CNNpred: CNN-based stock market prediction using a diverse set of variables

Ehsan Hoseinzade[a], Saman Haratizadeh[a,*]

[a] *Faculty of New Sciences and Technologies, University of Tehran, North Kargar Street, 1439957131 Tehran, Iran*

Abstract

Feature extraction from financial data is one of the most important problems in market prediction domain for which many approaches have been suggested. Among other modern tools, convolutional neural networks (CNN) have recently been applied for automatic feature selection and market prediction. However, in experiments reported so far, less attention has been paid to the correlation among different markets as a possible source of information for extracting features. In this paper, we suggest a CNN-based framework, that can be applied on a collection of data from a variety of sources, including different markets, in order to extract features for predicting the future of those markets. The suggested framework has been applied for predicting the next day's direction of movement for the indices of S&P 500, NASDAQ, DJI, NYSE, and RUSSELL based on various sets of initial variables. The evaluations show a significant improvement in prediction's performance compared to the state of the art baseline algorithms.

Keywords: `Stock markets prediction`, Deep learning, Convolutional neural networks, CNN, Feature extraction

---

*Corresponding author

Email addresses: `hoseinzadeehsan@ut.ac.ir` (Ehsan Hoseinzade), `haratizadeh@ut.ac.ir` (Saman Haratizadeh)

Preprint submitted to Journal of Expert Systems with Applications *March 18, 2019*

1. Introduction

Financial markets are considered as the heart of the world's economy in which billions of dollars are traded every day. Clearly, a good prediction of future behavior of markets would be extremely valuable in various areas. Stock markets play an important role in Economic growth, (Beck & Levine, 2004) so, analyzing their behavior and predicting their future can be very helpful in achieving economic goals. Another application of stock market prediction can be found in stock market trading systems, that usually consist of several modules for prediction, risk analysis and trading strategy. The goal of a trading module is to create a portfolio of stocks that maximizes the overall return regarding the risk of stocks in that portfolio (Markowitz, 1952). However, a prediction module focuses on the sub-problem of predicting the future of the markets that can be a very valuable piece of information in the process of stock trading. So, the performance of this module and by extent the whole trading system is influenced considerably by the quality of predictions that happen in the prediction module. In fact, without a reliable prediction, it is almost impossible to have an excellent trading system.

Machine learning techniques have proved to be useful for making such predictions. Artificial neural networks (ANN) and support vector machine (SVM) are the most common algorithms that have been utilized for this purpose (Guresen et al., 2011; Kara et al., 2011; Wang & Wang, 2015). Statistical methods, random forests (Khaidem et al., 2016), linear discriminant analysis, quadratic discriminant analysis, logistic regression and evolutionary computing algorithms, especially genetic algorithm (GA), (Hu et al., 2015b; Brown et al., 2013; Hu et al., 2015a; Atsalakis & Valavanis, 2009) are among other tools and techniques that have been applied for feature extraction from raw financial data and/or making predictions based on a set of variables (Ou & Wang, 2009; Ballings et al., 2015).

Deep learning (DL) is a class of modern tools that is suitable for automatic features extraction and prediction (LeCun et al., 2015). In many domains, such as machine vision and natural language processing, DL methods have been shown that are able to gradually construct useful complex features from raw data or simpler variables (He et al., 2016; LeCun et al., 2015). Since the behavior of stock markets is complex, nonlinear and noisy, extracting features that are informative enough for making predictions is a core challenge, and DL seems to be a promising approach to that. Algorithms like deep multilayer perceptron (MLP) (Yong et al., 2017), restricted Boltzmann machine (RBM) (Cai et al., 2012; Zhu et al., 2014), long short-term memory (LSTM) (Chen et al., 2015; Fischer & Krauss, 2018), autoencoder (AE) (Bao et al., 2017) and convolutional neural network (CNN) (Gunduz et al., 2017; Di Persio & Honchar, 2016) are famous deep learning algorithms utilized to predict stock markets.

It is important to pay attention to the diversity of the variables that can be used for making predictions. The raw price data, technical indicators which come out of historical data, other markets with a connection to the target market, exchange rates of currencies, oil price and many other variables can be useful for a market prediction task. Unfortunately, it usually is not a straightforward task to aggregate such a diverse set of information in a way that an automatic market prediction algorithm can use them. So, most of the existing works in this field have limited themselves to a set of technical indicators representing a single market's recent history (Kim, 2003; Zhang & Wu, 2009).

Another important subject in the field is automatic feature extraction. Since the initial variables are defined to be used by human experts, they are simple and even if they were chosen by a finance expert who has enough knowledge and experience in this domain, they may not be the best possible choices for making predictions by machines. In other words, an automatic approach to stock market prediction ideally is one that can extract useful features from different variables that seem beneficial for market prediction, train a prediction model based on those extracted features and finally make predictions using the resulted model. The focus of this paper is on the first phase of this process, that is to design a model for extracting features from several variables that contain information from historical records of relevant markets. This data include initial basic variables such as raw historical prices, technical indicators or fluctuation of those variables in the past days. Regarding the diversity of the input space and possible complexity of the feature space that may be required for a good prediction, a deep learning algorithm like CNN seems to be a promising approach for such a feature extraction problem.

To the best of our knowledge, convolutional neural network, CNN, has been applied in a few studies for stock market prediction (Gunduz et al., 2017; Di Persio & Honchar, 2016). Periso & Honchar (Di Persio & Honchar, 2016) used a CNN which took a one-dimensional input for making prediction only based on the history of closing prices while ignoring other possible variables like technical indicators. Gunduz et al. (Gunduz et al., 2017) took advantage of a CNN which was capable of using technical indicators as well for each sample. However, it was unable to consider the correlation which could exist between stock markets as another possible source of information. In addition, structure of used CNN was inspired by previous works in Computer Vision, while there are fundamental differences between Computer Vision and Stock market prediction. Since in stock market prediction variables interaction are radically different from pixel's interaction with each other, using 3×3 or 5×5 filters in convolutional layer may not be the best option. It seems cleverer to design filters of CNN specially for financial data instead of papers in Computer Vision.

We develop our framework based on CNN due to its proven capabilities in other domains as well as mentioned successful past experiments reported in market prediction domain. As a test case, we will show how CNN can be applied in our suggested framework, that we call CNNpred, to capture the possible correlations among different variables for extracting combined features from a diverse set of input data from five major U.S. stock market indices: S&P 500, NASDAQ, Dow Jones Industrial Average, NYSE and RUSSELL, as well as other variables including exchange rate of currencies, future contracts, price of commodities, important indices of markets around the world, price of major companies in U.S. market, and treasury bill rates. Furthermore, the filters are designed in a way that is compatible with financial characteristics of variables.

The main contributions of this work can be summarized as follows:

- Aggregating several variables in a CNN-based framework for feature extraction and market prediction. Since financial markets behavior is affected by many factors, it is important to gather related information as much as possible. Our initial variable set covers different aspects of stock related variables pretty well and basically, it can be easily extended to cover other possible variables.

- To our knowledge, this is the first work suggesting a CNN which takes a 3-dimensional tensor to aggregate and align a diverse set of variables as input and then trains the network in a way that extracts useful features for predicting each of the pertinent stock markets.

The rest of this paper is organized as follows: In section 2, related works and researches are presented. Then, in section 3, we introduce a brief background on related techniques in the domain. In section 4, the proposed method is presented in details followed by introduction of various utilized variables in section 5. Experimental setting and results are reported in section 6. In section 7, we discuss the results and there is a conclusion in section 8.

2. Related works

Different methods in stock prediction domain can be categorized into two groups. The first class includes algorithms try to improve the performance of prediction by enhancing the prediction models, while the second class of algorithms focuses on improving the features based on which the prediction is made.

In the first class of the algorithms that focus on the prediction models, a variety of tools have been used, including Artificial Neural Networks (ANN), naive Bayes, SVM, and random forests. The most popular tool for financial prediction seems to be ANN (Krollner et al., 2010). In (Kara et al., 2011), ten technical indicators were passed to ANN and SVM in order to forecast directional movement of the Istanbul Stock Exchange (ISE) National 100 Index. Authors found that ANN's ability in prediction is significantly better than SVM.

Feedforward shallow ANNs are popular types of ANNs that usually are trained by back-propagation algorithm (Hecht-Nielsen, 1992; Hagan & Menhaj, 1994). While obstacles like the noisy behavior of stock markets make ANNs learning process to converge to suboptimal solutions, sometimes local search algorithms like genetic algorithm (GA) or simulated annealing (SA) take responsibility of finding initial or final optimal weights for neural networks (Kim & Han, 2000; Qiu et al., 2016; Qiu & Song, 2016). In (Qiu et al., 2016), authors used GA and SA to find initial weights of an ANN, and then back-propagation algorithm is used to train the network. This hybrid approach outperformed the standard ANN-based methods in prediction of Nikkei 225 index return.

Authors of (Zhong & Enke, 2017) applied PCA and two variations of it in order to extract better features. A collection of different variables was used as input data while an ANN was utilized for prediction of S&P 500. The results showed an improvement of the prediction using the features generated by PCA compared to the other two variations of that. Another study on the effect of variables on the performance of prediction models has been reported in (Patel et al., 2015). This research used common tools including ANN, SVM, random forest and naive Bayes for predicting directional movement of Indian indices and stocks. This research showed that mapping the data from a space of ten technical variables to another feature space that represents trends of those variables improved performance of prediction.

The simplicity of shallow models can avoid them from achieving effective mappings from input space to successful predictions. So, with regards to availability of large amounts of data and emerging effective learning methods for training deep models, researchers have recently turned to such approaches for market prediction. An important aspect of deep models is that they are usually able to extract rich sets of features from the raw data and make predictions based on them. So, from this point of view, deep models usually combine both phases of feature extraction and prediction in a single phase.

Deep ANNs, that are basically neural networks with more than one hidden layers, are among the first deep methods used in the domain. In (Moghaddam et al., 2016), authors predicted NASDAQ prices based on the historical price of four and nine days ago. ANNs with different structures were tested and the experiments proved the superiority of deep ANNs over shallow ones. In (Arévalo et al., 2016), authors used a deep ANN with five hidden layers to forecast Apple Inc.'s stock price. Outputs showed up to about 65% directional accuracy.

In (Chong et al., 2017), authors draw an analogy between different data representation methods including RBM, Auto-encoder and PCA applied on raw data with 380 variables. The resulting representations were then fed to a deep ANN for prediction. The results showed that none of the data representation methods had superiority over the others in all of the tested experiments.

Recurrent Neural Networks are a kind of neural networks that are specially designed to have internal memory that enables them to extract historical features and make predictions based on them. So, they seem fit for the domains like market prediction. LSTM is one of the most popular kinds of RNNs. In (Nelson et al., 2017), technical indicators were fed to an LSTM in order to predict the direction of stock prices in the Brazilian stock market. According to the reported results, LSTM outperformed MLP.

Convolutional Neural Network is another deep learning algorithm applied in stock market prediction after LSTM and MLP while its ability to extract efficient features has been proven in many other domains as well. In (Di Persio & Honchar, 2016), CNN, LSTM, and MLP were applied to the historical data of close prices of S&P 500 index. Results showed that CNN outperformed LSTM and MLP.

Based on some reported experiments, the way the input data is designed to be fed and processed by CNN has an important role in the quality of the extracted feature set and the final prediction. For example, CNN was used in (Gunduz et al., 2017) in which data of 100 companies in Borsa Istanbul were utilized to produce technical indicators and time-lagged variables. Then, variables were clustered into different groups and similar variables were put beside each other. The experiments showed that the performance of CNN achieve F-measure of 56% and outperformed baseline algorithms including a CNN with random arrangement of variables.

Table 1 summarizes explained papers in terms of initial variable set, feature extraction algorithm, and prediction method. There is a tendency toward deep learning models in recent publications, due to the capability of these algorithms in automatic feature extraction from raw data. However, most of the researchers have used only technical indicators or historical price data of one market for prediction while there are various variables which could enhance accuracy of prediction of stock market. In this paper, we are going to introduce a novel CNN-based framework that is designed to aggregate several variables in order to automatically extract features to predict direction of stock markets.

3. Background

Before presenting our suggested approach, in this section, we review the convolutional neural network that is the main element of our framework.

3.1. Convolutional Neural Network

LeCun and his colleagues introduced convolutional neural networks in 1995 (LeCun et al., 1995; Gardner & Dorling, 1998). CNN has many layers which could be categorized into input layer, convolutional layer, pooling layer, fully connected layer, and output layer.

3.1.1. Convolutional layer

The convolutional layer is supposed to do the convolution operation on the data. In fact, input could be considered as a function, filter applied to that is another function and convolution operation is an algorithm used to measure changes caused by applying a filter on the input. Size of a filter shows the coverage of that filter. Each filter utilizes a shared set of weights to perform the convolutional operation. Weights are updated during the process of training.

Author/year	Target Data	Variables Set	Feature Extraction	Prediction Method
(Kara et al., 2011)	Borsa Istanbul BIST 100 Index	technical indicator	ANN	ANN SVM
(Patel et al., 2015)	4 Indian stocks & indices	technical indicator	ANN	ANN-SVM RF-NB
(Qiu et al., 2016)	Nikkei 225 index	financial indicator macroeconomic data	ANN	GA+ANN SA+ANN
(Qiu & Song, 2016)	Nikkei 225 index	technical indicator	ANN	GA+ANN
(Nelson et al., 2017)	Brazil Bovespa 5 stocks	technical indicator	LSTM	LSTM
(Di Persio & Honchar, 2016)	S&P 500 index	price data	MLP-RNN-CNN wavelet+CNN	MLP RNN CNN
(Moghaddam et al., 2016)	NASDAQ	price data	ANN-DNN	ANN-DNN
(Arévalo et al., 2016)	AAPL Inc.	3 extracted features	DNN	DNN
(Zhong & Enke, 2017)	S&P 500 index	various variables	PCA	ANN
(Chong et al., 2017)	Korea KOSPI 38 stock returns	price data	PCA-RBM AE	DNN
(Gunduz et al., 2017)	Borsa Istanbul BIST 100 stocks	technical indicator temporal variable	Clustering CNN	CNN
Our method	U.S. 5 major indices	various variables	3D representation of data+CNN	CNN

Table 1: Summary of explained papers

Let's posit input of layer $l-1$ is a $N \times N$ matrix and $F \times F$ convolutional filters are used. Then, input of layer l is calculated according to Eq 1. Fig 1 shows applying a filter to the input data in order to get the value of $v_{1,1}$ in the next layer. Usually, output of each filter is passed through an activation function before entering the next layer. Relu (Eq 2) is a commonly used nonlinear activation function.

$$v_{i,j}^{l} = \delta(\sum_{k=0}^{F-1} \sum_{m=0}^{F-1} w_{k,m} V_{i+k,j+m}^{l-1}) \tag{1}$$

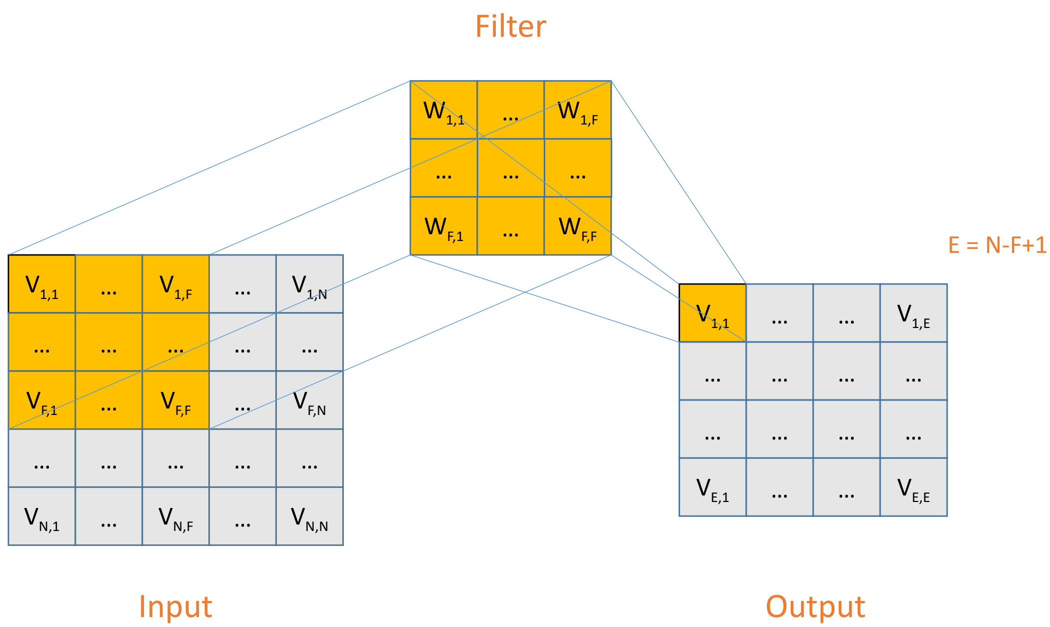

Figure 1: Applying filter($F \times F$) to the input data($N \times N$) in order to get value of $V_{1,1}$ in the next layer

In the Eq 1, $v_{i,j}^{l}$ is the value at row i, column j of layer l, $w_{k,m}$ is the weight at row k, column m of filter and δ is the activation function.

$$f(x) = max(0, x) \tag{2}$$

3.1.2. Pooling layer

Pooling layer is responsible for subsampling the data. This operation, not only reduces the computational cost of the learning process, but also is a way of handling the overfitting problem in CNN. Overfitting is a situation that arises when a trained model makes too fit to the training data, such that it cannot generalize to the future unseen data. It has a connection to the number of parameters that are learned and the amount of data that the prediction model is learned from. Deep models, including CNNs, usually have many parameters. So, they are prone to overfitting more than shallow models. Some methods have been suggested to avoid overfitting. Using pooling layers in CNNs can help to reduce the risk of overfitting. All the values inside a pooling window are converted to only one value. This transformation reduces the size of the input of the following layers, and hence, reduces the number of the parameters that must be learned by the model, that in turn, lowers the risk of overfitting. Max pooling is the most common type of pooling in which the maximum value in a certain window is chosen.

3.1.3. Fully connected layer

At the final layers of a CNN, there is a MLP network which is called its fully connected layer. It is responsible for converting extracted features in the previous layers to the final output. The relation between two successive layers is defined by Eq 3

$$v_i^j = \delta(\sum_k v_k^{j-1} w_{k,i}^{j-1}) \tag{3}$$

In Eq 3, v_i^j is the value of neuron i at the layer j, δ is activation function and weight of connection between neuron k from layer $j-1$ and neuron i from layer j are shown by $w_{k,i}^{j-1}$.

3.2. Dropout

In addition to pooling, we have also used another technique called dropout that was first developed for training deep neural networks. The idea behind the dropout technique is to avoid the model from too much learning of the training data. So, in each learning cycle during the training phase, each neuron has a chance equal to some *dropout rate*, not to be trained in that cycle. This avoids the model from being too flexible, and so, helps the learning algorithm to converge to a model that is not too much fit to the training data, and instead, can be generalized well for predicting the unlabeled future data (Hinton et al., 2012; Srivastava et al., 2014).

4. Proposed CNN: CNNpred

CNN has many parameters including the number of layers, number of filters in each layer, dropout rate, size of filters in each layer, and initial representation of input data which should be chosen wisely to get the desired outcomes. Although 3×3 and 5×5 filters are quite common in image processing domain, we think that size of each filter should be determined according to financial interpretation of variables and their characteristics rather than just following previous works in image processing. Here we introduce the architecture of CNNpred, a general CNN-based framework for stock market prediction. CNNpred has two variations that are referred to as 2D-CNNpred and 3D-CNNpred. We explain the framework in four major steps: representation of input data, daily feature extraction, durational feature extraction, and final prediction.

Representation of input data: CNNpred takes information from different markets and uses it to predict the future of those markets. As we mentioned, 2D-CNNpred and 3D-CNNpred take different approaches for constructing prediction models. The goal of the first approach is to find a general model for mapping the history of a market to its future fluctuations and by "general model we mean a model that is valid for several markets. In other words, we assume that the true mapping function from the history to the future is the one that is correct for many markets. For this goal, we need to design a single model that is able to predict the future of a market based on its own history, however, to extract the desired mapping function, that model needs to be trained by samples from different markets. 2D-CNNpred follows this general approach, but in addition to modeling the history of a market as the input data, it also uses a variety of other variables as well. In 2D-CNNpred, all this information is aggregated and fed to a specially designed CNN as a two-dimensional tensor, and that's why it is called 2D-CNNpred. On the other hand, the second approach, 3D-CNNpred, assumes that different models are needed for making predictions in different markets, but each prediction model can use information from the history of many markets. In other words, 3D-CNNpred, unlike 2D-CNNpred, does not train a single prediction model that can predict the future of each market given its own historical data, but instead, it extracts features from the historical information of many markets and uses them to train a separate prediction model for each market. The intuition behind this approach is that the mechanisms that dictate the future behavior of each market differ, at least slightly, from other markets. However, what happens in the future in a market, may depend on what happens inside and outside that certain market. Based on this intuition, 3D-CNNpred uses a tensor with three dimensions, to aggregate historical information from various markets and feed it to a specially designed CNN to train a prediction model for each market. Although the structure of the model is the same for all the markets, the data that is used for training is different for each market. In other words, in 3D-CNNpred, each prediction model can see all the available information as input but is trained to predict the future of a certain market based on that input. One can expect that 3D-CNNpred, unlike 2D-CNNpred, is able to combine information from different markets into high-level features before making predictions. Fig 2 shows a schema of how data is represented and used in CNNpred's variations.

Daily feature extraction: Each day in the historical data is represented by a series of variables like opening and closing prices. The traditional approach to market prediction is to analyze these variables for example in the form of candlesticks, probably by constructing higher-level features based on them, in order to predict the future behavior of the market. The idea behind the design of the first layer of CNNpred comes from this observation. In the first step of both

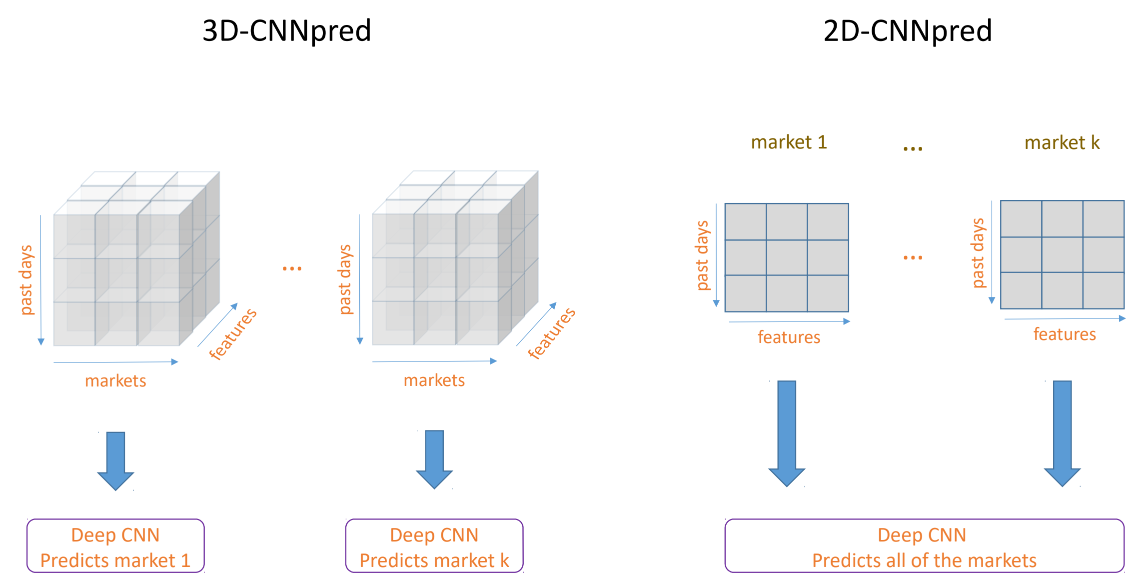

Figure 2: The structure of input data in two variations of CNNpred

variations of CNNpred, there is a convolutional layer whose task is to combine the daily variables into higher-level features for representing every single day of the history.

Durational feature extraction: Some other useful information for predicting the future behavior of a market comes from studying the behavior of the market over time. Such a study can give us information about the trends that appear in the market's behavior and find patterns that can predict the future based on them. So it is important to combine variables of consecutive days of data to gather high-level features representing trends or reflecting the market's behavior in certain time intervals. Both 2D-CNNpred and 3D-CNNpred data have layers that are supposed to combine extracted features in the first layer and produce even more sophisticated features summarizing the data in some certain time interval.

Final prediction: At the final step, the features that are generated in the previous layers are converted to a one-dimensional vector using a flattening operation and this vector is fed to a fully connected layer that maps the features to a prediction.

In the next two sections, we will explain the general design of 2D-CNNpred and 3D-CNNpred as well as how they have been adopted for the dataset that we have used in the specific experiments performed in this paper. In our experiments, we have used data from 5 different indices. Each index has 82 variables that means each day of the history of a market is represented by 82 variables. The 82 gathered variables are selected in a way that forms a complete variable set and consist of technical indicators, big U.S. companies, commodities, exchange rate of currencies, future contracts, world's stock indices, and other variables. The length of the history is 60 days that is for each prediction, the model can use information from 60 last days.

4.1. 2D-CNNpred

Representation of input data: As we mentioned before, the input to the 2D-CNNpred is a two-dimensional matrix. The size of the matrix depends on the number of variables that represent each day, as well as the number of days back into the history that is used for making a prediction. If the input used for prediction consists of d days each represented by f variables then the size of input tensor will be $d \times f$.

Daily feature extraction: To extract daily features in 2D-CNNpred, $1 \times$ *number of initial variables* filters are utilized. Each of those filters covers all the daily variables and can combine them into a single higher-level feature. 2D-CNNpred can construct different combinations of primary variables using this layer. It is also possible for the network to drop useless variables by setting their corresponding weights in thr filters equal to zero. So, this layer works as an initial feature extraction/feature selection module. Fig 3 represents applying a simple filter on the input data.

Durational feature extraction: While the first layer of 2D-CNNpred extracts features out of primary daily variables, the following layers combine extracted

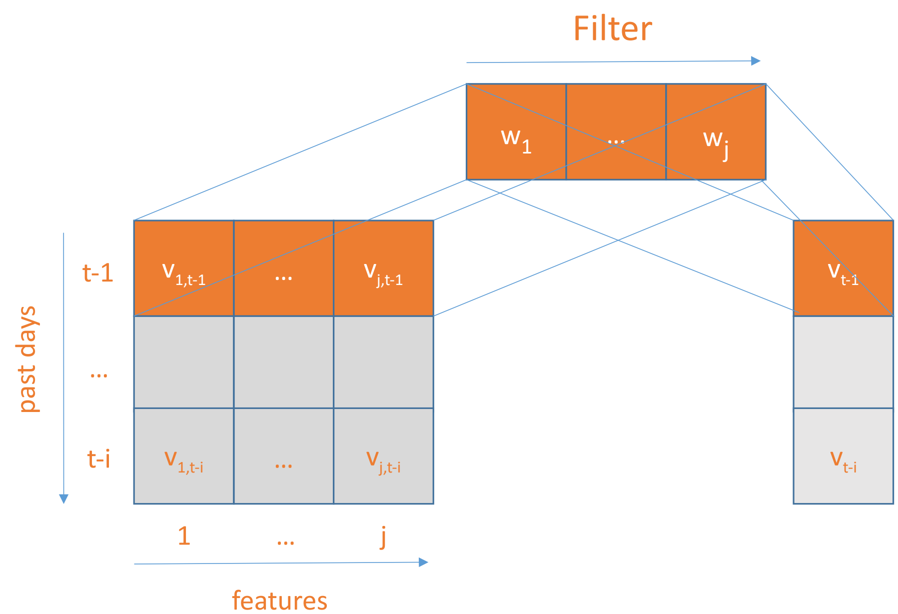

Figure 3: Applying a 1×*number of variables* filter to 2D input tensor.

features of different days to construct higher-level features for aggregating the available information in certain durations. As the first layer, these succeeding layers use filters for combining lower level features from their input to higher-level ones. 2D-CNNpred uses 3×1 filters in the second layer. Each of those filters covers three consecutive days, a setting that is inspired by the observation that most of the famous candlestick patterns like Three Line Strike and Three Black Crows try to find meaningful patterns in three consecutive days (Nison, 1994; Bulkowski, 2012; Achelis, 2001). We take this as a sign of the potentially useful information that can be extracted from a time window of three consecutive time unites in the historical data. The third layer is a pooling layer that performs a 2×1 max pooling, that is a very common setting for the pooling layers. After this pooling layer and in order to aggregate the information in longer time intervals and construct even more complex features, 2D-CNNpred uses another convolutional layer with 3×1 filters followed by another pooling layer just like the first one.

Final prediction: features generated by the last pooling layer are flattened into a final feature vector. This feature vector is then converted to a final prediction through a fully connected layer. Sigmoid (Eq 4) is the activation function that we choose for this layer. Since the output of sigmoid is a number in [0-1] interval, the prediction that is made by 2D-CNNpred for a market can be interpreted as the probability of an increase in the price of that market for the next day, that is a valuable piece of information. Clearly, it is rational to put more money on a stock that has a higher probability of going up. On the other hand, stocks with a low probability of going up are good candidates for short selling. However, in our experiments, we discretize the output to either 0 or 1, whichever is closer to the prediction.

$$f(x) = \frac{1}{1 + \exp(x)} \tag{4}$$

A sample configuration of 2D-CNNpred: As we mentioned before, the input we used for each prediction consists of 60 days each represented by 82 variables. So, the input to the 2D-CNNpred is a matrix of 60 by 82. The first convolutional layer uses eight 1×82 filters after which there are two convolutional layers with eight 3×1 filters, each followed by a layer of 2×1 max-pooling. The final flattened feature vector contains 104 features that are fed to the fully connected layer to produce the final output. Fig 4 shows a graphical visualization of the described process.

4.2. 3D-CNNpred

Representation of input data: 3D-CNNpred, unlike 2D-CNNpred, uses a three-dimensional tensor to represent data. The reason is that each sample that is fed to 3D-CNNpred contains information from several markets. So, the initial daily variables, the days of the historical record and the markets from which the data is gathered form the three dimensions of the input tensor. Suppose our dataset consists of i different markets, k variables for each of these markets and

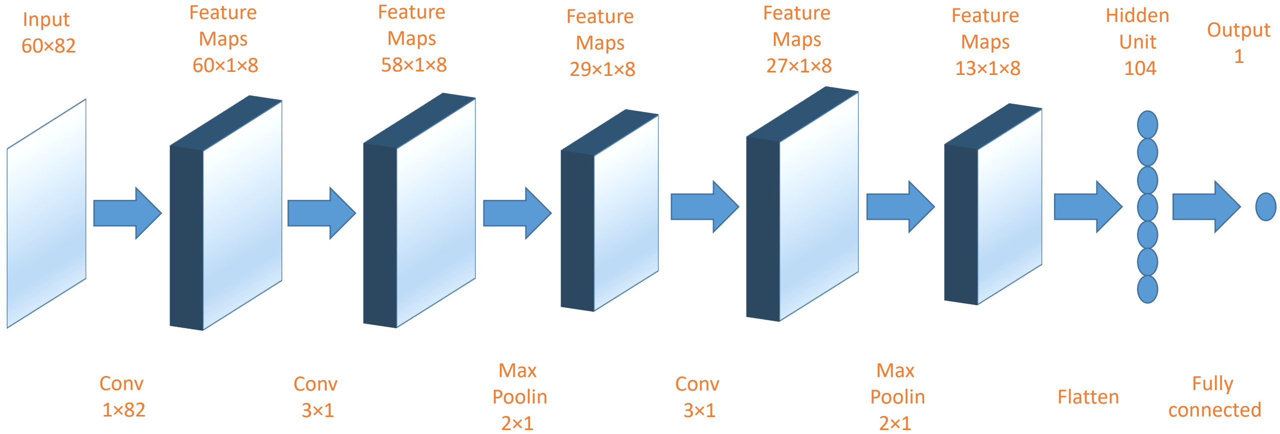

Figure 4: Graphical Visualization of 2D-CNNpred

our goal is to predict day t based on past j days. Fig 5 shows how one sample of the data would be represented.

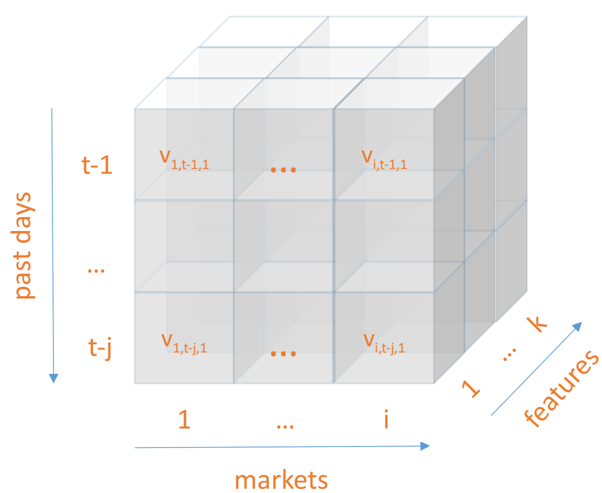

Figure 5: Representation of input data in 3D-CNNpred based on k primary variables, i related markets and j days before the day of prediction

Daily feature extraction: The first layer of filters in 3D-CNNpred is defined as a set of 1×1 convolutional filters, while the primary variables are represented along the depth of the tensor. Fig 6 shows how a 1×1 filter works. This layer of filters is responsible for combining subsets of basic variables that are available through the depth of the input tensor into a set of higher-level features. The input tensor is transformed by this layer into another tensor whose width and height is the same but its depth is equal to the number of 1×1 convolutional filters of layer one. Same as 2D-CNNpred, the network has the capability to act as a feature selection/extraction algorithm.

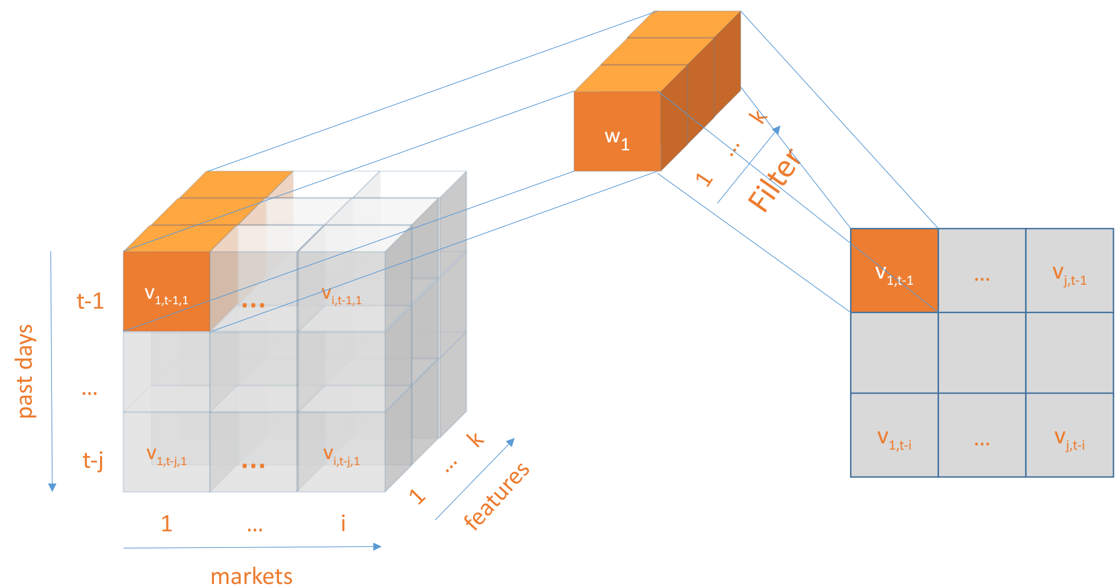

Figure 6: Applying a 1×1 filter to the first part of the 3D input tensor.

Durational feature extraction: In addition to daily variables, 3D-CNNpred's input data provides information about other markets. Like 2D-CNNpred, the next four layers are dedicated to extracting higher-level features that summarize the fluctuation patterns of the data in time. However, in 3D-CNNpred, this is done over a series of markets instead of one. So, the width of the filters in the second convolutional layer is defined in a way that covers all the pertinent markets. Same as 2D-CNNpred and motivated by the same mentioned reason, the height of filters is selected to be 3 so as to cover three consecutive time units. Using this setting, the size of filters in the second convolutional layer is $3\times$*number of markets*. The next three layers, like those of 2D-CNNpred, are defined as a 2×1 max pooling layer, another 3×1 convolutional layer followed by a final 2×1 max pooling layer.

Final prediction: Same as 2D-CNNpred, here in 3D-CNNpred the output of the durational feature extraction phase is flattened and used to produce the final results.

A sample configuration of 3D-CNNpred: In our experiments, the input to the 3D-CNNpred is a matrix of 60 by 5 with depth of 82. The first convolutional layer uses eight filters to perform 1×1 convolutional operation, after which there is one convolutional layer with eight 3×5 filters followed by a 2×1 max pooling layer. Then, another convolutional layer utilizes eight 3×1 filters, again followed by a 2×1 max-pooling layer generates the final 104 features. In the end, a fully connected layer converts 104 neurons to 1 neuron and produces the final output. Fig 7 shows a graphical visualization of the process.

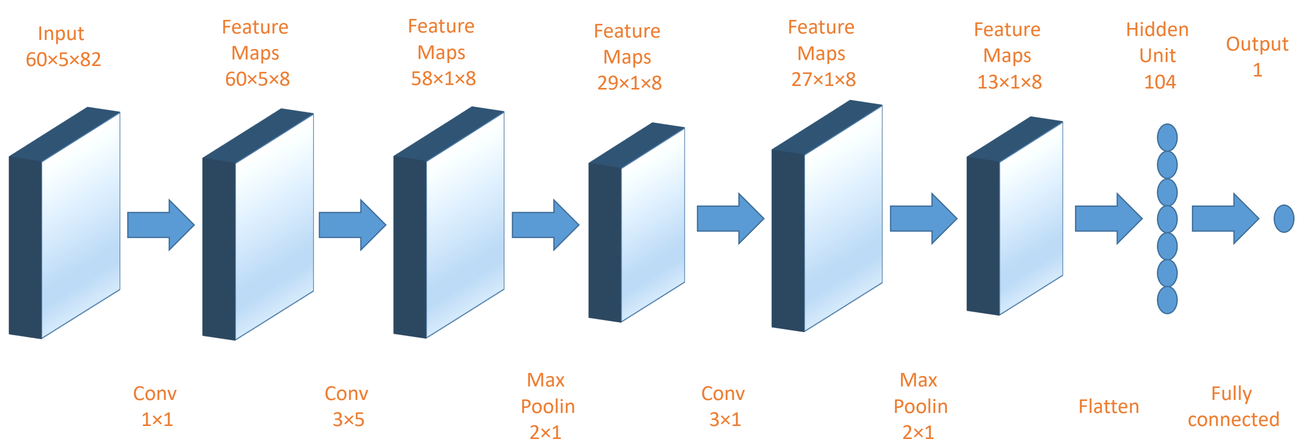

Figure 7: Graphical Visualization of 3D-CNNpred

5. Initial variable set for each market

As we mentioned before, our goal is to develop a model for prediction of the direction of movements of stock market prices or indices. We applied our approach to predict the movement of indices of S&P 500, NASDAQ, Dow Jones Industrial Average, NYSE, and RUSSELL. For this prediction task, we use 82 variables for representing each day of each index. Some of these variables are index-specific while the rest are general economic variables and are replicated for every index in the data set. This rich set of variables could be categorized in eight different groups that are primitive variables, technical indicators, world stock market indices, the exchange rate of U.S. dollar to the other currencies, commodities, data from big companies of the U.S. markets, future contracts and other useful variables. Some of these variables are important as they represent mechanisms that naturally affect the stock markets, directly or indirectly. Some other variables, on the other hand, are useful as they provide clues or signs that can help the system to predict the short-term future of the markets, even if they do not represent causal relations. We briefly explain different groups of our variable set here and more details about them can be found in Appendix I.

Primitive variables: Close price and the day of week for which the prediction is supposed to be made are primitive variables used in this work.

Technical indicators: Technical analysts use technical indicators which are extracted from historical data of stocks prices and trading information to analyze short-term movement of prices. They are quite common in stock market research. The moving averages are examples of this type of variables.

World stock markets: Usually, stock markets all over the world have interaction with each other because of the phenomenon of globalization of the economy. This connection would be more appreciated when we consider time difference in various countries which makes it possible to gain information about the future of a country's market by monitoring other countries markets (Brzeszczyński & Ibrahim, 2019; Ibrahim & Brzeszczynski, 2014). For instance, effect of other countries stock markets like China, Japan, and South Korea on the U.S. market.

The exchange rate of U.S. dollar: There are multinational companies that import their needs from other countries or export their product to other countries. So, the fluctuation of U.S. dollar to the other currencies like the Canadian dollar and European Euro affects the profit of these companies. When this fluctuation in profit is announced, demand for the stock of these companies and by extent their stock price changes. Domestic companies stock prices are also affected by change in the demand of multinational companies. Consequently, stock prices are affected by exchange rate of currencies in either direct or indirect manner (Bahmani-Oskooee & Sohrabian, 1992; Aggarwal, 2003).

Commodities: Another factor that can be used for predicting behavior of stock markets is price of commodities like gold, silver, oil, wheat and so on. This kind of information can reflect a view of the global market. Researchers have shown that there is a link between commodities and stock markets, especially after the 2007-2008 financial crisis (Creti et al., 2013). In addition, Commodities have become an important part of portfolios as well as stocks. This means that the information about the prices of commodities can be useful in prediction of the fluctuations of stock prices.

Big U.S. Companies: Stock market indices are calculated based on different stocks. Each stock carries a weight in this calculation that matches its share in the market. In other words, big companies are more important than small ones in prediction of stock market indices. Examples of that could be return of Exxon Mobil Corporation and Apple Inc.

Futures contracts: Futures contracts are contracts in which one side of agreement is supposed to deliver stocks, commodities and so on in the future. These contracts show expected value of the merchandise in the future. Investors tend to buy stocks that have higher expected value than their current value. For instance, S&P 500 Futures, DJI Futures, and NASDAQ Futures prices could affect current price of S&P 500 and other indices.

Other useful variables: According to different papers, other variables including Treasury bill rates, The term and default spreads have shown to be useful in stock market prediction (Zhong & Enke, 2017; Niaki & Hoseinzade, 2013; Enke

& Thawornwong, 2005).

6. Experimental settings and results

In this section, we describe the settings that are used to evaluate the models, including datasets, parameters of the networks, evaluation methodology and baseline algorithms. Then, the evaluation results are reported.

6.1. Data gathering and preparation

The datasets used in this work include daily direction of close of S&P 500 index, NASDAQ Composite, Dow Jones Industrial Average, NYSE Composite, and RUSSELL 2000. Each sample has 82 variables that already have been explained and its assigned label is determined according to the Eq 5. It is worth mentioning that for each index only technical indicators and primitive variables are unique and the other variables, like big U.S. companies or price of commodities, are common between different indices.

$$target = \begin{cases} 1 & Close_{t+1} > Close_t \\ 0 & \text{else} \end{cases} \tag{5}$$

Where $Close_t$ refers to the closing price at day t.

This data are from the period of Jan 2010 to Nov 2017. The first 60% of the data is used for training the models, the next 20% forms the validation data and the last 20% is the test data.

Different variables could have various ranges. It is usually confusing for learning algorithms to handle variables with different ranges. Generally, the goal of data normalization is to map the values of all variables to a single common range, and it usually improves the performance of the prediction model. We use Eq 6 for normalizing the input data, where x_{new} is normalized variable vector, x_{old} is the original variable vector, $\bar{x}$ and σ are the mean and the standard deviation of original variable.

$$x_{new} = \frac{x_{old} - \bar{x}}{\sigma} \tag{6}$$

6.2. Evaluation methodology

Evaluation metrics are needed to compare results of our method with the other methods. Accuracy is one of the common metrics have been used in this area. However, in an imbalanced dataset, it may be biased toward the models that tend to predict the more frequent class. To address this issue, we report the Macro-Averaged-F-Measure that is the mean of F-measures calculated for each of the two classes (Gunduz et al., 2017; Özgür et al., 2005).

6.3. Parameters of network

Numerous deep learning packages and software have been developed. In this work, Keras (Chollet et al., 2015) was utilized to implement CNN. The activation function of all the layers is RELU except the last one which is Sigmoid. Each convolutional layer consists of 8 filters. Adam (Kingma & Ba, 2014) with batch size of 128 was used to train the network.

6.4. Baseline algorithms

We compare the performance of the suggested methods with that of the algorithms applied in the following researches. In all the baseline algorithms the same settings reported in the original paper were used.

- The first baseline algorithm is the one reported in (Zhong & Enke, 2017). In this algorithm, the initial data is mapped to a new feature space using PCA and then the resulting representation of the data is used for training a shallow ANN for making predictions.

- The second baseline is based on the method suggested in (Kara et al., 2011), in which the technical indicators are used to train a shallow ANN for prediction.

- The third baseline algorithm is a CNN with two-dimensional input (Gunduz et al., 2017). First, the variables are clustered and reordered accordingly. The resulting representation of the data is then used by a CNN with a certain structure for prediction.

Algorithm	Explanation
3D-CNNpred	Our method
2D-CNNpred	Our method
PCA+ANN (Zhong & Enke, 2017)	PCA as dimension reduction and ANN as classifier
Technical (Kara et al., 2011)	Technical indicators and ANN as classifier
CNN-cor (Gunduz et al., 2017)	A CNN with mentioned structure in the paper

Table 2: Description of used algorithms

Market \Model	Technical	CNN-cor	PCA+ANN	2D-CNNpred	3D-CNNpred
S&P 500	0.4469	0.3928	0.4237	0.4914	0.4837
DJI	0.415	0.39	0.4283	0.4975	0.4979
NASDAQ	0.4199	0.3796	0.4136	0.4944	0.4931
NYSE	0.4071	0.3906	0.426	0.4885	0.4751
RUSSELL	0.4525	0.3924	0.4279	0.5002	0.4846

Table 3: Average F-measure of different algorithms

6.5. Results

In this section, results of five different experiments are explained. Since one of the baseline algorithms uses PCA for dimension reduction, the performance of the algorithm with different number of principal components is tested. In order to make the situation equal for the other baseline algorithms, these algorithms are tested several times with the same condition. Then, average F-measure of the algorithms are compared. More details about used notations are in Table 2.

Table 3 summarizes the results for the baseline algorithms as well as our suggested models on S&P 500 index, Dow Jones Industrial Average, NASDAQ Composite, NYSE Composite, and RUSSELL 2000 historical data in terms of F-measure. The difference between baseline algorithms with 2D-CNNpred and 3D-CNNpred is statistically significant. The best performance of algorithms in different indices is also reported in table 4.

6.6. Trading Simulation

Ideally a market prediction system can be used as a module in a trading system, and one can expect that better accuracy in prediction can lead to higher profit in trading. In the last section we observed that the suggested framework outperformed other modern market prediction systems. Here we present some

Market \Model	Technical	CNN-cor	PCA+ANN	2D-CNNpred	3D-CNNpred
S&P 500	0.5627	0.5723	0.5165	0.5408	0.5532
DJI	0.5518	0.5253	0.5392	0.5562	0.5612
NASDAQ	0.5487	0.5498	0.5312	0.5521	0.5576
NYSE	0.5251	0.5376	0.5306	0.5472	0.5592
RUSSELL	0.5665	0.5602	0.5438	0.5463	0.5787

Table 4: Best F-measure of various algorithms

experiments in which we used the CNNpred system as the prediction subsystem of a simple trading system. Clearly the performance of the whole system depends on the way the predictions are used for trading. The trading strategy that is used is as follows: Each of the prediction algorithms is executed several times and their average prediction for the probability of the price going up in day t is calculated. If this value is higher than 0.5, then the predicted label for day t is *up*, otherwise it is *down*. When the predicted label for the next day is *up* the trading system fully invests on that index and holds the shares until some day with a *down* label comes in which situation the system sells all its shares and engages in a short selling process. In this trading strategy, every single prediction of the prediction module affects the trading's performance as well as the final amount of profit. Two commonly used performance measures, Sharpe ratio and certainty-equivalent (CEQ) return(DeMiguel et al., 2007), are used to evaluate the performance of the trading in our experiments. In our experiments, we also take into account the transaction costs as well. Investors usually have to pay transaction costs to their broker, that is an important factor affecting their net return. While transaction costs varies between %0 to %0.25, %0.1 seems to be a reasonable rate (Brzeszczyński & Ibrahim, 2019). Tables 5, 6 show the results of Sharpe ratio and CEQ return of CNNpred as well as other baseline algorithms and buy and hold strategy, with and without transaction costs. In calculating CEQ return, risk aversion is 1. Table 7 shows value of investing $1 in both versions of CNNpred as well as buy and hold strategy at the end of the test period.

Strategy	Rate of costs	S&P 500	DJI	NASDAQ	NYSE	RUSSELL
Buy and hold	%0	0.1056	0.1472	0.1347	0.0753	0.0739
	%0.1	0.1050	0.1465	0.1343	0.0747	0.0736
Technical	%0	0.1056	0.1472	0.1395	0.0753	0.07053
	%0.1	0.1050	0.1465	0.1386	0.0747	0.07022
PCA+ANN	%0	0.1056	0.1472	0.1347	0.0753	-0.003
	%0.1	0.1050	0.1465	0.1343	0.0747	-0.0033
CNN-cor	%0	-0.1155	-0.1574	-0.1422	-0.085	-0.0798
	%0.1	-0.1143	-0.1561	-0.1413	-0.0838	-0.079
2D-CNNpred	%0	0.1422	0.1703	0.1163	0.1039	0.08039
	%0.1	0.1392	0.1668	0.1155	0.1012	0.07952
3D-CNNpred	%0	0.1413	0.1344	0.1642	0.0830	0.0910
	%0.1	0.1386	0.1301	0.1622	0.0822	0.0902

Table 5: Sharpe ratio of various algorithms

Strategy	Rate of costs	S&P 500	DJI	NASDAQ	NYSE	RUSSELL
Buy and hold	%0	0.0004946	0.0006716	0.0008389	0.0003598	0.0005738
	%0.1	0.0004925	0.0006687	0.0008360	0.0003568	0.0005708
Technical	%0	0.0004955	0.0006716	0.000869	0.0003598	0.0005459
	%0.1	0.0004925	0.0006687	0.0008659	0.0003568	0.000543
PCA+ANN	%0	0.0004955	0.0006716	0.0008389	0.0003598	-0.00006
	%0.1	0.0004925	0.0006687	0.0008360	0.0003568	-0.00006
CNN-cor	%0	-0.00002	-0.00003	-0.00002	0.00002	-0.00003
	%0.1	-0.00002	-0.00003	-0.00002	0.00002	-0.00003
2D-CNNpred	%0	0.0006681	0.000776	0.0007234	0.0004988	0.0006265
	%0.1	0.0006615	0.0007694	0.0007176	0.0004924	0.0006201
3D-CNNpred	%0	0.000664	0.0006129	0.001023	0.0003974	0.0007127
	%0.1	0.0006576	0.0006069	0.001016	0.0003915	0.0007064

Table 6: CEQ return of various algorithms

Strategy	Rate of costs	S&P 500	DJI	NASDAQ	NYSE	RUSSELL
Buy and hold	%0	1.1794	1.2387	1.2985	1.1338	1.2134
	%0.1	1.1784	1.2378	1.2975	1.1328	1.2124
2D-CNNpred	%0	1.2378	1.2740	1.2595	1.1808	1.2312
	%0.1	1.2356	1.2718	1.2575	1.1787	1.2291
3D-CNNpred	%0	1.2364	1.2191	1.3606	1.1456	1.2604
	%0.1	1.2343	1.2170	1.3585	1.1445	1.2582

Table 7: Value of investing $1 in various strategies at the end of the test period

7. Discussion

It is obvious from the results that both 2D-CNNpred and 3D-CNNpred statistically outperform the baseline algorithms. The difference between F-measure of our model and baseline algorithm which uses only ten technical indicators is obvious. A possible reason for that could be related to the information insufficiency of those ten technical indicators. However, using more initial variables and incorporating a PCA module, which is a famous feature extraction algorithm, did not improve the results as expected. The reason for failure of these two baseline approaches may be the fact that they use shallow ANNs that has only one hidden layer and a limited power in feature extraction and prediction compared to deep CNN models. This observation demonstrates that adding more basic variables is not enough by itself without improving the model that processes the information for feature extraction and prediction. Our framework has two advantages over these two baseline algorithms that have led to its superiority in performance: First, it uses a rich set of features containing useful information for stock prediction. Second, it uses a deep learning algorithm that extracts sophisticated features out of primary ones.

The next baseline algorithm was CNN-Cor which had the worst results among all the tested algorithms. CNN's ability in feature extraction depends on wisely selection of its parameters in a way that fits the problem for which it is supposed to be applied. With regards to the fact that both 2D-CNNpred and CNN-Cor used the same variable set and they were trained almost in the same way, poor results of CNN-Cor compared to 2D-CNNpred is possibly the result of the design of the 2D-CNN. Generally, the idea of using 3×3 and 5×5 filters for every application of CNN seems skeptical. The fact that these kinds of filters are popular in computer vision does not guarantee that they would work well in stock market prediction as well. In fact, prediction with about 9% lower F-measure on average in comparison to the 2D-CNNpred showed that designing the structure of CNN is an important challenge in applying CNNs for stock market prediction. A poorly designed CNN can adversely influence the results and make CNN's performance even worse than that of a shallow ANN.

Finally, CNNpred was tested as a part of a stock market trading system to give us an intuition about its effect on trading performance, in terms of standard evaluation measures for trading strategies. Although, it seems clear that a good market prediction module can lead to a higher performance in trading, it is not clear how much it can contribute to the net return that a real trading system will achieve. Our experiments show that using the predictions of CNNpred as a base for trading strategy of a trading system leads to good results in terms of Sharpe ratio and CEQ return measures, in most of the tested indices. Also to see the effect of transaction costs on the performance of the trading system, it was evaluated against a buy and hold trading system. Buy and hold is a passive trading strategy that transaction costs almost does not affect its Sharpe ratio and CEQ return. As expected, increasing the rate of transaction costs for both 2D-CNNpred and 3D-CNNpred resulted in a small decrease in Sharpe ratio and CEQ return since a portion of investor's money was paid to the broker. However, both CNNpred trading systems significantly outperform the buy and hold trading system in presence of trading costs in most of the test markets. These observations show that CNNpred framework can be a good candidate to be used as the prediction module of real trading systems.

8. Conclusion

The noisy and nonlinear behavior of prices in financial markets makes prediction in those markets a difficult task. A better prediction can be gained by having better variables. In this paper, we tried to use a wide collection of information, including historical data from the target market, commodities, exchange rate of currencies, and information from other possibly correlated stock markets. Also, two variations of a deep CNN-based framework were introduced and applied to extract higher-level features from that rich set of initial variables.

The suggested framework, CNNpred, was tested to make predictions in S&P 500, NASDAQ, DJI, NYSE, and RUSSELL. Final results showed the significant superiority of two versions of CNNpred over the state of the art baseline algorithms. CNNpred was able to improve the performance of prediction in all the five indices over the baseline algorithms by about 3% to 11%, in terms of F-measure. In addition to confirming the usefulness of the suggested approach, these observations also suggest that designing the structures of CNNs for the stock prediction problems is possibly a core challenge that deserves to be further studied.

Although the main purpose of this paper was to predict directional movements of stock markets, CNNpred was succesfully used in a trading system and the achieved results were a clear sign that further investigation of CNNpred with the aim of being utilized in a trading system can be a promising direction for research.

Appendix I. Description of variables

The list of variables from different categories used as initial variable set representing each sample:

#	Variable	Description	Type	Source / Calculation
1	Day	which day of week	Primitive	Pandas
2	Close	Close price	Primitive	Yahoo Finance
3	Vol	Relative change of volume	Technical Indicator	TA-Lib
4	MOM-1	Return of 2 days before	Technical Indicator	TA-Lib
5	MOM-2	Return of 3 days before	Technical Indicator	TA-Lib
6	MOM-3	Return of 4 days before	Technical Indicator	TA-Lib
7	ROC-5	5 days Rate of Change	Technical Indicator	TA-Lib
8	ROC-10	10 days Rate of Change	Technical Indicator	TA-Lib
9	ROC-15	15 days Rate of Change	Technical Indicator	TA-Lib
10	ROC-20	20 days Rate of Change	Technical Indicator	TA-Lib
11	EMA-10	10 days Exponential Moving Average	Technical Indicator	TA-Lib
12	EMA-20	20 days Exponential Moving Average	Technical Indicator	TA-Lib
13	EMA-50	50 days Exponential Moving Average	Technical Indicator	TA-Lib
14	EMA-200	200 days Exponential Moving Average	Technical Indicator	TA-Lib
15	DTB4WK	4-Week Treasury Bill: Secondary Market Rate	Other	FRED
16	DTB3	3-Month Treasury Bill: Secondary Market Rate	Other	FRED
17	DTB6	6-Month Treasury Bill: Secondary Market Rate	Other	FRED
18	DGS5	5-Year Treasury Constant Maturity Rate	Other	FRED
19	DGS10	10-Year Treasury Constant Maturity Rate	Other	FRED
20	DAAA	Moody's Seasoned Aaa Corporate Bond Yield	Other	FRED
21	DBAA	Moody's Seasoned Baa Corporate Bond Yield	Other	FRED
22	TE1	DGS10-DTB4WK	Other	FRED
23	TE2	DGS10-DTB3	Other	FRED
24	TE3	DGS10-DTB6	Other	FRED
25	TE5	DTB3-DTB4WK	Other	FRED
26	TE6	DTB6-DTB4WK	Other	FRED
27	DE1	DBAA-BAAA	Other	FRED
28	DE2	DBAA-DGS10	Other	FRED
29	DE4	DBAA-DTB6	Other	FRED
30	DE5	DBAA-DTB3	Other	FRED
31	DE6	DBAA-DTB4WK	Other	FRED
32	CTB3M	Change in the market yield on U.S. Treasury securities at 3-month constant maturity, quoted on investment basis	Other	FRED
33	CTB6M	Change in the market yield on U.S. Treasury securities at 6-month constant maturity, quoted on investment basis	Other	FRED
34	CTB1Y	Change in the market yield on U.S. Treasury securities at 1-year constant maturity, quoted on investment basis	Other	FRED
35	Oil	Relative change of oil price(WTI), Oklahoma	Commodity	FRED
36	Oil	Relative change of oil price(Brent)	Commodity	Investing.com
37	Oil	Relative change of oil price(WTI)	Commodity	Investing.com
38	Gold	Relative change of gold price (London market)	Commodity	FRED
39	Gold-F	Relative change of gold price futures	Commodity	Investing.com
40	XAU-USD	Relative change of gold spot U.S. dollar	Commodity	Investing.com
41	XAG-USD	Relative change of silver spot U.S. dollar	Commodity	Investing.com
42	Gas	Relative change of gas price	Commodity	Investing.com
43	Silver	Relative change of silver price	Commodity	Investing.com
44	Copper	Relative change of copper future	Commodity	Investing.com
45	IXIC	Return of NASDAQ Composite index	World Indices	Yahoo Finance
46	GSPC	Return of S&P 500 index	World Indices	Yahoo Finance
47	DJI	Return of Dow Jones Industrial Average	World Indices	Yahoo Finance
48	NYSE	Return of NY stock exchange index	World Indices	Yahoo Finance
49	RUSSELL	Return of RUSSELL 2000 index	World Indices	Yahoo Finance
50	HSI	Return of Hang Seng index	World Indices	Yahoo Finance
51	SSE	Return of Shang Hai Stock Exchange Composite index	World Indices	Yahoo Finance

#	Variable	Description	Type	Source / Calculation
52	FCHI	Return of CAC 40	World Indices	Yahoo Finance
53	FTSE	Return of FTSE 100	World Indices	Yahoo Finance
54	GDAXI	Return of DAX	World Indices	Yahoo Finance
55	USD-Y	Relative change in US dollar to Japanese yen exchange rate	Exchange Rate	Yahoo Finance
56	USD-GBP	Relative change in US dollar to British pound exchange rate	Exchange Rate	Yahoo Finance
57	USD-CAD	Relative change in US dollar to Canadian dollar exchange rate	Exchange Rate	Yahoo Finance
58	USD-CNY	Relative change in US dollar to Chinese yuan exchange rate	Exchange Rate	Yahoo Finance
59	USD-AUD	Relative change in US dollar to Australian dollar exchange rate	Exchange Rate	Investing.com
60	USD-NZD	Relative change in US dollar to New Zealand dollar exchange rate	Exchange Rate	Investing.com
61	USD-CHF	Relative change in US dollar to Swiss franc exchange rate	Exchange Rate	Investing.com
62	USD-EUR	Relative change in US dollar to Euro exchange rate	Exchange Rate	Investing.com
63	USDX	Relative change in US dollar index	Exchange Rate	Investing.com
64	XOM	Return of Exon Mobil Corporation	U.S. Companies	Yahoo Finance
65	JPM	Return of JPMorgan Chase & Co.	U.S. Companies	Yahoo Finance
66	AAPL	Return of Apple Inc.	U.S. Companies	Yahoo Finance
67	MSFT	Return of Microsoft Corporation	U.S. Companies	Yahoo Finance
68	GE	Return of General Electric Company	U.S. Companies	Yahoo Finance
69	JNJ	Return of Johnson & Johnson	U.S. Companies	Yahoo Finance
70	WFC	Return of Wells Fargo & Company	U.S. Companies	Yahoo Finance
71	AMZN	Return of Amazon.com Inc.	U.S. Companies	Yahoo Finance
72	FCHI-F	Return of CAC 40 Futures	Futures	Investing.com
73	FTSE-F	Return of FTSE 100 Futures	Futures	Investing.com
74	GDAXI-F	Return of DAX Futures	Futures	Investing.com
75	HSI-F	Return of Hang Seng index Futures	Futures	Investing.com
76	Nikkei-F	Return of Nikkei index Futures	Futures	Investing.com
77	KOSPI-F	Return of Korean stock exchange Futures	Futures	Investing.com
78	IXIC-F	Return of NASDAQ Composite index Futures	Futures	Investing.com
79	DJI-F	Return of Dow Jones Industrial Average Futures	Futures	Investing.com
80	S&P-F	Return of S&P 500 index Futures	Futures	Investing.com
81	RUSSELL-F	Return of RUSSELL Futures	Futures	Investing.com
82	USDX-F	Relative change in US dollar index futures	Exchange Rate	Investing.com

Table 8: Description of used variables

References

Achelis, S. B. (2001). *Technical Analysis from A to Z*. McGraw Hill New York.

Aggarwal, R. (2003). Exchange rates and stock prices: A study of the us capital markets under floating exchange rates, .

Arévalo, A., Niño, J., Hernández, G., & Sandoval, J. (2016). High-frequency trading strategy based on deep neural networks. In *International conference on intelligent computing* (pp. 424–436). Springer.

Atsalakis, G. S., & Valavanis, K. P. (2009). Surveying stock market forecasting techniques–part ii: Soft computing methods. *Expert Systems with Applications*, *36*, 5932–5941.

Bahmani-Oskooee, M., & Sohrabian, A. (1992). Stock prices and the effective exchange rate of the dollar. *Applied economics*, *24*, 459–464.

Ballings, M., Van den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, *42*, 7046–7056.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, *12*, e0180944.

Beck, T., & Levine, R. (2004). Stock markets, banks, and growth: Panel evidence. *Journal of Banking & Finance*, *28*, 423–442.

Brown, M. S., Pelosi, M. J., & Dirska, H. (2013). Dynamic-radius species-conserving genetic algorithm for the financial forecasting of dow jones index stocks. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition* (pp. 27–41). Springer.

Brzeszczyński, J., & Ibrahim, B. M. (2019). A stock market trading system based on foreign and domestic information. *Expert Systems with Applications*, *118*, 381–399.

Bulkowski, T. N. (2012). *Encyclopedia of candlestick charts* volume 332. John Wiley & Sons.

Cai, X., Hu, S., & Lin, X. (2012). Feature extraction using restricted boltzmann machine for stock price prediction. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on* (pp. 80–83). IEEE volume 3.

Chen, K., Zhou, Y., & Dai, F. (2015). A lstm-based method for stock returns prediction: A case study of china stock market. In *Big Data (Big Data), 2015 IEEE International Conference on* (pp. 2823–2824). IEEE.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, *83*, 187–205.

Creti, A., Joëts, M., & Mignon, V. (2013). On the links between stock and commodity markets' volatility. *Energy Economics*, *37*, 16–28.

DeMiguel, V., Garlappi, L., & Uppal, R. (2007). Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy? *The review of Financial studies*, *22*, 1915–1953.

Di Persio, L., & Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International Journal of Circuits, Systems and Signal Processing*, *10*, 403–413.

Enke, D., & Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with applications*, *29*, 927–940.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*, 654–669.

Gardner, M. W., & Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences. *Atmospheric environment*, *32*, 2627–2636.

Gunduz, H., Yaslan, Y., & Cataltepe, Z. (2017). Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations. *Knowledge-Based Systems*, *137*, 138–148.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, *38*, 10389–10397.

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, *5*, 989–993.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception* (pp. 65–93). Elsevier.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, .

Hu, Y., Feng, B., Zhang, X., Ngai, E., & Liu, M. (2015a). Stock trading rule discovery with an evolutionary trend following model. *Expert Systems with Applications*, *42*, 212–222.

Hu, Y., Liu, K., Zhang, X., Su, L., Ngai, E., & Liu, M. (2015b). Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, *36*, 534–551.

Ibrahim, B. M., & Brzeszczynski, J. (2014). How beneficial is international stock market information in domestic stock market trading? *The European Journal of Finance*, *20*, 201–231.

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, *38*, 5311–5319.

Khaidem, L., Saha, S., & Dey, S. R. (2016). Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003*, .

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*, 307–319.

Kim, K.-j., & Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert systems with Applications*, *19*, 125–132.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, .

Krollner, B., Vanstone, B., & Finnie, G. (2010). Financial time series forecasting with machine learning techniques: A survey, .

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*, 436.

LeCun, Y., Bengio, Y. et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*, 1995.

Markowitz, H. (1952). Portfolio selection. *The journal of finance*, *7*, 77–91.

Moghaddam, A. H., Moghaddam, M. H., & Esfandyari, M. (2016). Stock market index prediction using artificial neural network. *Journal of Economics, Finance and Administrative Science*, *21*, 89–93.

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with lstm neural networks. In *Neural Networks (IJCNN), 2017 International Joint Conference on* (pp. 1419–1426). IEEE.

Niaki, S. T. A., & Hoseinzade, S. (2013). Forecasting s&p 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, *9*, 1.

Nison, S. (1994). *Beyond candlesticks: New Japanese charting techniques revealed* volume 56. John Wiley & Sons.

Ou, P., & Wang, H. (2009). Prediction of stock market index movement by ten data mining techniques. *Modern Applied Science*, *3*, 28.

Özgür, A., Özgür, L., & Güngör, T. (2005). Text categorization with class-based and corpus-based keyword selection. In *International Symposium on Computer and Information Sciences* (pp. 606–615). Springer.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, *42*, 259–268.

Qiu, M., & Song, Y. (2016). Predicting the direction of stock market index movement using an optimized artificial neural network model. *PloS one*, *11*, e0155133.

Qiu, M., Song, Y., & Akagi, F. (2016). Application of artificial neural network for the prediction of stock market returns: The case of the japanese stock market. *Chaos, Solitons & Fractals*, *85*, 1–7.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*, 1929–1958.

Wang, J., & Wang, J. (2015). Forecasting stock market indexes using principle component analysis and stochastic time effective neural networks. *Neurocomputing*, *156*, 68–78.

Yong, B. X., Rahim, M. R. A., & Abdullah, A. S. (2017). A stock market trading system using deep neural network. In *Asian Simulation Conference* (pp. 356–364). Springer.

Zhang, Y., & Wu, L. (2009). Stock market prediction of s&p 500 via combination of improved bco approach and bp neural network. *Expert systems with applications*, *36*, 8849–8854.

Zhong, X., & Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, *67*, 126–139.

Zhu, C., Yin, J., & Li, Q. (2014). A stock decision support system based on dbns. *Journal of Computational Information Systems*, *10*, 883–893.