is implemented in various fields such as wind speed forecasting, stock price forecasting, electricity demand forecasting, pollen concentrations forecasting in airborne, human activity recognition forecasting, user behavior forecasting in internet of things applications, and so on. In literature, there are many applications and usage of machine learning on time series data analytics. Arizmendi et al. [14] used ANN in the time series data to predict pollens concentrations in the atmosphere and observed that predictions using ANN performed better than traditional approaches. Srinivasan et al. [15] used a four-layer feedforward ANN to estimate the hourly electrical charge in the power system. Kaastra and Boyd [16] developed an eight-step procedure involving an ANN model in predicting financial and economic time series data. Bezerianos et al. [21] estimated and evaluated the pulse rate change with radial-based function neural network (RBF). Li et al. [22] used a back-propagation artificial neural network (BPANN) and autoregressive (AR) models for estimating the highest values of vibrations in high buildings, which are difficult to measure with instruments. Guan et al. [23] analyzed the sensor data acquired by 40 motion sensors on human legs, by using ANN. With this mechanism, human activities (running, walking, lying, jumping, etc.) are estimated with 97% accuracy. Choi et al. [24] used ANN in the learning part of smart home system. Mohandes et al. [13] applied SVM and MLP on time varying wind speed data and compared the results. In the field of medicine, Morchen [19] used SOM for the extraction of patterns of muscle activities and for the identification of extracted patterns. An et al. [25] proposed a new electricity demand forecasting model called “MFES” that uses feed forward ANN. In their proposal, after application of filtering and seasonal adjustment process, ANN is used to predict future demand. In finance, different machine learning models are also used for forecasting future values. Next subsection covers the financial time series analytics methods in literature.

2.2. Financial time series data analytics

For stock market forecasting, traditional machine learning models have been quite popular. Some researchers directly implemented time-series forecasting based on the financial data, whereas others used technical and/or fundamental analysis data in order to achieve good forecasting performance. ANN, genetic algorithms (GA), fuzzy rule-based systems, as well as hybrid models are among the preferred choices.

Cavalcante et al. [1] surveyed all forecasting model approaches such as ANN, SVM, hybrid mechanisms, optimization and ensemble methods in their survey. Besides, Krollner et al. [26] reviewed machine learning based stock market forecasting papers in different categories such as ANN based models, evolutionary & optimization techniques, and multiple/hybrid methods. Most of the researchers used ANN models to forecast stock market index values [27,28]. Chen et al. [29] proposed a neural network model for forecasting and trading the Taiwan Stock Index. Guresen et al. [30] evaluated the neural network models in particular multi-layer perceptron (MLP) and dynamic ANN to predict NASDAQ stock index. In addition, Sezer et al. [31] proposed an ANN that uses the financial technical analysis indicators (MACD, RSI, Williams%R) to predict Dow30 stock prices turning points. Dhar and Mukherjee [32] used a classical three layer MLP network in their studies to estimate the closing values of the Indian Stock Exchange stocks. Researchers also studied various combinations of network parameters (number of neurons in the input and hidden layers, learning rate) to find the best MLP configuration. Vanstone et al. [33] used MLP to create a system that can give buy/sell points for the Australian market. Fundamental analysis data (price earning ratio, book value, return on equity (ROE) and dividend payout ratio) are used as inputs for MLP.

In addition, genetic and evolutionary approaches are used to predict stock prices and trends [34,35]. Kwon and Moon [36] proposed a RNN with GA optimization to forecast stock values. They tested their proposals with 36 companies in NYSE and NASDAQ from 1992 to 2004. Sezer et al. [37] proposed a deep MLP approach with GA optimization to predict Dow Jones 30 companies’ stock prices. In their studies, RSI parameters (buy value, buy interval, sell value, sell interval) are determined with GA. Best points are used as training data set in deep MLP. Evans et al. [38] used a GA to correct prediction errors and find the best network topology of MLP for forecasting foreign exchange (FOREX) data. Huang [39] used GA to optimize the support vector regression (SVR) parameters and to find which stock should be used as input for method. Pulido et al. [40] used particle swarm optimization (PSO) for the MLP network structure parameters (number of hidden layers and number of neurons in layers and linkage) for time series prediction of the Mexican Stock Exchange.

Also, hybrid machine learning models are proposed to analyze financial time series data. Wang et al. [41] proposed a hybrid SVR model that is combined with principal component analysis (PCA) and brain storm optimization (BSO) to forecast stock prices. In their proposal, 20 different technical indicators are chosen as input to their model. After processing of PCA and BSO, SVR is applied on technical indicators. Mabu et al. [42] proposed the use of an ensemble learning mechanism that combines MLP with a rule-based evolutionary algorithm to be able to determine buy/sell points in stock prices. Ballings et al. [43] compared the performances of ensemble solutions (random forest, adaboost and kernel factory) with classifier models (ANN, logistic regression, SVM and KNN) to estimate movements of stocks in the market.

In financial time series analytics, new approaches started appearing with increasing computational intelligence capacity. In the following subsection, deep learning models used for financial time series analytics in literature will be mentioned.

2.3. Deep learning

A neural network that utilizes deep learning is a specific type of ANN that consists of multiple layers which have different contributions at each layer in such a way that the overall network performs better than its shallow counterparts [44]. There are different types of deep learning models namely convolutional neural network (CNN), recurrent neural network (RNN), deep belief networks (DBN), restricted Boltzmann machines (RBMs), and long short term memory (LSTM) networks. Proposed deep learning models are used for different purposes. In literature, CNNs, DBN, RBM are mostly used for classification and recognition of the images. RNNs and LSTMs are used for analysis of the sequential data, natural language processing, speech recognition and time-series data analytics. In addition, CNNs are mostly used in image/video processing, classification and recognition processes [45–48], but also used in natural language processing and sentence classification [49,50].

Even though deep learning models, in particular deep CNNs have been among the most popular choices in recent years, there are only a limited number of implementations of deep neural networks for financial problems. Ding et al. [51] proposed deep learning method for event driven stock market prediction to extract news texts, information from internet and newspapers. They also used a deep CNN and neural tensor network to model short-term and long-term impacts of circumstances on stock price changes. They used S&P 500 stock historical data to test their models. Längkvist et al. [52] surveyed the methods for analyzing the time-series data in terms of RBM, autoencoder, RNN, deep learning, convolution and pooling, and hidden markov model (HMM). In addition, they mentioned and surveyed the deep learning methods to evaluate stock market indexes and prices in their review. Fischer et al. [5] used LSTM to forecast the direction of trend for stocks of the S&P 500 between 1992 and 2015. Krauss et al. [3] compared deep neural nets, gradient-boosted-trees, random forests in their research. They used deep neural network model to forecast stock prices in S&P 500 between 1992 and 2015.

In addition, Yoshihara et al. [53] proposed an approach that uses RBM and DBN to predict the trend of stock prices on the Nikkei Stock Exchange using news events. The proposed solution was tested with ten Nikkei stocks and the results were compared with SVM. Shen et al. [54] proposed a new model to forecast the exchange rates by combining an advanced DBN and the conjugate gradient method. The proposed technique was compared with feed forward neural networks. They concluded that the proposed deep learning approach was better than the traditional methods. Tino et al. [55] compared the Markov model (MM) and RNN methods with different parameters to estimate the movements of the German DAX and British FTSE indexes (volatiliy, etc.). According to the results obtained, RNN could not give better results than MM. Deng et al. [56] proposed a method that uses Deep direct reinforcement (DDR) method and fuzzy deep direct reinforcement (FDDDR) method together with Recurrent DNN (RDNN). The proposed method was applied in the Chinese futures market and commodity futures market (silver and sugar prices).

In their previous implementation [37], the authors used evolutionary algorithms to optimize the technical analysis parameters of commonly used indicators and developed a deep feedforward neural network using these optimized parameters as inputs. The results indicate deep learning can achieve good learning and generalization of buy–sell points for individual stocks over long out-of-sample test periods.

In literature, CNN is mostly used for image classification / analysis problems, it is generally not preferred for time series data analytics directly. Meanwhile, their success in computer vision over traditional models is quite remarkable. For financial time series forecasting, deep learning algorithms, most commonly RNN and LSTM networks were the preferred choices in recent years. At the same time, algorithmic trading systems mostly depend on technical analysis indicators along with some other inputs. However, models that integrate technical analysis data with deep neural networks is not very common in literature. Moreover, using CNN with 2-D matrix representation of the technical analysis data for algorithmic trading is novel. With the proposed study, technical analysis data and deep CNN are combined. The difference between the proposed model and the other methods is that the technical analysis data is applied on the prices to create feature vectors and matrices (two-dimensional images); hence, the financial time series forecasting problem is implicitly converted into an image classification problem. In this study, we aimed to develop algorithmic trading models that can make financial forecasts in the medium and short term, with stable decisions that can provide maximum profit and less risk (variance).

3. Model features and convolutional neural network (CNN)

For analyzing and developing inference models from financial data, there are two widely-adopted approaches: technical analysis and fundamental analysis [1]. Fundamental analysis can be implemented through examining company specific financial data such as balance sheet, cash flow, return on assets. Meanwhile, technical analysis can be implemented through analyzing past financial time series data using mathematical and/or rule-based modeling. There are many technical indicators that are used for predicting future directions of financial assets. In our study, we used 15 separate technical indicators with different time intervals. Selected technical analysis indicators and their corresponding formulas are summarized in Appendix A.

CNN is a feedforward ANN that takes its inputs as 2-D matrices. Unlike a fully connected neural network like MLP, the locality of data within the input vector (or matrix) is important. Hence, the neighboring data points within the matrix should be carefully chosen. This is not an issue for image classification problems, since the aforementioned requirement is satisfied directly because the neighboring pixels are related to each other in both directions.

CNN generally consists of two types of layers: the convolutional layer and the subsampling layer. It structurally has successive convolutional and sampling layers. In convolution layer, convolution operation is applied, results are passed to the next layer. In subsampling layer, number of parameters and the spatial size of the representation are reduced. In the last subsampling layer, the data becomes a one-dimensional vector. Finally, the last layer is connected to a fully connected MLP. Through that, the high-level decision making is performed just as in the case of a traditional

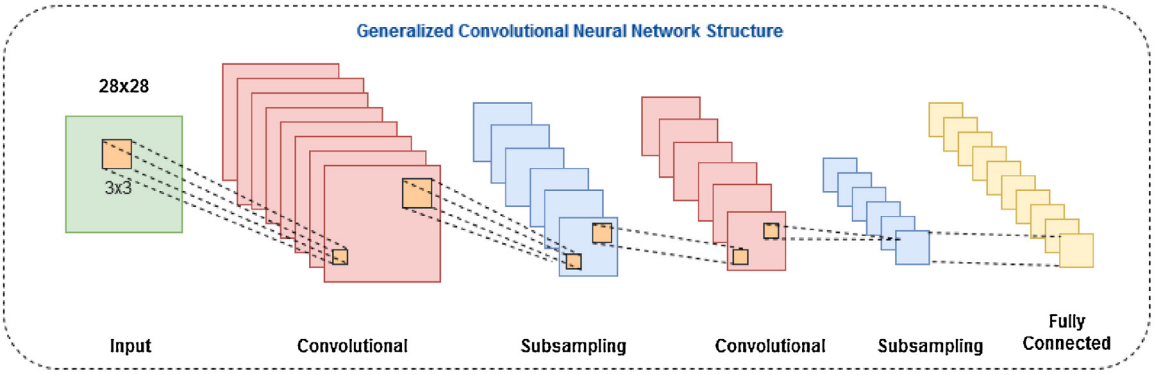

Fig. 1. Generalized convolutional neural network.

classifier. As a result, the previous layers of CNN actually performs an implicit feature extraction. CNNs have a wide range of applications in image and video recognition, natural language processing and expert systems. The general CNN structure is shown in Fig. 1 [57,47].

4. Method

For our algorithmic trading model, we propose a novel method that uses CNN to determine the "Buy" and "Sell" points in stock prices using 15 different technical indicators with different time intervals and parameter selections for each daily stock price time series to create images. We also use Apache Spark, Keras and Tensorflow to create and analyze the images and perform big data analytics. As can be seen in Fig. 2, our proposed method is divided into five main steps: dataset extract/transform, labeling data, image creation, CNN analysis and financial evaluation phases. Our objective is to determine the best fit for the buy, sell, and hold points in the time series of the associated stock prices.

4.1. Preprocessing (DataSet extract/transform)

In our study, the daily stock prices of Dow 30 stocks and daily Exchange-Traded Fund (ETFs) prices are obtained from finance.yahoo.com for training and testing purposes. Stock/ETF prices between 1/1/2002 to 1/1/2017 are used for training and testing purposes. We adapted a sliding window with retraining approach where we chose a 5 year period for training and the following 1 year for testing, i.e. training period: 2002–2006, testing period: 2007. Then we moved both training and testing periods one year ahead, retrained the model and tested with the following year, i.e. training period: 2003–2007, testing period: 2008. As a result, each year between 2007 and 2016 is tested using repeated retraining. Fig. 3 illustrates this sliding training and testing approach. In the first step, dataset extract/transform phase, the downloaded prices are normalized according to the adjusted close prices.

4.2. Labeling

After extracting the data for the intended period in labeling phase, all daily close prices are manually marked as "Hold", "Buy", or "Sell" by determining the top and bottom points in a sliding window. Bottom points are labeled as "Buy", top points are labeled as "Sell", and the remaining points are labeled as "Hold". The structure of the labeling process is given in Algorithm 1.

Algorithm 1. Labeling Method

```
procedure LABELING()
windowSize = 11 days
while(counterRow < numberOfDaysInFile)
    counterRow ++
    If (counterRow > windowSize)
      windowBeginIndex = counterRow − windowSize
      windowEndIndex = windowBeginIndex + windowSize − 1
      windowMiddleIndex = (windowBeginIndex + windowEndIndex)/2
      for (i = windowBeginIndex;i <= windowEndIndex;i ++)
        number = closePriceList. get(i)
        if(number < min)
          min = number
          minIndex = closePriceList. indexOf(min)
        if(number > max)
          max = number
          maxIndex = closePriceList. indexOf(max)
      if(maxIndex == windowMiddleIndex)
        result="SELL"
      elif(minIndex == windowMiddleIndex)
        result="BUY"
      else
        result="HOLD"
```

4.3. Image creation

In image creation phase, for each day, RSI, Williams %R, WMA, EMA, SMA, HMA, Triple EMA, CCI, CMO, MACD, PPO, ROC, CMFI, DMI, and PSI values for different intervals (6–20 days) are calculated using TA4J (Technical Analysis For Java)[1] library. These particular indicators are mostly oscillator and trend based financial time series filters that are commonly preferred by short to medium term traders. Since 6–20 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.

For each day a 15×15 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. Fig. 4 illustrates sample 15x15 Pixel images that are created during the image creation phase.

In our study, there are approximately 1250 images for each stock price training data in a 5 year period. Even though each year from 2007 to 2016 is tested separately, their results are combined and corresponding annualized metrics are calculated to represent

[1] https://github.com/mdeverdelhan/ta4j.

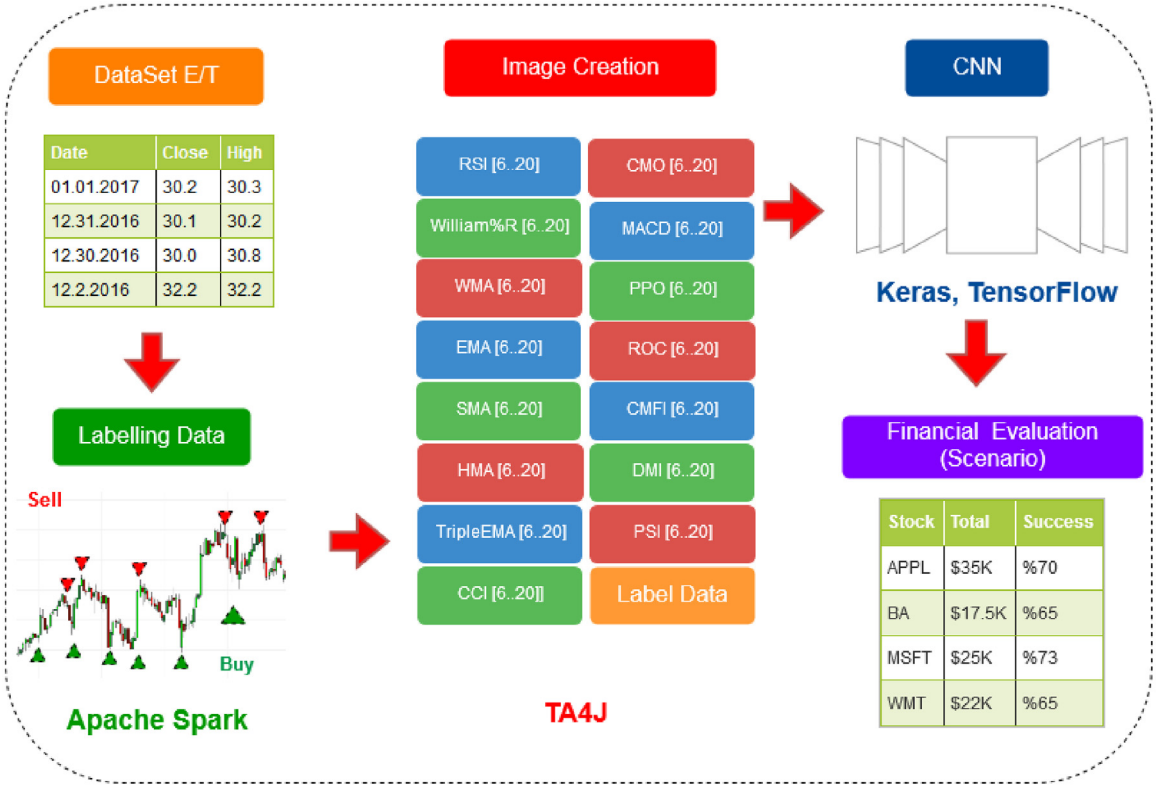

Fig. 2. Proposed method for CNN-TA.

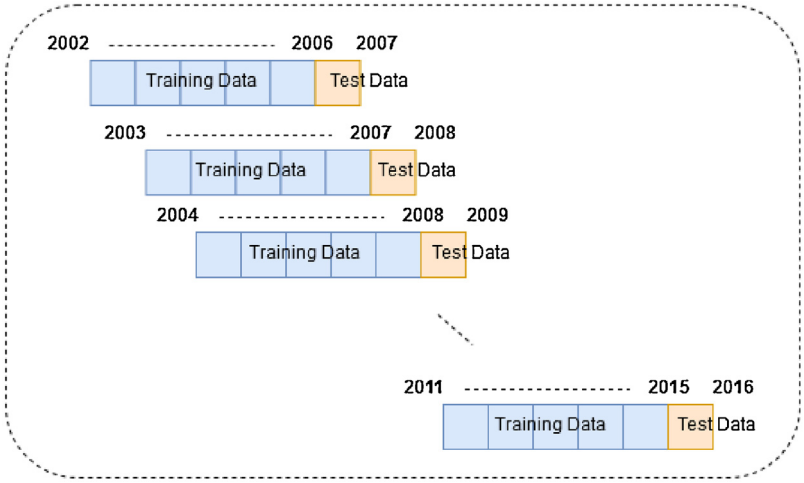

Fig. 3. Training and testing approach.

longer test periods. Two sets of long term test periods are chosen to be able to verify the performance of the proposed model in different market conditions. Approximately 2500 images for each stock price are generated for the first test case (1/1/2007 to 1/1/2017) which is aimed to verify the sustainability of the model performance in a 10 year span. In addition, 1250 test images are used for the second test case (between 1/1/2007 to 1/1/2012) which covers the 2008–2009 financial crisis period. For each stock and ETF, different training and test data image files are prepared and they are evaluated separately for their different characteristic features.

4.4. CNN

In our proposed CNN analysis phase, as can be seen in Fig. 5, nine layers are used. These are listed as follows: input layer (15×15), two convolutional layers ($15 \times 15 \times 32$, $15 \times 15 \times 64$), a max pooling ($7 \times 7 \times 64$), two dropout (0.25, 0.50), fully connected layers (128), and an output layer (3). Dropout layers are added to prevent overfitting. In our proposed model CNN-TA, 3×3 filter size is used for CNN filter. In literature, different size of CNN filters are adapted: 3×3, 5×5 and 7×7. Decreasing filter size generally results in catching more details of the images. 3×3 is the smallest and most commonly used kernel size in the image processing application

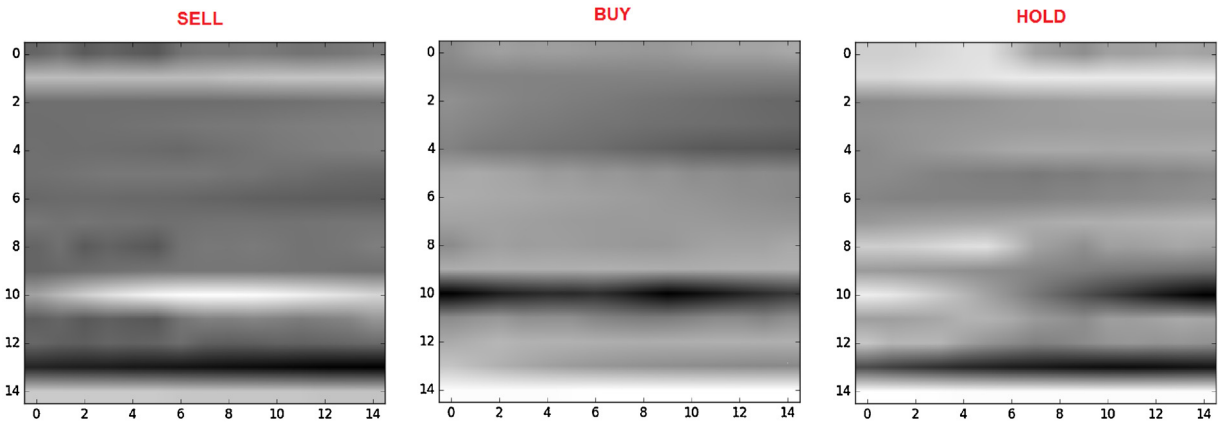

Fig. 4. 15 × 15 pixel labeled sample images after image creation phase.

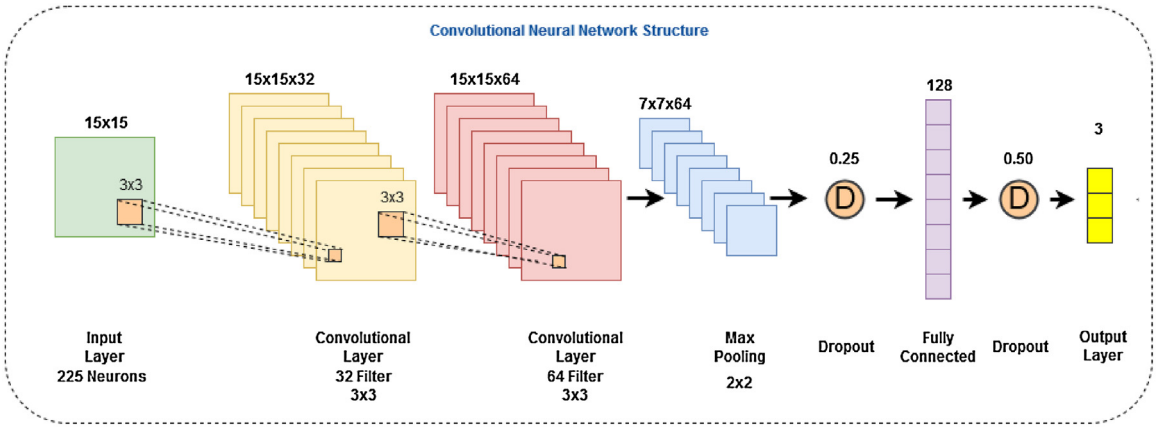

Fig. 5. CNN process.

(AlexNet [45]).Using a filter size of 3 × 3 provides the convolution capability with closest neighbors' (upper, lower, right, left, upper left, upper right, lower left, lower right) information while processing the current layer; hence sharp variations within the image can be captured. In our particular study, we also preferred 3 × 3 filter size, since we have relatively small images (15 × 15) and there can be significant intensity variations within the images (see Fig. 4).

In the proposed model CNN-TA, the adapted CNN structure is similar to the deep CNN used in the MNIST algorithm except 28 × 28 images were used as inputs in that particular implementation. LeNet CNN structures, [58] the deep CNN model with first successful results, consist of six layers. In addition, adding more layers increases the complexity of the algorithm. Without a large training set, an increasingly complex network is likely to overfit and reduce the accuracy on the test data. For future work, deeper models with more processing layers can be configured when more training data is available.

In our proposed CNN structure, there are different layers: convolutional, maxpooling, dropout and full connected MLP layer. Convolutional layer consists of convolution operation. Basic convolution operation is shown in Eq. (1) (t denotes time). In addition, convolution operation is implemented on two dimensional images. Eq. (2) illustrates the convolution operation of two dimensional image (I denotes input image, K denotes the kernel). Besides, consecutive convolutional and maxpooling layers build the deep neural network structure. Eq. (3) provides the details about the neural network architecture (W denotes weights, x denotes input and b denotes bias). At the end of the network, softmax function is used to get the output. Eq. (4) shows the softmax function (y denotes output) [59].

In this study, we implemented the CNN structure using Keras[2], Tensorflow[3] infrastructure and each test run lasts for 200 epochs. Number of epochs is fine-tuned with choosing different number of epochs in different tests.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{1}$$

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n). \tag{2}$$

$$e_i = \sum_{j} W_{i,j}x_j + b_i. \tag{3}$$

$$y = softmax(e) \tag{4}$$

4.5. Financial evaluation

In the last step, buy–sell decisions are made according to the predicted Buy, Sell, Hold labels. With associated consecutive Buy–Sell pairs, financial trades are implemented. These trades are stored in a transaction table and each transaction is evaluated through a financial evaluation model. The results of the financial evaluation will be presented in the next section for the selected test periods.

[2] https://keras.io/.
[3] https://github.com/tensorflow.

Table 1
Selected ETFs and their descriptions.

Name	Description	Inception date	Volume
XLF	Financial Select Sector SPDR ETF	12/16/1998	71,886,065
XLU	Utilities Select Sector SPDR ETF	12/16/1998	11,342,530
QQQ	PowerShares QQQ ETF	10/03/1999	33,918,165
SPY	SPDR S&P 500 ETF	1/22/1993	68,675,793
XLP	Consumer Staples Select Sector SPDR ETF	12/16/1998	9,721,714
EWZ	iShares MSCI Brazil Capped ETF	7/10/2000	19,613,073
EWH	iShares MSCI Hong Kong ETF	3/12/1996	2,586,985
XLY	Consumer Discret Sel Sect SPDR ETF	12/16/1998	4,257,841
XLE	Energy Select Sector SPDR ETF	12/16/1998	16,494,257

Algorithm 2 summarizes the overview of the proposed algorithmic trading model.

Algorithm 2. Generalized Proposed Algorithm

procedure AllPhases()
Phase DataSet E/T:
dataset = *read*(*open*, *close*, *high*, *low*, *adjustedClose*, *volume*)
dataset. adjustRatio = *dataset. close*/*dataset. adjustedClose*
adjust(*dataset. open*, *dataset. close*, *dataset. high*, *dataset. low*) *with adjustRatio*
Phase Data Labeling:
calculate Label (*Buy*/*Sell*/*Hold*) *using sliding window*
Phase Image Creation:
calculate technical analysis values (*RSI*, *EMA*, *MACD. .*) *for each line in dataset*
create 15*x*15 *images*
merge label and technical analysis values
normalize technical analysis values between [1, −1]
for(*i* = 0;*i* < 15;*i* ++)
    trainingDataset[*i*] = *dataset. split*(*dates* = (1997 + *i*) *to* (2002 + *i*))
    testDataset[*i*] = *dataset. split*(*dates* = (2003 + *i*))
Phase CNN:
foreach(*trainingDataset*[*i*] *and testDataset*[*i*])
    trainingDataset[*i*] = *resample*(*trainingDataset*) *to solve data imbalance problem*
    model = *CNN*(*epochs* = 200, *blocksize* = 1028)
    model. train(*trainingDataset*[*i*])
    model. test(*testDataset*[*i*])
Phase Financial Evaluation:
foreach(*trainingDataset*[*i*] *and testDataset*[*i*])
    evaluateResults()

5. Performance evaluation

The overall performance of our proposed model CNN-TA is evaluated using two different evaluation criteria: 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.

5.1. Test data

In the financial evaluation phase, our proposed method (CNN-TA) is evaluated with ETFs and Dow Jones 30 Stocks with different time periods (2007–2012 for analyzing the effects of 2008 financial crisis, 2007–2017 for evaluating the performance of the last 10 years). Our proposed model is trained with five-years training data and tested with one-year out-of-sample data. Then, the network is retrained with the next five-years training data (with one year move ahead as explained in the previous section) and tested with next one-year out-of-sample data. Selected ETFs and their descriptions are illustrated in Table 1. The chosen ETFs have the highest trading volumes with enough training data.

Table 2
Confusion matrix of test data (Dow-30).

	Predicted			
		Hold	Buy	Sell
Actual	Hold	52,364	18,684	23,592
	Buy	1268	5175	3
	Sell	1217	8	5059

Table 3
Evaluation of test data (Dow-30).

Total accuracy: 0.58			
	Hold	Buy	Sell
Recall	0.55	0.80	0.81
Precision	0.95	0.22	0.18
F1 Score	0.70	0.34	0.29

Table 4
Confusion matrix of test data (ETFs).

	Predicted			
		Hold	Buy	Sell
Actual	Hold	18,629	5180	6498
	Buy	478	1215	0
	Sell	587	0	1127

Table 5
Evaluation of test data (ETFs).

Total accuracy: 0.62			
	Hold	Buy	Sell
Recall	0.61	0.72	0.66
Precision	0.95	0.19	0.15
F1 Score	0.75	0.30	0.24

5.2. Computational model performance

The prediction performance of the proposed model is analyzed. Even though, each stock/ETF performance is considered separately, due to space constraints, only summary results will be presented. Table 2 tabulates the confusion matrix for Dow-30 test data. Table 3 illustrates the performance evaluation of the results obtained through the confusion matrix for Dow-30 data. Recall values of class "Buy" and class "Sell" are better when compared with class "Hold". However, classes "Buy" and "Sell" have worse precision values compared to class "Hold". For stock trading systems, accurate entry and exit points (classes "Buy" and "Sell") are important for the overall success of the trading algorithm. In our case, most of the "Buy and Sell" points are captured correctly by the proposed model. However, a lot of false entry and exit points are also generated. This is mainly due to the fact that "Buy" and "Sell" points appear much less frequent than "Hold" points, it is not easy for the neural network to catch the "seldom" entry and exit points without jeopardizing the general distribution of the dominant "Hold" values. In other words, in order to be able to catch most of the "Buy" and "Sell" points (recall), the model has a trade-off by generating false alarms for non-existent entry and exit points (precision). Besides, Hold points are not as clear as "Buy" and "Sell" (hills and valleys). It is quite possible for the neural network to confuse some of the "Hold" points with "Buy" and "Sell" points, especially if they are close to the top of the hill or bottom of the valley on sliding windows. Table 2 provides the confusion matrix of Dow30 test data and Table 3 illustrates the evaluation of the confusion matrix of Dow30 test data. Table 4 provides the confusion matrix of ETFs test data and Table 5 illustrates the evaluation of the confusion matrix of ETFs

test data. Performance Evaluation results indicate that ETFs' results (overall accuracy) are better than Dow30 results. This can be as a result of ETFs being more stable and less sensitive to events, economic crises, political decisions compared to stocks making them less volatile. This lack of volatility results in a more stable environment for algorithmic trading models to learn the trading model easier.

5.3. Financial evaluation

In the last step of our algorithmic trading model, the generated transactions are analyzed using the financial evaluation method. In our model, each stock is bought, sold or held according to the predicted label. If the predicted label is "Buy", the stock is bought at that point with all of the current available capital (if not bought before already). If the predicted label is "Sell", the stock is sold at that price (if it has been bought). If the predicted label is "Hold", no action is taken at that point. During a financial trade, if the same label comes consecutively, only the first label is activated and the corresponding transaction is performed. Repeating labels are ignored until the label changes. Starting capital for financial evaluation is $10,000, trading commission is $1 per transaction. Financial evaluation scenario is illustrated in Eq. (5) ("S" denotes financial evaluation scenario, "tMoney" denotes totalMoney, "#OfStocks" denotes numberOfStocks). The corresponding formulas for the evaluation metrics (presented in Tables 10 and 11) are listed in Eqs. (6)–(11):

$$S = \begin{cases} \#OfStocks = \dfrac{tMoney}{price}, & \text{if label = 'Buy'} \\ no \quad action, & \text{if label = 'Hold'} \\ tMoney = price * \#OfStocks & \text{if label = 'Sell'} \end{cases} \tag{5}$$

$$AR = \left(\left(\frac{totalMoney}{startMoney} \right)^{1/number \ \ of \ \ years} - 1 \right) * 100 \tag{6}$$

$$AnT = \frac{transactionCount}{numberOfYears} \tag{7}$$

$$PoS = \frac{successTransactionCount}{transactionCount} * 100 \tag{8}$$

$$ApT = \frac{totalPercentProfit}{transactionCount} * 100 \tag{9}$$

$$L = \frac{totalTransactionLength}{transactionCount} * 100 \tag{10}$$

$$IdleR = \frac{data.length - totalTransLength}{data.length} * 100 \tag{11}$$

5.4. Compared models

Our proposed model is also compared with "Buy&Hold" Strategy (BaH), RSI (14 days, 70–30), SMA (50 days), LSTM and MLP regression methods. Each method, model and strategy has been implemented and relevant financial calculation scenarios have been analyzed. In the "Buy&Hold" strategy, the stock is bought at the beginning of the test data, sold at the end of the test data. In the RSI model, the RSI value is calculated for each day in the test data. If the RSI value of the corresponding test data is less than 30, buy signal is generated. If the RSI value of the corresponding test data is more than 70, sell signal is generated. In the SMA model, a 50-day SMA value is calculated for each day in the test data. Buy signal is generated if the corresponding test data is more than 50 days-SMA value, whereas if it is less than 50 days-SMA value, sell signal is generated. LSTM [5] and MLP regression [30,60] models are also used in the analysis of financial time series data in the literature. The implemented LSTM model is composed of 25 neurons (input layer: 1 neuron, hidden layer: 25 neurons, output layer: 1 neuron, dropout: 0.5, epoch: 1000, time steps: 240) [5]. The implemented MLP is a model consisting of 4 layers (layers: 1, 10, 5, 1, dropout: 0.5, epoch: 200, time steps: 100) [5].

Table 6
Comparison of annualized returns of the proposed system (CNN-TA) with BaH, RSI, SMA, LSTM, MLP Reg. models (ETFs – test period: 2007–2017).

ETFs	CNN-TAr	BaHr	RSIr	SMAr	LSTMr [5]	MLPr [5]
SPY	**10.77%**	4.63%	6.14%	0.54%	3.35%	7.27%
QQQ	**11.57%**	10.52%	6.46%	5.37%	−1.33%	3.92%
XLU	**10.13%**	2.97%	4.91%	0.90%	1.29%	1.12%
XLE	**15.80%**	2.85%	3.64%	5.88%	4.01%	5.41%
XLP	**11.10%**	6.92%	5.29%	5.17%	5.97%	0.84%
XLY	**9.55%**	7.68%	5.74%	1.80%	2.13%	1.31%
EWZ	**20.40%**	−3.38%	−3.25%	4.15%	−1.30%	5.42%
EWH	11.69%	1.73%	2.32%	3.08%	**12.30%**	6.11%
XLF	16.05%	2.31%	1.22%	−5.60%	**16.18%**	2.68%
Average	**13.01%**	4.63%	3.95%	2.81%	6.22%	4.01%
St.Dev.	3.61%	4.03%	3.12%	3.57%	5.96%	**2.40%**

5.5. ETF analysis

The average annualized return for our proposed method (ETFs-2007-2017) is 13.01% and percent of successful transactions is 71.51% (Table 11), whereas BaH average annualized return is 4.63%, RSI model average annualized return is 3.95%, SMA model average annualized return is 2.81%, LSTM model average annualized return is 6.22%, and MLP regression average annualized return is 4.01% (Table 6). Proposed method's average annualized return is almost three times better than BaH, RSI, SMA, MLP regression average annualized returns. At the same time, our proposed model and MLP regression are the only models with positive annualized returns for all ETFs during the 10 year test period. Meanwhile, the standard deviation of annualized returns of our model is also low indicating stable and consistent returns. (Table 6). Fig. 6 shows the proposed model's accumulation of the capital for selected ETFs. In each case, the model performance is compared against Buy&Hold during the corresponding period. The performance results for other ETFs also show similar characteristics. Best annualized return values in Table 6 are shown in bold.

The method is also tested and compared against other aforementioned models for the test period between 2007 and 2012 which coincides with the 2008 and 2009 financial crisis. The stock market was hit very hard during that period and a lot of stocks/funds/ETFs had negative returns. During that period, the average annualized return for our proposed method (ETFs-2007-2012) is 13.17% and percent of successful transactions is 71.44% (Table 11), whereas BaH average annualized return is 2.60%, RSI model average annualized return is −0.01%, and SMA model average annualized return is 1.30%, LSTM model average annualized return is 8.44%, and MLP regression average annualized return is 8.23%, (Table 7). Proposed method's average annualized return is almost five times better than BaH average annualized return in that particular period. In addition, our proposed solution's average annualized return is almost 1.5 times better than LSTM and MLP regression average annualized returns during the financial crisis period. The standard deviation for the annualized returns was higher for this period when compared with 2007–2017 case. However, when compared with the other models, it was still better than the majority. Best annualized return values in Table 7 are presented in bold.

5.6. Dow30 analysis

As mentioned earlier, our proposed method was also evaluated with Dow Jones 30 Stocks in different time periods (2007–2012 and

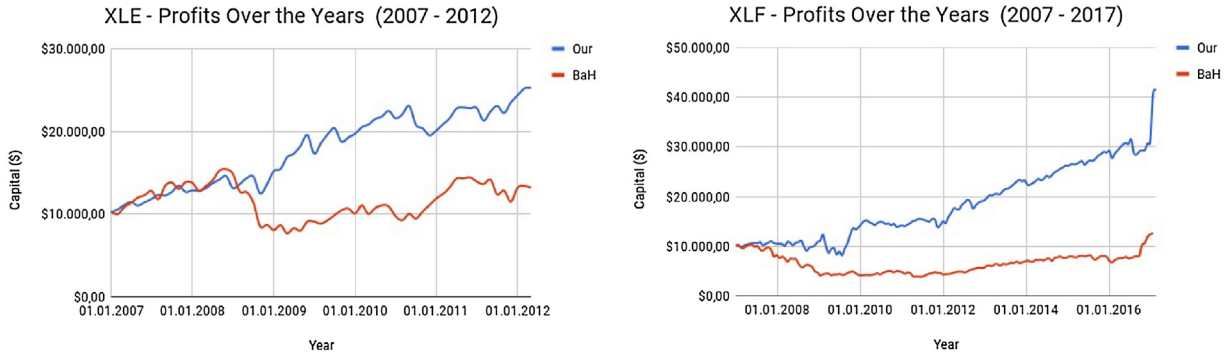

Fig. 6. Comparison of the proposed algorithm and BaH method results on XLE and XLF ETFs.

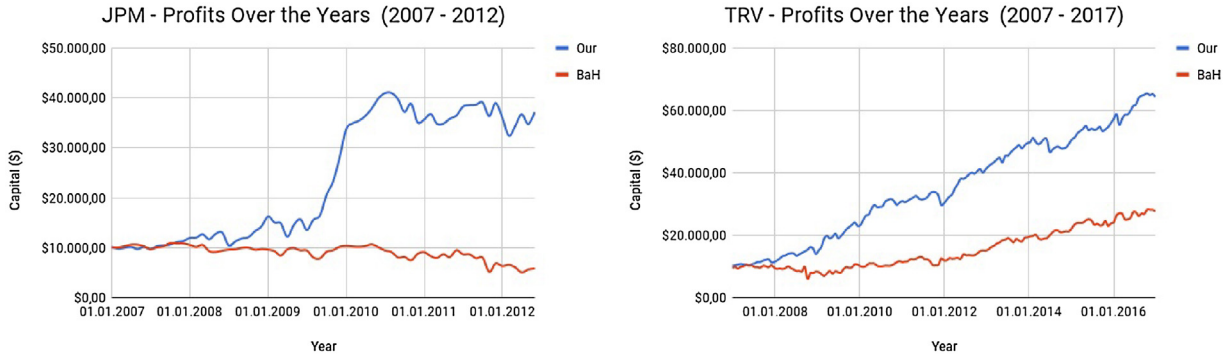

Fig. 7. Comparison of the proposed algorithm and BaH method results on JPM and TRV stocks.

Table 7
Comparison of annualized returns of the proposed system (CNN-TA) with BaH, RSI, SMA, LSTM, MLP Reg. models (ETFs test period: 2007–2012).

ETFs	CNN-TAr	BaHr	RSIr	SMAr	LSTMr [5]	MLPr [5]
SPY	9.25%	−0.39%	−1.64%	−1.56%	**11.05%**	10.62%
QQQ	**8.75%**	5.73%	2.60%	6.35%	5.38%	2.85%
XLU	**12.43%**	3.72%	0.84%	0.13%	6.89%	3.78%
XLE	**23.41%**	5.61%	−0.32%	7.45%	3.76%	12.93%
XLP	**12.59%**	6.97%	3.88%	4.07%	1.20%	4.37%
XLY	5.43%	1.65%	2.00%	−3.68%	3.88%	**7.41%**
EWZ	**25.07%**	7.64%	−1.54%	7.55%	1.69%	3.54%
EWH	12.32%	1.75%	−0.01%	−0.63%	**26.07%**	17.64%
XLF	**9.34%**	−16.91%	−7.68%	−16.06%	6.39%	7.02%
Average	**13.17%**	2.60%	−0.01%	1.30%	8.44%	8.23%
St.Dev.	6.69%	7.49%	**3.36%**	7.44%	7.61%	5.03%

Table 8
Comparison of annualized returns of the proposed system (CNN-TA) with BaH, RSI, SMA, LSTM, MLP Reg. models (DOW30 – test period: 2007–2017).

Stock	CNN-TAr	BaHr	RSIr	SMAr	LSTMr [5]	MLPr [5]
MMM	10.88%	**11.36%**	4.64%	2.25%	6.43%	2.28%
AXP	**25.05%**	4.25%	−0.85%	−0.96%	7.23%	9.56%
APPL	11.37%	**26.42%**	10.11%	19.55%	5.03%	2.57%
BA	7.03%	**8.60%**	2.30%	2.07%	4.01%	2.59%
CAT	4.33%	7.19%	−3.02%	**10.72%**	−1.53%	0.66%
CVX	**14.91%**	8.67%	4.60%	1.07%	7.63%	5.76%
CSCO	**10.02%**	2.76%	5.57%	−5.28%	9.05%	9.15%
KO	**11.13%**	8.85%	7.78%	3.09%	3.70%	2.71%
DIS	**13.97%**	13.14%	0.96%	6.36%	4.67%	5.36%
XOM	**14.51%**	4.78%	4.81%	−2.34%	6.67%	3.46%
GE	**10.35%**	2.17%	−7.35%	3.92%	4.74%	5.74%
GS	6.18%	2.35%	−4.92%	2.83%	5.24%	**11.62%**
HD	15.20%	**15.91%**	7.07%	5.34%	6.27%	2.12%
IBM	**8.15%**	7.77%	5.35%	2.37%	0.91%	−1.52%
INTC	**18.23%**	8.99%	5.18%	5.90%	6.18%	4.71%
JNJ	**13.45%**	9.01%	7.53%	1.81%	3.76%	0.42%
JPM	12.79%	8.25%	8.66%	−4.77%	15.72%	**16.27%**
MCD	**17.94%**	14.43%	8.37%	2.04%	4.09%	0.24%
MRK	**15.93%**	6.55%	3.60%	0.91%	2.88%	5.46%
MSFT	**13.43%**	9.95%	7.07%	5.58%	5.29%	4.45%
NKE	**18.00%**	17.10%	9.34%	0.58%	1.54%	4.68%
PFE	**8.07%**	6.51%	−0.35%	1.83%	7.07%	1.34%
PG	**9.79%**	5.72%	3.31%	0.88%	4.99%	3.93%
TRV	17.34%	12.01%	6.24%	−7.62%	**19.98%**	10.97%
UTX	**9.36%**	7.67%	2.76%	3.18%	6.52%	1.65%
UNH	9.74%	**13.21%**	7.31%	9.50%	−1.01%	2.13%
VZ	**10.23%**	9.29%	9.37%	0.28%	4.49%	5.63%
WMT	**15.20%**	6.28%	5.22%	−2.88%	5.30%	8.33%
Average	**12.59%**	10.47%	5.01%	3.78%	6.48%	5.45%
St.Dev.	4.45%	5.10%	4.38%	5.28%	4.26%	**3.99%**

2007–2017). Our proposed method's (Dow30-2007-2017) average annualized return is 12.59% and percent of successful transactions is 71.32% (Table 10), whereas BaH average annualized return is 10.47%, RSI model average annualized return is 5.01%, SMA model average annualized return is 3.78%, LSTM model average annualized return is 6.48%, and MLP regression average annualized return is 5.45%. The analyses provided for ETFs are also applicable to Dow30 performance, since similar outcomes are observed. The proposed model performed the best overall annualized return with a relatively low standart deviation. (Table 8). Fig. 7 shows the proposed model's accumulation of the capital for different Dow30 stocks. In each case, the model performance is compared against Buy&Hold during the selected period. The performance results for other Dow30 stocks also show similar characteristics. Best annualized return values in Table 8 are illustrated in bold.

In addition, during the financial crisis period the average annualized return for our proposed method (Dow30-2007-2012) is 12.83% and percent of successful transactions is 70.63% (Table 10), whereas BaH average annualized return is 6.98%, RSI model average annualized return is 1.92%, and SMA model average annualized return is 0.75%, LSTM model average annualized return is 10.82%, and MLP regression average annualized return is 9.98%. In addition, our proposed solution's average annualized return is almost twice

Table 9
Comparison of annualized returns of the proposed system (CNN-TA) with BaH, RSI, SMA,LSTM, MLP Reg. Models (DOW30 – test period: 2007–2012).

Stock	CNN-TAr	BaHr	RSIr	SMAr	LSTMr [5]	MLPr [5]
MMM	**13.42%**	3.52%	−1.58%	−6.63%	15.26%	3.24%
AXP	**31.64%**	−2.21%	−11.33%	−5.65%	27.47%	24.68%
APPL	10.03%	**36.54%**	7.54%	32.73%	20.92%	0.49%
BA	2.73%	−0.56%	0.30%	1.82%	**13.53%**	3.94%
CAT	0.22%	10.97%	−8.82%	**17.61%**	4.75%	4.56%
CVX	**19.77%**	11.87%	2.49%	−0.02%	16.31%	14.09%
CSCO	8.34%	−7.05%	0.13%	−15.38%	0.80%	**20.82%**
KO	**14.15%**	11.13%	4.67%	5.13%	5.83%	8.25%
DIS	**13.33%**	3.55%	−1.92%	−4.28%	8.47%	3.11%
XOM	**18.92%**	5.12%	4.45%	−5.17%	3.66%	9.15%
GE	**7.11%**	−9.62%	−17.51%	−3.02%	3.73%	3.20%
GS	1.47%	−15.00%	−13.65%	−4.50%	−11.25%	**6.41%**
HD	**12.01%**	4.32%	2.77%	−7.59%	5.23%	6.21%
IBM	11.61%	**15.52%**	4.46%	6.55%	1.34%	−1.14%
INTC	**18.56%**	6.42%	4.08%	0.68%	7.82%	15.83%
JNJ	**16.63%**	3.03%	4.81%	−3.09%	7.52%	2.63%
JPM	9.19%	−5.84%	5.93%	−19.40%	**33.74%**	29.12%
MCD	21.86%	**22.14%**	9.49%	0.91%	−2.18%	2.45%
MRK	**13.15%**	0.33%	−3.89%	−1.21%	6.95%	11.71%
MSFT	4.44%	−1.24%	−2.50%	0.62%	12.58%	**16.02%**
NKE	**19.12%**	16.93%	12.80%	−6.01%	11.80%	6.30%
PFE	**8.56%**	0.99%	−5.84%	−0.82%	2.92%	7.97%
PG	10.47%	3.26%	2.53%	−2.33%	**12.92%**	1.12%
TRV	**23.01%**	5.91%	5.77%	−18.90%	21.79%	15.03%
UTX	**10.65%**	4.37%	3.53%	2.39%	5.41%	3.40%
UNH	2.38%	0.22%	−1.85%	**7.59%**	−8.37%	−0.09%
VZ	14.56%	7.95%	9.59%	−3.16%	13.46%	**16.30%**
WMT	**21.82%**	6.89%	10.37%	−10.88%	12.91%	13.09%
Average	**12.83%**	6.98%	1.92%	0.75%	10.82%	9.98%
St.Dev.	7.38%	10.09%	**7.33%**	10.21%	9.74%	7.77%

as BaH annualized return during financial crisis period. Again, the proposed model achieved the best overall performance while keeping the variance low. These results show that proposed solution is more stable than the other models (Table 9). Best annualized return values in Table 9 are shown in bold.

5.7. Discussions

Consistently beating Buy and Hold strategy is challenging in a long period of time (like 10 years). Our proposed method's (Dow30-2007-2012) annualized return performed better than BaH strategy's annualized return in 24 out of 28 stocks during the out-of-sample test period (Table 9). (In the same period, Visa stock [V] did not have sufficient data points, so V is neglected. Also Dupont [DD] stock prices had data inconsistencies in finance.yahoo.com, so that was not used in the analyses.) In addition, our proposed method's (ETF-2007-2012) annualized return performed better than BaH strategy's annualized return in 9 out of 9 ETFs during the out-of-sample test period (Table 7).

When Tables 10 and 11 are analyzed, it is observed that Percent of Success for Trade Transactions are between 70% and 80%. Hence, the trades generated by the proposed model are successful (profitable) most of the time. Since the out-of-sample test period is selected as 10 years, different market conditions are observed during such a lengthy period (i.e. uptrend, downtrend, stationary). But these fluctuations in the market conditions did not affect the overall trading performance of the proposed model. As a result, the model was able to generate good profits even under deteriorating market conditions. For the most commonly traded ETFs, the proposed model (ETF-2007-2017) outperformed the Buy & Hold strategy 9 out of 9 times (Table 6) and for Dow30 stocks, the proposed model (Dow30-2007-2017) outperformed Buy & Hold 22 out of 28 times over the span of 10 years (Table 8).

Table 10
TTest results and average results of the proposed CNN-TA model for Dow30.

Performance metrics	TTest	Avg(07-12)	Avg(07-17)
Proposed CNN Strategy Annualized Return (CNN-TAr)	0.337	12.83%	12.59%
Annualized Number of Transaction (AnT)	0.007	19.1	21.3
Percent of Success (PoS)	0.556	70.63%	71.32%
Average Percent Profit Per Transactions (ApT)	0.981	0.78%	0.78%
Average Transaction Length in Days (L)	0.000	9.0	6.9
Maximum Profit Percentage in Transaction (MpT)	0.127	11.04%	8.82%
Maximum Loss Percentage in Transaction (MlT)	0.052	−18.97%	−14.33%
Idle Ratio (IdleR)	0.002	53.32%	57.11%
Sharpe Ratio (Daily)	–	0.08	0.10

Table 11
TTest results and average results of the proposed CNN-TA model for ETFs.

Performance metrics	TTest	Avg(07-12)	Avg(07-17)
Proposed CNN Strategy Annualized Return (CNN-TAr)	0.787	13.17%	13.01%
Annualized Number of Transaction (AnT)	0.298	18.8	17.3
Percent of Success (PoS)	0.982	71.44%	71.51%
Average Percent Profit Per Transactions (ApT)	0.815	0.74%	0.70%
Average Transaction Length in Days (L)	0.405	8.8	9.4
Maximum Profit Percentage in Transaction (MpT)	0.962	9.98%	10.12%
Maximum Loss Percentage in Transaction (MlT)	0.747	−19.84%	−18.51%
Idle Ratio (IdleR)	0.546	52.64%	54.04%
Sharpe Ratio (Daily)	–	0.09	0.11

5.8. Statistical significance tests

The statistical significance test results tabulated in Table 10 and Table 11 indicate the proposed model does not change its behavior neither for different asset classes (Dow30 stocks or ETFs) nor between different time periods and varying market conditions (2007–2012 and 2007–2017 time periods). For most of the evaluation metrics, the proposed model shows stable and robust operating characteristics. Even though statistically no significant differences are observed for the proposed model's trading performance under different market conditions, the proposed model performs better than traditional trading models like Buy & Hold, RSI, SMA, LSTM and MLP regression especially when the overall market is not in an uptrend.

Also, in Tables 10 and 11 some trading summary results are provided. The results are consistent not only between ETFs and Dow30 stocks, but also between varying market conditions (2007–2012) and (2007–2017) periods. The number of transactions (AnT) are between 17 and 21 in all cases indicating the model performs a trade (buy–sell pair) once every 3 weeks which is consistent with the input technical analysis resolution (6–20 days). Average trade length (L) is also 7–9 days which indicates there is also 9–14 days of idle time where the model sits on cash waiting for a trade trigger (indicated by Idle Ratio).

In addition, the statistical significance test results of proposed CNN-TA compared with BaH, LSTM and MLP are tabulated in Tables 12 and 13 . The results indicate CNN-TA trading performance is significantly better than all models over the long run (2007–2017) for both Dow30 stocks and ETFs. For 2007–2012 period, the outcome is similar, however only for the LSTM case, the outperformance of CNN-TA is not significant.

Table 12
TTest results of annualized return of Dow30 stocks.

Time interval	Performance metrics	TTest results
2007–2017	CNN-TA – BaH	0.0100241
	CNN-TA – LSTM	0.0000001
	CNN-TA – MLP	0.0000001
2007–2012	CNN-TA – BaH	0.0012111
	CNN-TA – LSTM	0.1072803
	CNN-TA – MLP	0.0490155

Table 13
TTest results of annualized return of ETFs.

Time interval	Performance metrics	TTest results
2007–2017	CNN-TA – BaH	0.0000302
	CNN-TA – LSTM	0.0010461
	CNN-TA – MLP	0.0000013
2007–2012	CNN-TA – BaH	0.0013547
	CNN-TA – LSTM	0.0755492
	CNN-TA – MLP	0.0463610

5.9. Future directions

Even though the performance of the model is promising, more improvements can still be achieved. The model might perform better if CNN structural parameters are optimized. Evolutionary algorithms for model optimization may enhance the network performance. Similarly, selecting the proper image size, window size, technical analysis optimization can improve the overall performance considerably. 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. However, 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. Similarly, based on the stock/ETF performance, a portfolio with multiple stocks/ETFs can be dynamically allocated and enhanced overall performance with less risk can be achieved.

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 [61–63], gait recognition through radar signals [64] and Malware classification [65,66]. Following these recent developments, in the near future, we might expect similar implementations in other fields to start utilizing image-based CNN.

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 Dow Jones 30 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.

Acknowledgement

This study was funded by The Scientific and Technological Research Council of Turkey (TUBITAK) under grant number 215E248.

Appendix A

A.1 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 oscillate between 0 and 100 which indicates whether the stock prices are in the "overbought" or "oversold" region. The most common usage of RSI indicator and its interpretation works as follows: If the value is over 70, the stock is considered to be in the "overbought" region. Meanwhile, if the value is under 30, the stock is assumed to be in the "oversold" region. Equation for calculating the RSI value is provided in Eq. (12):

$$RSI = 100 - \frac{100}{1 + (averagegain/averageloss)} \tag{12}$$

A.2 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. Eq. (13) shows how Williams %R value is calculated.

$$R = \frac{\max(high) - close}{\max(high) - \min(low)} * -100 \tag{13}$$

A.3 Simple moving average (SMA)

Simple moving average (SMA) shows the moving average of the prices for a given period. In its most widely accepted interpretation, the intersection of the SMA values with different interval values are used to determine the trend direction. As a result, multiple SMAs can be combined to be used together, or one single SMA value can be used in conjunction with the underlying stock, i.e. if the stock price is higher than the SMA (for instance 50 day), it is assumed that the stock is in uptrend, indicating the stock price will continue to increase (Buy trigger), whereas if the stock price is lower than the SMA, it is assumed that the stock is in downtrend, indicating the stock price will decrease (Sell trigger). Calculation of SMA is summarized in Eq. (14):

$$SMA(M, n) = \sum_{k=a+1}^{a+n} \frac{M(k)}{n} \tag{14}$$

A.4 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. Importance of the latest data is exponentially increasing in EMA calculations. Eq. (15) illustrates the calculation of EMA of the stock prices.

$$(M(t) - EMA(M, t-1, \tau)).\frac{2}{\tau + 1} + EMA(M, t-1, \tau) \tag{15}$$

A.5 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. On the other hand, the significance of the close price of stock is decreasing exponentially in EMA. Eq. (16) shows how WMA is calculated:

$$WMA(M, n) = \frac{Sum\ \ of\ \ Weighted\ \ Averages}{Sum\ \ of\ \ Weight} \tag{16}$$

A.6 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. HMA improves this reduction of the lag and gets better results when compared with EMA and WMA. Eq. (17) exhibits the calculation of HMA:

$$WMA(M, n) = WMA(2 * WMA\left(\frac{n}{2}\right) - WMA(n)), sqrt(n) \tag{17}$$

A.7 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. It can be calculated as follows:

$$(3 * EMA - 3 * EMA(EMA)) + EMA(EMA(EMA)) \tag{18}$$

A.8 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. Eqs. (19) and (20) show the calculations for CCI:

$$CCI = \frac{Typical\ \ Price - 20\ \ Period\ \ SMA\ \ of\ \ TP}{0.015 * Mean\ \ Deviation} \tag{19}$$

$$Typical\ \ Price\ \ (TP) = \frac{High + Low + Close}{3} \tag{20}$$

A.9 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. The formula of the indicator is illustrated in Eq. (21). S_u is the sum of the momentum of up days and S_d is the sum of the momentum of down days:

$$CMO = 100 * \frac{(S_u - S_d)}{(S_u + S_d)} \tag{21}$$

A.10 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. Eqs. (22) and (23) show the calculations of MACD and Signal Lines:

$$MACD\ \ Line : (12\ \ Day\ \ EMA - 26\ \ Day\ \ EMA) \tag{22}$$

$$Signal\ \ Line : 9\ \ Day\ \ EMA\ \ of\ \ MACD\ \ Line \tag{23}$$

A.11 Percentage price oscillator (PPO)

Percentage price oscillator (PPO) is similar to MACD. The calculation of the PPO and Signal Line of PPO are illustrated in Eqs. (24) and (25):

$$PPO = \frac{(12\ \ Day\ \ EMA - 26\ \ Day\ \ EMA)}{26\ \ Day\ \ EMA} * 100 \tag{24}$$

$$Signal\ \ Line : 9\ \ Day\ \ EMA\ \ of\ \ PPO \tag{25}$$

A.12 Rate of change (ROC)

Rate of change is a technical indicator that illustrates the speed of price change over a period of time. Eq. (26) shows the calculation of the formula.

$$ROC = \frac{(Latest\ \ Close - Previous\ \ Close)}{(Previous\ \ Close)} * 100 \tag{26}$$

A.13 Chaikin money flow indicator (CMFI)

Chaikin money flow (CMF) is a technical indicator that is used to measure money flow volume over a period of time. Indicator's value fluctuates between 1 and −1. If the value is closer 1, it is interpreted that buying pressure is higher. On the contrary, if the value is closer −1, it is interpreted that selling pressure is higher. Eqs. (27)–(29) illustrate the calculation of CMFI:

$$Multiplier = \frac{((Close - Low) - (High - Close))}{(High - Low)} \tag{27}$$

$$Money\ \ Flow\ \ Volume\ \ (MFV) = Volume * Multiplier \tag{28}$$

$$21\ \ Period\ \ CMF = \frac{21\ \ Period\ \ Sum\ \ of\ \ MFV}{21\ \ Period\ \ Sum\ \ of\ \ Volume} \tag{29}$$

A.14 Directional movement indicator (DMI)

Directional movement indicator is a technical indicator that shows the trend's strength and direction. It consists of three separate indicators: Average Directional Index (ADX), plus directional indicator (+DI) and minus directional indicator (−DI). DMI oscillates 0 and 100 values. Algorithm 3, Eqs. (27)–(29) show the calculation of DMI:

Algorithm 3. Calculating DMI

```
1:  procedure DMI()
2:  UpMove = Current High − Previous High
3:  DownMove = Current Low − Previous Low
4:  If(UpMove > DownMove and UpMove > 0)
5:      then return(+DMI) = UpMove,
6:      else return(+DMI) = 0
7:  If(DownMove > Upmove and DownMove > 0)
8:      then return(−DMI) = DownMove,
9:      else return(−DMI) = 0
```

$$+DI = 100 * EMA\left(\frac{+DMI}{Average\ \ True\ \ Range}\right) \tag{30}$$

$$-DI = 100 * EMA\left(\frac{-DMI}{Average\ \ True\ \ Range}\right) \tag{31}$$

$$ADX = 100 * EMA\left(Absolute\ \ Value\ \ of\ \ \left(\frac{+DI - -DI}{+DI + -DI}\right)\right) \tag{32}$$

A.15 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 0.02 and increases by 0.02 every time when EP rises in a Rising SAR. AF decreases by 0.02 every time when EP falls in a Falling SAR. Eq. (33) shows the calculation of rising parabolic SAR. Falling Parabolic SAR is calculated as in Eq. (34):

$$PSAR + Previous \quad AF(Previous \quad EP + PSAR) \tag{33}$$

$$PSAR - Previous \quad AF(PSAR - Previous \quad EP) \tag{34}$$

References

[1] 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 Syst. Appl. 55 (2016) 194–211.
[2] A. Canziani, A. Paszke, E. Culurciello, An Analysis of Deep Neural Network Models for Practical Applications, 2016 arXiv:1605.07678.
[3] C. Krauss, X.A. Do, N. Huck, Deep neural networks, gradient-boosted trees, random forests: statistical arbitrage on the s&p 500, Eur. J. Oper. Res. 259 (2) (2017) 689–702.
[4] J.-F. Chen, W.-L. Chen, C.-P. Huang, S.-H. Huang, A.-P. Chen, Financial time-series data analysis using deep convolutional neural networks, in: 2016 7th International Conference on Cloud Computing and Big Data (CCBD), IEEE, 2016, pp. 87–92.
[5] T. Fischer, C. Krauß, Deep Learning with Long Short-term Memory Networks for Financial Market Predictions, Tech. Rep., FAU Discussion Papers in Economics, 2017.
[6] F. Ganz, D. Puschmann, P. Barnaghi, F. Carrez, A practical evaluation of information processing and abstraction techniques for the internet of things, IEEE Internet Things J. 2 (4) (2015) 340–354.
[7] G.E. Box, G.M. Jenkins, G.C. Reinsel, G.M. Ljung, Time Series Analysis: Forecasting and Control, John Wiley & Sons, 2015.
[8] J.D. Hamilton, Time Series Analysis, vol. 2, Princeton University Press, Princeton, 1994.
[9] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, P. Smyth, Rule discovery from time series, AAAI (1998).
[10] M. Ramoni, P. Sebastiani, P. Cohen, Bayesian clustering by dynamics, Mach. Learn. 47 (1) (2002) 91–121.
[11] L. Cao, Support vector machines experts for time series forecasting, Neurocomputing 51 (2003) 321–339.
[12] N.K. Ahmed, A.F. Atiya, N.E. Gayar, H. El-Shishiny, An empirical comparison of machine learning models for time series forecasting, Econometr. Rev. 29 (5–6) (2010) 594–621.
[13] M. Mohandes, T. Halawani, S. Rehman, A.A. Hussain, Support vector machines for wind speed prediction, Renew. Energy 29 (6) (2004) 939–947.
[14] C.M. Arizmendi, J.R. Sanchez, N.E. Ramos, G.I. Ramos, Time series predictions with neural nets: application to airborne pollen forecasting, Int. J. Biometeorol. 37 (3) (1993) 139–144.
[15] D. Srinivasan, A. Liew, C. Chang, A neural network short-term load forecaster, Electr. Power Syst. Res. 28 (3) (1994) 227–234.
[16] I. Kaastra, M. Boyd, Designing a neural network for forecasting financial and economic time series, Neurocomputing 10 (3) (1996) 215–236.
[17] K. Kalaitzakis, G. Stavrakakis, E. Anagnostakis, Short-term load forecasting based on artificial neural networks parallel implementation, Electr. Power Syst. Res. 63 (3) (2002) 185–196.
[18] T. Kohonen, The self-organizing map, Neurocomputing 21 (1-3) (1998) 1–6.
[19] F. Mörchen, A. Ultsch, O. Hoos, Extracting interpretable muscle activation patterns with time series knowledge mining, Int. J. Knowl-Based Intell. Eng. Syst. 9 (3) (2005) 197–208.
[20] S.-C. Kuo, S.-T. Li, Y.-C. Cheng, M.-H. Ho, Knowledge discovery with SOM networks in financial investment strategy, in: Fourth International Conference on Hybrid Intelligent Systems (HIS'04), IEEE, 2004, pp. 98–103.
[21] A. Bezerianos, S. Papadimitriou, D. Alexopoulos, Radial basis function neural networks for the characterization of heart rate variability dynamics, Artif. Intell. Med. 15 (3) (1999) 215–234.
[22] Q. Li, D. Liu, J. Fang, A. Jeary, C. Wong, Damping in buildings: its neural network model and AR model, Eng. Struct. 22 (9) (2000) 1216–1223.
[23] D. Guan, W. Yuan, S.J. Cho, A. Gavrilov, Y.-K. Lee, S. Lee, Devising a context selection-based reasoning engine for context-aware ubiquitous computing middleware, in: Ubiquitous Intelligence and Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 849–857.
[24] J. Choi, D. Shin, D. Shin, Research and implementation of the context-aware middleware for controlling home appliances, IEEE Trans. Consumer Electron. 51 (1) (2005) 301–306.
[25] N. An, W. Zhao, J. Wang, D. Shang, E. Zhao, Using multi-output feedforward neural network with empirical mode decomposition based signal filtering for electricity demand forecasting, Energy 49 (2013) 279–288.
[26] B. Krollner, B. Vanstone, G. Finnie, Financial Time Series Forecasting with Machine Learning Techniques: A Survey, 2010.
[27] J.-Z. Wang, J.-J. Wang, Z.-G. Zhang, S.-P. Guo, Forecasting stock indices with back propagation neural network, Expert Syst. Appl. 38 (11) (2011) 14346–14355.
[28] Z. Liao, J. Wang, Forecasting model of global stock index by stochastic time effective neural network, Expert Syst. Appl. 37 (1) (2010) 834–841.
[29] A.-S. Chen, M.T. Leung, H. Daouk, Application of neural networks to an emerging financial market: forecasting and trading the Taiwan stock index, Comput. Oper. Res. 30 (6) (2003) 901–923.
[30] E. Guresen, G. Kayakutlu, T.U. Daim, Using artificial neural network models in stock market index prediction, Expert Syst. Appl. 38 (8) (2011) 10389–10397.
[31] 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.
[32] S. Dhar, T. Mukherjee, Performance evaluation of neural network approach in financial prediction: evidence from Indian Market, Proceedings of the International Conference on Communication and Computational Intelligence (2010).
[33] B. Vanstone, G. Finnie, T. Hahn, Creating trading systems with fundamental variables and neural networks: the Aby case study, Math. Comput. Simul. 86 (2012) 78–91.
[34] R. Aguilar-Rivera, M. Valenzuela-Rendón, J. Rodríguez-Ortiz, Genetic algorithms and Darwinian approaches in financial applications: a survey, Expert Syst. Appl. 42 (21) (2015) 7684–7697.
[35] A.M. Ozbayoglu, U. Erkut, Stock market technical indicator optimization by genetic algorithms, in: Intelligent Engineering Systems through Artificial Neural Networks, vol. 20, ASME Press, 2010.
[36] Y.-K. Kwon, B.-R. Moon, A hybrid neurogenetic approach for stock forecasting, IEEE Trans. Neural Netw. 18 (3) (2007) 851–864.
[37] O.B. Sezer, M. Ozbayoglu, E. Dogdu, A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters, Proc. Comput. Sci. 114 (2017) 473–480.
[38] C. Evans, K. Pappas, F. Xhafa, Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation, Math. Comput. Modell. 58 (5) (2013) 1249–1266.
[39] C.-F. Huang, A hybrid stock selection model using genetic algorithms and support vector regression, Appl. Soft Comput. 12 (2) (2012) 807–818.
[40] M. Pulido, P. Melin, O. Castillo, Particle swarm optimization of ensemble neural networks with fuzzy aggregation for time series prediction of the Mexican Stock Exchange, Inform. Sci. 280 (2014) 188–204.
[41] J. Wang, R. Hou, C. Wang, L. Shen, Improved v-support vector regression model based on variable selection and brain storm optimization for stock price forecasting, Appl. Soft Comput. 49 (2016) 164–178.
[42] S. Mabu, M. Obayashi, T. Kuremoto, Ensemble learning of rule-based evolutionary algorithm using multi-layer perceptron for supporting decisions in stock trading problems, Appl. Soft Comput. 36 (2015) 357–367.
[43] M. Ballings, D. Van den Poel, N. Hespeels, R. Gryp, Evaluating multiple classifiers for stock price direction prediction, Expert Syst. Appl. 42 (20) (2015) 7046–7056.
[44] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[45] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inform. Process. Syst. (2012) 1097–1105.
[46] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale video classification with convolutional neural networks, Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2014) 1725–1732.
[47] S. Lawrence, C.L. Giles, A.C. Tsoi, A.D. Back, Face recognition: a convolutional neural-network approach, IEEE Trans. Neural Netw. 8 (1) (1997) 98–113.
[48] D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber, Convolutional neural network committees for handwritten character classification, in: 2011 International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2011, pp. 1135–1139.
[49] Y. Kim, Convolutional Neural Networks for Sentence Classification, 2014 arXiv:1408.5882.
[50] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A Convolutional Neural Network for Modelling Sentences, 2014 arXiv:1404.2188.
[51] X. Ding, Y. Zhang, T. Liu, J. Duan, Deep learning for event-driven stock prediction, IJCAI (2015) 2327–2333.
[52] M. Längkvist, L. Karlsson, A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Pattern Recogn. Lett. 42 (2014) 11–24.
[53] A. Yoshihara, K. Fujikawa, K. Seki, K. Uehara, Predicting Stock Market Trends by Recurrent Deep Neural Networks, 2014, pp. 759–769.
[54] F. Shen, J. Chao, J. Zhao, Forecasting exchange rate using deep belief networks and conjugate gradient method, Neurocomputing 167 (2015) 243–253.
[55] P. Tino, C. Schittenkopf, G. Dorffner, Financial volatility trading using recurrent neural networks, IEEE Trans. Neural Netw. 12 (4) (2001) 865–874.
[56] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, Deep direct reinforcement learning for financial signal representation and trading, IEEE Trans. Neural Netw. Learn. Syst. 28 (3) (2017) 653–664, http://dx.doi.org/10.1109/TNNLS.2016.2522401.
[57] Y. LeCun, B.E. Boser, J.S. Denker, D. Henderson, R.E. Howard, W.E. Hubbard, L.D. Jackel, Handwritten digit recognition with a back-propagation network, Advances in Neural Information Processing Systems (1990) 396–404.

[58] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, et al., Learning algorithms for classification: a comparison on handwritten digit recognition, Neural Netw. 261 (1995) 276.
[59] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016 http://www.deeplearningbook.org.
[60] M.M. Mostafa, Forecasting stock exchange movements using neural networks: empirical evidence from Kuwait, Expert Syst. Appl. 37 (9) (2010) 6302–6309.
[61] Y. Zheng, Q. Liu, E. Chen, Y. Ge, J.L. Zhao, Time series classification using multi-channels deep convolutional neural networks, in: International Conference on Web-Age Information Management, Springer, 2014, pp. 298–310.
[62] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: a strong baseline, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 1578–1585.
[63] A. Le Guennec, S. Malinowski, R. Tavenard, Data augmentation for time series classification using convolutional neural networks, ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (2016).
[64] M.S. Seyfioglu, A.M. Ozbayoglu, S.Z. Gurbuz, Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities, IEEE Trans. Aerosp. Electron. Syst. PP (99) (2018), 1-1.
[65] E.K. Kabanga, C.H. Kim, Malware images classification using convolutional neural network, J. Comput. Commun. 6 (01) (2017) 153.
[66] S. Yue, Imbalanced Malware Images Classification: A CNN Based Approach, 2017 arXiv:1708.08042.

Omer Berat Sezer received the B.S. degree in Electrical and Electronics Engineering from Middle East Technical University NCC, Ankara, Turkey, in 2009, with an emphasis on telecommunications and computers; M.S. degree in Electrical and Electronics Engineering from Middle East Technical University, Ankara, Turkey, in 2013,with an emphasis on computer networks. He is currently a Ph.D. candidate at Department of Computer Engineering of TOBB University of Economics and Technology, in Ankara, Turkey and he is also working as a senior researcher and software engineer at The Scientific and Technological Research Council of Turkey – Space Technologies Research Institute, in Ankara, Turkey. His research interests are machine learning, Internet of things, big data and time series data analytics.

Ahmet Murat Ozbayoglu graduated from the Department of Electrical Engineering at Middle East Technical University, Ankara, Turkey in 1991 with an emphasis on telecommunications and computers. Then he continued his education as a graduate student in Missouri University of Science and Technology (formerly known as University of Missouri-Rolla (UMR)) in Engineering Management. He obtained his M.Sc. and Ph.D. degrees from UMR in 1993 and 1996, respectively. After graduation, he joined Beyond Inc., St. Peters, MO as a product development engineer and consultant. In 2001 he joined MEMC Electronic Materials Inc., St. Peters, MO as a software project engineer, programmer and analyst. During his work in Beyond Inc. and MEMC, he worked on software data automation projects for silicon wafer manufacturing systems. In 2005, he went back to academia by joining the Department of Computer Engineering of TOBB University of Economics and Technology, in Ankara, Turkey. His research interests are machine learning, pattern recognition, big data, computational intelligence, machine vision.