Technical analysis forecasting and evaluation of stock markets: the probabilistic recovery neural network approach

Andreas Maniatopoulos*, Alexandros Gazis and Nikolaos Mitianoudis

Department of Electrical and Computer Engineering,
Democritus University of Thrace,
Xanthi, 67100, Greece
Fax: +30-2541079015
Fax: +30-2541079569
Email: amaniato@ee.duth.gr
Email: agazis@ee.duth.gr
Email: nmitiano@ee.duth.gr
*Corresponding author

Abstract: The market efficiency theory suggests that stock market pricing reflects all publicly available information regarding a given stock. To outperform the market, a shareholder must study the market's price volume patterns and predict human behaviour and tendencies. Except for conventional approaches based on fundamental or technical analysis, new tools are currently proposed using big data and artificial intelligence. This publication analyses and evaluates four commonly used deep-learning artificial neural network models. Then, it proposes a new method by adopting the 'probabilistic recovery' algorithmic approach. The dataset used consists of 501 unique company names based on real data derived from US Dow Jones. This method closely follows the market's behaviour, providing daily upwards-downwards data trends. The proposed system can be used as a tool for technical analysis regarding the prediction accuracy of trading strategies, providing approximately 60% future movements' accuracy, over 90% relative price prediction and annual investment return slightly over 60%.

Keywords: technical analysis; probablistic neural network; neural networks; stock market prediction; stock market forecast; stock market dynamics; stock market neural networks forecasting; algorithmic trading; finance generative adversarial networks; finance convolutional neural networks; CNNs; fully connected neural networks; recurrent neural networks; RNNs; technical indicators; decision making; trading strategies.

Reference to this paper should be made as follows: Maniatopoulos, A., Gazis, A. and Mitianoudis, N. (xxxx) 'Technical analysis forecasting and evaluation of stock markets: the probabilistic recovery neural network approach', *Int. J. Economics and Business Research*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Andreas Maniatopoulos completed his integrated Bachelor’s and Master’s in Electrical and Computer Engineering in 2018 from the Democritus University of Thrace, Greece. Since then, he is a PhD candidate in Artificial Intelligence and Topology Optimization. He was an engineer intern in the Civil Aviation Authority of Mytilene, Greece in 2017 and a team leader in the ‘Network Forensics’ exercise, led by the Greek National Cyber Defense in 2018. He is a research assistant in Signal Processing Lab from November 2018, a teaching assistant since January 2019, and an information technology administrator since October 2019 in the Democritus University of Thrace. Lastly, since 2020, he is a lab moderator at the iGEM Competition (Biotechnology) in Alexandroupolis, Greece, and an AI engineer for ADwork, a technology driven advertising agency. He was also an engineer for OiS, in Dörpen, Lower Saxony (Germany) in July 2020 and his research interests include Artificial Intelligence, high performance computing, systems optimisation, image and signal processing.

Alexandros Gazis received his Diploma in Electronic and Computer Engineering and his MSc in Microelectronics and Computer Systems from the Democritus University of Thrace, Greece in 2016 and 2018 respectively. Since 2018, he is a PhD candidate at the same university, in the field of Computer Science and a member of its ‘Operating Systems and Middleware for Pervasive Computing and Wireless Sensor Networks’ research group. Moreover, he is a teaching assistant and a lab demonstrator, supervised by Assistant Professor Eleftheria Katsiri. He is a member of the Technical Chamber of Greece and he works as a software engineer specialising in core banking systems and mainframe development. He has published articles on artificial intelligence, big data, web data analytics, remote sensing, and neural networks. His research focuses on the internet of things via wireless sensor networks, cloud computing, and middleware development for pervasive computing.

Nikolaos Mitianoudis received his Diploma in Electronic and Computer Engineering from the Aristotle University of Thessaloniki, Greece in 1998. He received his MSc in Communications and Signal Processing from the Imperial College London, UK in 2000 and PhD in Audio Source Separation using Independent Component Analysis from the Queen Mary, University of London, UK in 2004. Between 2003 and 2009, he was a research associate at the Imperial College London, UK working on the Data Information Fusion-Defense Technology Centre project ‘Applied Multi-Dimensional Fusion’, sponsored by General Dynamics UK and QinetiQ. From 2009 until 2010, he was an academic assistant at the International Hellenic University. Since 2010, he is with the Electrical and Computer Engineering Department at the Democritus University of Thrace, Greece, where he currently serves as an Associate Professor in Audio and Image Processing. He also serves as an associate editor at *IEEE Trans. on Image Processing* from 2018–2021 and at *MDPI Journal of Imaging*. His research interests include machine learning, deep learning, computer vision, music information retrieval and blind source separation/extraction.

This paper is a revised and expanded version of a paper entitled [title] presented at [name, location and date of conference].

1 Introduction

Efficient interpretation and inference of stock market tendencies are a very important assets for modern analysts. More specifically, there are many stock markets worldwide, the largest of which are located in New York, Tokyo, Shanghai, and Europe. A typical example of a dataset used in the field of economic analysis is the stock market, i.e., a secured public market. On the one hand, scientists are presented with high-frequency and high volume datasets comprising an intriguing case study for preprocessing and mining the noisy data detected by Ntakaris et al. (2018). Within the stock market dynamics, investors act on their opinion regarding the course of the company and the world. On the other hand, investing companies are interested in forecasting stock price fluctuations as it addresses their need to make strategically informed decisions as suggested in Li and Ma (2010) and Iuhasz et al. (2012).

Askari and Rafae (2019) highlight that, as these studies suggest financial markets may have predictable behaviour, as mentioned in thus, they can be modelled and studied using technical analysis for investments (Sharma et al., 2021). In many cases, this is achieved via the use of software that outperforms traditional economic models (Borovykh et al., 2018).

As a result, modelling the market has absorbed huge quantities of time and effort. Algorithmic trading (AT) is an example of one of these models that, mainly tries to exploit the market latency, making thousands of transactions in a second, when traders see fit (Scholtus et al., 2014; Huang, 2018). These AT models, whilst they can produce a profit, can only be implemented by large investor groups as their cost makes them inaccessible to the regular user. At the same time, AT narrows spreads, reduces adverse selection, and reduces trade related price discovery. Also, AT causes an improvement in the frequency of triangular arbitrage opportunities and autocorrelation of high frequency returns. However, whilst improving informational efficiency by speeding up price discovery, it imposes higher adverse selection costs on slower traders. For more on this read (Chaboud et al., 2014). At the same time, AT can be used for market-making, in a way that a human is physically unable to do (Kirilenko and Lo, 2013; Lo, 2016).

Recent areas of advancement in big data technology focus on expanding marketing activities (Nuseir, 2018), using pattern and graph tracking predictions (Jeon et al., 2018; Kusuma et al., 2019), cryptocurrency price prediction (Nakano et al., 2018; Gopal and Senthilkumar, 2020), high performance computing (Malakar et al., 2018), network and system resources (Gazis and Katsiri, 2019), stock market analysis (Chong et al., 2017), applied mathematics (Pouyanfar et al., 2018), healthcare (Priyadarshini et al., 2020), environment monitoring (Tsekouras et al., 2018), transportation (Welch and Widita, 2019), traffic monitoring (Gravvanis et al., 2019), education (Cantabella et al., 2019), humanitarian aid (Gazi and Gazis, 2021), business (Khanboubi et al., 2019), and finance (Begenau et al., 2019). A particularly interesting area is economics as both macro (see Teräsvirta et al., 2005; Önder et al., 2013; Kock and Teräsvirta, 2016) and microeconomics (see Terna, 1992; Erbas and Stefanou, 2009; Erdogdu, 2016) are centred on handling, analysing, pre-processing, and making predictions based on real datasets, like Chen et al. (2006) for Asia, Mostafa (2010) for Kuwait, Alhazbi et al. (2020) and Ahmed (2015) for Qatar, Qiu et al. (2016) for Japan, Long et al. (2020) for China, Manojlovic and Stajduhar (2015) for Croatia, Selim (2009), Kara et al. (2011) for Turkey, Ramezanian et al. (2019) and Parsva (2020) for Iran, Thakur et al. (2016) for India, Ismail et al. (2020) for Kuala Lumpur, Temponeras et al. (2019) and Kokkinos and Potamianos (2017) for Greece, and Oliveira et al. (2013) for Brazil. There are new statistical tools available each day and due to advancements in computer hardware new capabilities to test and simulate. As a result, traditional traders, have to consult forecasting models as their profit mainly depends on long-term results rather than day-to-day profits. Up until recently, the only forecasting models were based on linear regression or were used to support vector machine models [e.g., Kewat et al. (2017), or risk management (Barta and Görcsi, 2021)]. Over the past few years, there has finally been a surge of artificial neural network (ANN), giving people a powerful tool to be able to model such nonlinearities that are present in modelling the stock market. In the famous book, *A Non-Random Walk Down Wall Street*, written by Lo and Mackinlay (1999), through detailed analysis, significant flaws in both technical and fundamental analysis are noted, concluding that, for most investors, following these methods will produce inferior results compared to passive strategies. However, this book was written in the late '90s, long before the existence of actual working ANNs. Today, neural networks again offer popular solutions to many technological problems, therefore, we should explore all the countless possibilities they can offer in other fields.

Other research efforts rely on previous stock price data to predict future prices through extrapolation or linear regression. The accuracy of these kinds of models is heavily bottlenecked, not only by the inherent nonlinearity of the problem but also by the human factor of determining the margin of effect each indicator has on the final prediction (Hudson et al., 2017; Shen and Shafiq, 2020; Hu et al., 2021).

In this paper, we first apply feed-forward conventional neural network (FF-ANN) models, varying in size, to the historical prices of each company. We then propose an algorithm for probabilistic recovery of the classified data to improve the output model. A recurrent neural network (RNN) is also used, which is a type of ANN widely regarded as the best candidate to model time-series data, such as stock price history. Lastly, a convolutional neural network (CNN) is implemented to explore the visual interpretation of our dataset. We aim to use the best of these networks in terms of accuracy, to benchmark our model in real test data, and determine its actual predicting value. This paper is organised, as follows: in the next sections, we present the basics of neural network-based prediction using deep architectures, such as the novel post-processing reclassification module, the long short-time memory (LSTM) architectures and the CNN architectures. The proposed architectures are then evaluated over a real dataset derived from the US Dow Jones, producing interesting results.

Lastly, it is noted that the three architectures presented were selected based on the three fundamental topologies of neural networks. More analytically, derived from their characteristics, neural networks can be categorised centred on their application as follows:

1 *Fully connected neural networks:* Work best with numerical non-location-specific data.

2 *CNNs:* Work best with location-specific data.

3 *Recurrent/LSTM neural networks:* Work best with serial, time-related data.

4 *Generative adversarial neural networks:* Work best at simulating realistic data and can be used to test existing financial data (i.e., real-time stock market prices).

2 Related work

All stakeholders, regardless of their beliefs on fundamental or technical analysis, base their speculation on mathematical models in order to measure a company's impact, lower potential risk exposure, and maximise their profit. Although conventional approaches have produced interesting results throughout the years, new methods are starting to appear using mainly artificial intelligence and forecast-specific neural networks (Sermpinis et al., 2019).

ANNs are highly complex computational models functioning similarly to a human system of neurons. Many types of neural networks exist in stock market dynamics regarding regression analysis (e.g., Nunes et al., 2019), time series prediction (e.g., Stoean et al., 2019; Henrique et al., 2019), classification (e.g., Zhong and Enke, 2017), pattern recognition (e.g., Kaeppel, 2008), market trends (e.g., Atsalakis and Valavanis, 2009; Sezer and Ozbayoglu, 2018; Li et al., 2019) and decision making (e.g., Kraus and Feuerriegel, 2017; Merello et al., 2019).

The basic theory regarding stock price forecasting is the efficient market hypothesis, which asserts that everyone has some degree of access to the relevant information, and the price of a stock reflects all available information. The market supposedly reacts instantaneously to news and no one can outperform this in the long run. Several time series analysis models have been developed in an effort to forecast the market direction, such as by Pavlidis et al. (2003), Wei et al. (2005), Worasucheep (2016), Mingyue and Yu (2016), Vargas et al. (2017), Selvin et al. (2017), Hongping et al. (2018), Lee et al. (2019) and Bustos and Pomares-Quimbaya (2020). The first models, such as those presented by Moghaddam et al. (2016) and Zhong and Enke (2019), were derivatives of linear regression, offering sub-optimal accuracy, as the stock movement is highly irregular and completely nonlinear were presented. Almost all of these studies suggest that additional factors should be taken into account on top of the basic or unmodified model as suggested in Jagwani et al. (2018). The most common additional information considered, is the online news information related to the stock. Furthermore, recent research efforts have been focused on data mining and estimating the correlation between online information and price movement, using textual information in public media to evaluate news. This information was used in different studies, including but not limited to Maltoudoglou et al. (2015), Mingyue et al. (2016), Chen et al. (2017), Vargas et al. (2018), Symeonidis et al. (2018), Zhang et al. (2018), Khan et al. (2020) and Althelaya et al. (2021).

Lastly, it is noted that the technical novelty of this publication lies in the way the inputs are constructed as well as the detailed presentation of the data cleaning and processing for the proposed neural networks. This article manages to provide an annual return on investment of 60%, which is significantly higher than the return resulted from existing strategies, such as Huang et al. (2019), Song (2019) and Zhou (2019). Most researchers in this field focus their interest on providing complex multi-layer network comparisons over existing statistical analysis. In contrast, this publication showcases some of the most commonly used neural network approaches regarding daily stock exchange rates, i.e., for short-term historical stock prices, and suggests a novel neural network approach derived from our statistical and empirical analysis.

3 Deep neural network architectures for prediction

3.1 Feed-forward ANNs

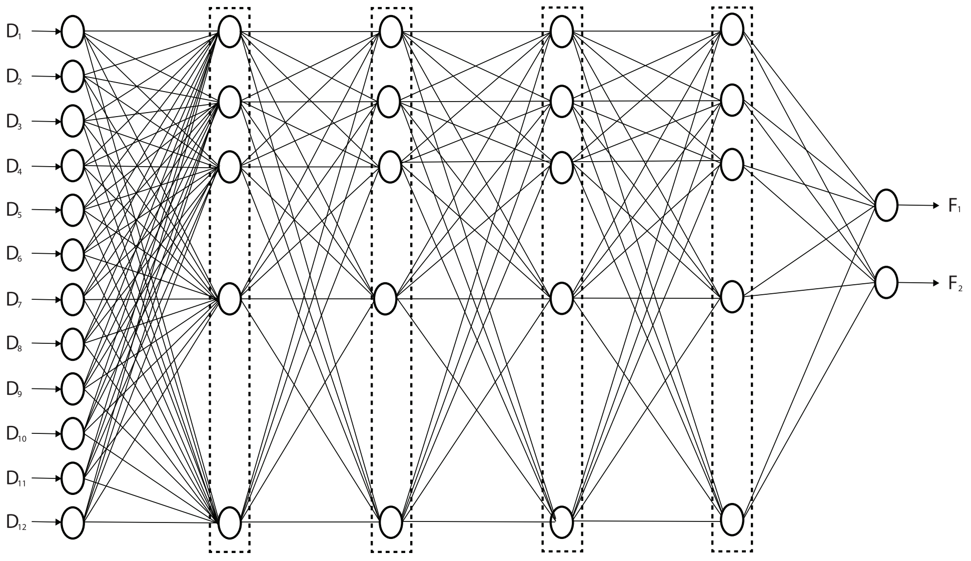
Feed-forward artificial neural networks (FF-ANN) are well-studied prediction tools as noted by Rosenblatt (1962) and Kumar and Yadav (2011). They are based on the function of the neurons of the human brain, where the ANN consists of a large number of artificial neurons linked together, and working together to solve a particular problem. Neurons are usually arranged on multiple levels and an ANN with multiple levels is called a 'deep neural network' (DNN). The topology of such a network consists of an input level, an unknown number of hidden levels, and a final decision level. The output of each layer in the neural network depends on the activation function and the connections between the levels and the neurons therein, given by the weight matrix W.

For a two-hidden-level neural network, an output plane and a vector $x \epsilon R$, the output is defined by ϵR, as follows:

$$h(1) = f(1)(w^{(1)}T + b^1)$$
$$h(2) = f(2)(w^{(2)}T + b^2)$$
$$h(3) = f(3)(w^{(2)}T + b^3)$$

where $h(i)$ with i = 1, 2, ... is the output of level i, $f(i)$ with i = 1, 2, 3 is the activation function of level j, b^i is the bias of level i, and w^i is the weight matrix of the plane i. One can observe that the neural network is just a synthesis of functions.

Figure 1 A common feed-forward neural network architecture



It is universally accepted by the universal approximation theorem, presented in Li and Wang (2017), that a FF neural network (as shown in Figure 1) with at least one hidden layer, a linear activation function for the hidden layer, and an activation function (such as tanh or sigmoid) can approach any function with only the requirement of having several neurons. Therefore, neural networks are a very powerful tool in describing complex nonlinear functions.

A common problem in neural network training is over-fitting, where a neural network becomes very familiar with the training dataset and fails to generalise and perform well with testing data, i.e., data not presented to the network before. One recent development to address the problem of over-fitting in DNNs is the dropout technique suggested by Srivastava et al. (2014) and Labach et al. (2019). According to this technique, a percentage p of the total number of neurons in the network is randomly switched off during training. This procedure randomly shrinks the network and serves to force the network to seek a simpler, more general solution, i.e., to avoid over-fitting. Using dropout, the network is given the opportunity to 'forget' the road that leads to the local minimum and to seek other opinions, that are likely to lead to the right destination, i.e., the total minimum. The aforementioned 'opinions' are the rest of the 'shallow' networks, each using a different subset of neurons. This technique is reported to improve the performance of DNNs.

3.2 An improvement to traditional FF-ANN architectures

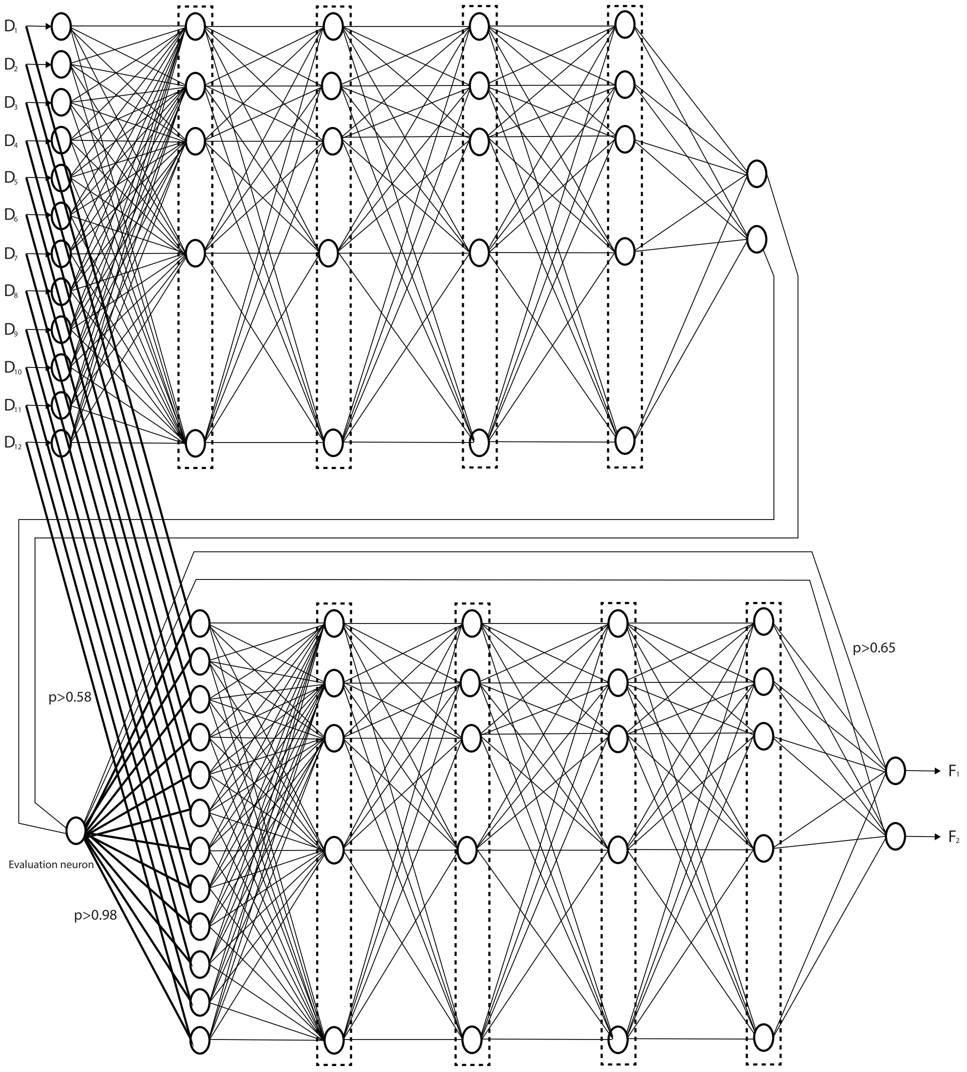
Similarly to Li et al. (2020), Sazli (2006) proposes an improvement of the original feed-forward topology through the addition of an extra layer that evaluates the output of the final softmax classification neuron. This layer selects the classification results from those that are reliable, and those that will be deleted and re-classified. At this point, we clarify that immediately before the execution of the neural network's evaluation, we have added an extra step where the results of the first neural network are evaluated based on the evaluation function presented in this publication. This step aims to develop a 'ground truth' validation mechanism since, in case of error, the second neural network acts as a means to evaluate, enhance, and calculate the output results based on the correct (i.e., real) dataset studied.

In a traditional FF-ANN, neural network weights are trained using a finite number of samples; when an acceptable error has been reached, the ANN stops updating the weights, and the training phase ends. The network cost function used to be the mean square error between the input and the output, however, it has been replaced by cross-entropy for classification networks with better results, as noted by Liu et al. (2017). The optimisation of the network cost function is usually performed using gradient descent, with an adaptive learning rate. In the testing phase, the most educated ANN is used to classify new samples that it has never encountered before.

Bishop (1995) claimed that the majority of the misclassification errors occured in those regions of x-space where the largest of the posterior probabilities are relatively low since there is then a strong overlap between different classes. In some applications, it may be better not to make a classification decision in such cases. This technique is called the 'reject option', where the rejected samples are later classified by a human expert. However, in our case, an augmented neural network, trained and modified based on the success of the original one, will take on the role of the human expert, relieving the pressure and obviating the time constraint, especially on big data applications, such as the stock market.

The new ANN proposed topology (Figure 2), which will be coined 'probabilistic FF-ANN', consists of two simple feed-forward networks that are connected in series with input data sharing, similarly to that described by Maniatopoulos et al. (2020b). The second network has additional training input samples that achieve a high probability of correct classification by the first FF-ANN. The second FF-ANN reclassifies only the most likely wrong results, leaving the correct classification unchanged.

Figure 2 The proposed probability-based cascade training neural network



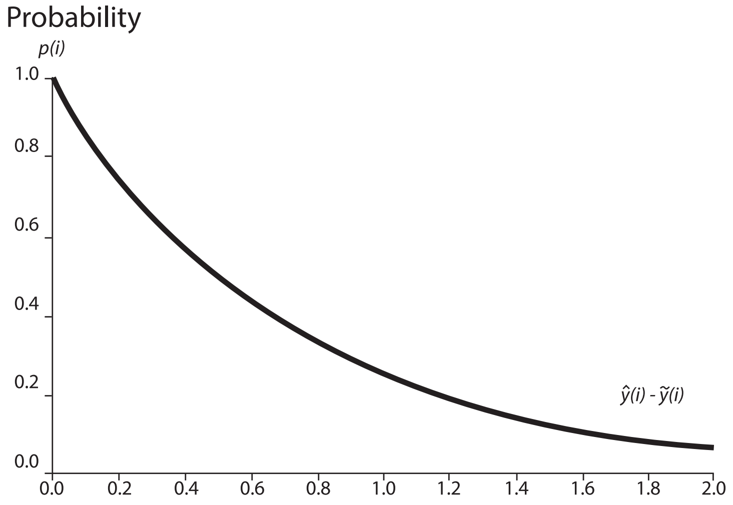
This idea is now illustrated in more detail. In the beginning, a simple deep FF-ANN is used to roughly sort the aggregate samples. This simple ANN is trained and evaluated as an independent neural network using the training dataset that is available. The result of this ANN is $\widehat{y}$. This is the first phase of training. The second phase of training includes the training of a second neural network of a similar size to the first one. A typical method would train the network using the initial training dataset. Using a combination of the two networks, we can now use more data to train the second ANN. The data are derived from the first ANN which has already been trained and has roughly classified the data from the test dataset. Therefore, we can use the results from the training dataset as well as the results from the test set that have been correctly classified, as training samples for the second ANN.

To identify the samples that have been correctly identified by the first ANN, we need a formula to calculate the probability of correct classification. In the case of positive classification, these samples will be used as training data for the second ANN. In the opposite case, they will be reclassified again by the second ANN, as soon as it is trained. The classification evaluation criterion is proposed, as follows:

$$p(i) = e^{-|\hat{y}(i)-\tilde{y}(i)|^2 ln2}$$

where represents the index of each sample and $\widetilde{y}$ represents the output of each sample of ANN, when rounded to the nearest whole number. Since the final classification is based on rounding to the nearest integer, the probability function must be equal to 0.5 in the middle of the two classes and must be nearly 0 for a distance of one class. The graph of each possible output $p(i)$, for a given $\hat{y}(i) - \tilde{y}(i)$ distance, is shown in Figure 3. With the aforementioned formula, the probability of correctness can be formed with acceptable accuracy. Note that, only probabilities of up to 0.5 may occur, but the formula calculates values to almost zero.

Figure 3 Probability of correct classification for the distance $\hat{y}(i) - \tilde{y}(i)$ between the estimated and the correct label



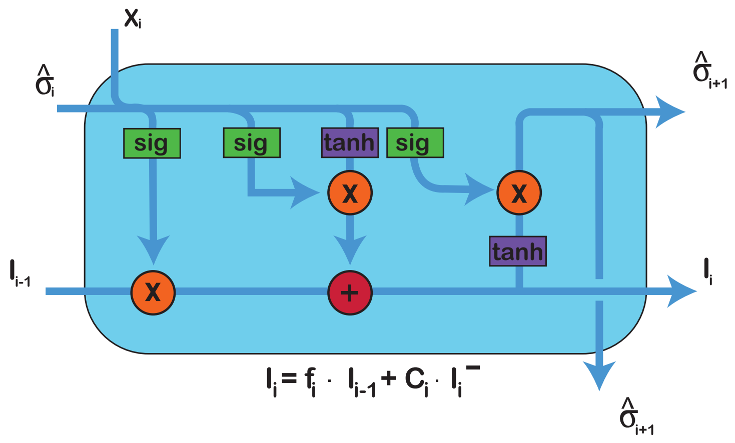
Each sample is then assigned a probability and the next FF-ANN is trained with all training data, plus any samples that achieve a probability value greater than 0.98. In this set of second-generation NN training, the training dataset is enriched with more data from the test dataset, which will improve its training. The learning function of both networks is based on the gradient descent with adaptive learning rate. For the second ANN, we use random initialisation of weights, to avoid falling into the same optima with the first ANN, i.e., producing the same classification.

Once the second ANN has been trained, it is ready to classify all the given evaluations. To save time in the calculation, we require the second network to reclassify only the samples with a probability of less than 0.58 as they are classified by the first ANN during the training phase. The remaining samples retain their own initial classification from the first coarse network. Using the proposed methodology, the ANN reduces the chance of being trapped to the local minimum so there is no real risk of destroying an initially good classification. We chose the value of 0.58 because it is the validation accuracy given by the first network; therefore, we consider any sample that does not satisfy the this probability value to be potentially wrong. Lastly, it is noted that the above-mentioned networks are characterised as a cascade (back-to-back) and act as a means to measure and validate the final results. More specifically, they are not used as a means to compare and select the optimal network, but, in case different results occur, our system's architecture selects the second one to take control and act as a conjugative solution.

3.3 RNNs with LSTM modules

The second architecture we studied is an ANN with a memory and feedback element, i.e., a RNN with a LSTM, as presented by Zaremba et al. (2014) and Yu et al. (2019). The basic RNN consists of a network of nodes, organised into successive layers as presented in Lipton et al. (2015). Each node in a layer is connected with a directed connection to every other node in the next successive layer. Each node has a time-varying real-valued activation and each connection features real-valued weights. Nodes can be either input nodes, hidden nodes, or output nodes. An LSTM network consists of a cell, an input gate, an output gate, and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. The architecture of the neural network is shown in Figure 4.

Figure 4 Diagram of a long short-term memory neuron (see online version for colours)



At each, $step_i$, the neuron takes the input vector x_i, the volatility estimation $\widehat{\sigma}_i$, and the information I_{i-1} that comes from the previous step. The gates are controlled by either the sigmoid function or the hyperbolic tangent (tanh) function, whilst the multiplication and addition are indicated by * and + within the model. The LSTM neuron update equation is as follows:

$$I_i = f_i I_{i-1} + c_i I_i^-$$

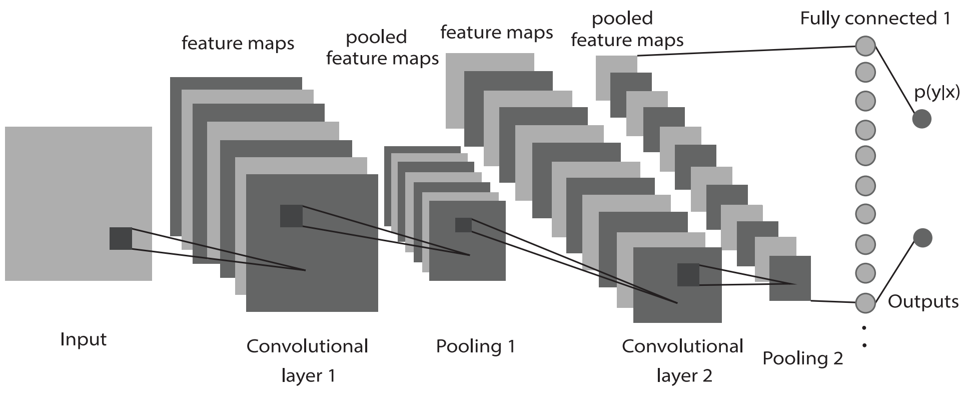
Here, f_i is the fraction of past information passed so far, counting the information flowing at this time, and c_i is the weight of how important this information stream is. These three quantities are the inputs of the function x_i and the variability estimator of the previous step.

$$f_i = sigmoid\left[(\widehat{\sigma}_i, w_f + b_f)\right]$$
$$c_i = sigmoid\left[(\widehat{\sigma}_i, w_c + b_c)\right]$$
$$I_i^- = tanh\left[(\widehat{\sigma}_i, w_I + b_I)\right]$$

3.4 Convolutional neural networks

The last architecture we studied was a CNN similar to the works of Chen and He (2018) and Liu et al. (2020). A typical CNN contains an input, and output layer and between them multiple hidden layers (as shown in Figure 5). The hidden layers include convolutional layers, pooling layers, fully connected layers as well as normalisation layers. The convolutional layers are what describe the nature of a CNN and are typically found in the largest number among other layers. These layers consist of some learnable 2D filters which despite being small in size, they manage to cover the entire input by moving spatially. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input, and producing an activation map of that filter. As a result, the network recognises and learns filters that activate when a specific type of feature is detected at a spatial position in the input.

Figure 5 A typical CNN architecture



However, to compress the size of the network while retaining the highest possible amount of information, CNN includes another type of layer, a form of nonlinear down sampling, called 'pooling'. There are several nonlinear functions to implement pooling among which max-pooling is the most common. Max-pooling function partitions the input image into a set of non-overlapping rectangles and, for each such sub-region it outputs the maximum. The intuition is that the rough location of a feature relative to other features is more important than its exact location. The pooling layer manages to progressively reduce the spatial size of the representation, decreasing the number of parameters and amount of computation in the network, and hence to also control over-fitting.

3.5 Generative adversarial network

In this section, we will present an adversarial network and, more specifically, the machine learning framework designed by Goodfellow et al. (2014), the 'generative adversarial network' (GAN). More specifically, GAN is a class of machine learning system that operates on the logic of rival learning and it is considered by Kishore et al. (2020) as "the smartest idea in machine learning in the last 20 years." A typical GAN topology is presented in Figure 6. The rationale behind GAN is that two neural networks compete in a (usually zero-sum) game; given a set of training samples and iterations, this technique teaches them to adapt and create new data with the same statistics. Although originally proposed as a purely productive model form for unsupervised learning applications, it can also be used accurately in semi-supervised learning, fully-supervised learning, and supportive learning (Luc et al., 2016; Creswell et al., 2018). An example of a GAN used in a real-life scenarios is the analysis of an image collection archive to create new images, which look at least superficially authentic to human observers but have plenty of realistic features (Wang et al., 2016).

A generator creates candidates whilst a 'discriminating network' (DN) evaluates them. The competition operates in terms of data distribution. Analytically, as shown in Figures 6 and 7 the generator learns to project from a latent space into the desired data distribution, whilst the DN distinguishes the generated candidates from the actual distribution. The training goal of a generator is to increase the error rate of a DN. To achieve this, the generator tries to 'trick' the DN by producing new (data) candidates that the DN will use along with the data from the actual distribution. More specifically, the generator's input is a random incoming sampling from a predetermined latent space (e.g., multiple normal distributions). It is then evaluated by a discriminator and, by using reverse propagation to both networks, the generator produces new (enhanced) data, whilst, the discriminator becomes more specialised in detecting synthetic data. For more on this subject see Karpathy et al. (2016).

Figure 6 A typical GAN architecture (see online version for colours)

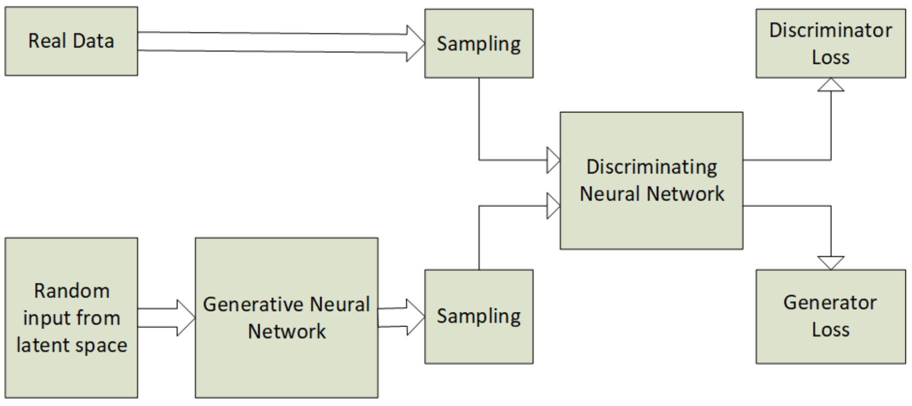
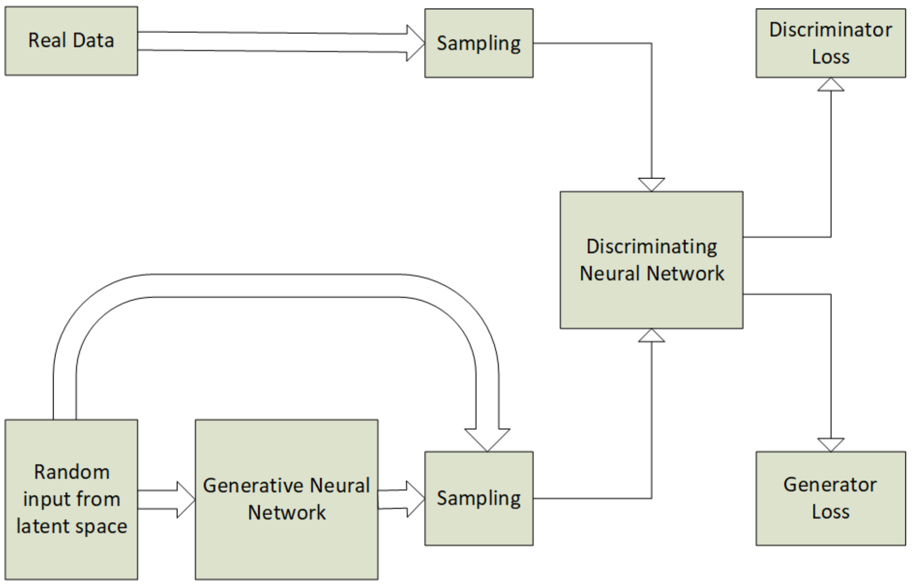

Real Data
Sampling
Discriminator Loss
Discriminating Neural Network
Random input from latent space
Generative Neural Network
Sampling
Generator Loss

Figure 7 A GAN topology for financial test data (see online version for colours)


Real Data
Sampling
Discriminator Loss
Discriminating Neural Network
Random input from latent space
Generative Neural Network
Sampling
Generator Loss

4 Data

In this section, we describe the dataset used in our experiments. The dataset presented (Maniatopoulos et al., 2020a), consists of daily data from the US Dow Jones for 501 large companies over the period 2010–2016; monthly publicly available indexes are also used. The first four years are used as training, whilst approximately the last two years are used for evaluating the efficiency of the model, benchmarking the annual return on investment at about 60%. A sliding window of 15 days has been used, giving us enough time-relevant information on price movement to use the dataset on regular classification networks. A duration of 15 days was determined by thorough experimentation. The raw price and volume information provided by the data could not be used efficiently due to large variances of price among companies. Our model should be able to work on all 501 companies regardless of their size/capital, therefore, a degree of regularisation is necessary.

Two datasets were built: one consisting of 12 assets and another of 16 assets. Both were used and evaluated for the FF-ANN models, whilst the recurrent and convolutional models were evaluated using the 12-asset dataset, which presented more stable results. The datasets consist of the following indices that were extracted in a 15-day window: simple moving average (opening prices), exponential moving average (opening prices), Garman-Klass (daily high, low, opening price, and closing price), momentum (closing prices), return vs. risk (closing prices), volume, exponential volume (stock volume), Williams R (high, low, and closing prices), baseline overall index (Policy Uncertainty Index), news-based index (Policy Uncertainty Index), real dividends (SP500), and real earnings (SP500).

The 12 above-mentioned indices, normalised to the simple moving average, constitute the dataset of this paper. The dataset is also divided into approximately

67% training data and 33% control/test data. Note that the dataset is not perfectly size-consistent, since new companies opened, whilst others went bankrupt. As a result, the dataset is only approximately split to 2/3 and 1/3. However, since the data are normalised and the names of the companies are not used, the accuracy and validity of our model are not affected by that. The results from the test data will be used to specify the efficiency of the methods used and analysed below. All indices were normalised to the simple moving average, so as not to let the scale difference deteriorate the training. Moreover, a higher value in these indices does not necessarily mean higher significance. The 16-indices dataset also contains the latter: consumer price index (SP500), long interest rate (SP500), and real price (SP500).

5 Model delineation

The algorithm accepts these indices as input and attempts to recognise whether or not the price of the share in question will be up or down at the end of the next 15 days. The outcome-output of the algorithm is the ratio of the predicted $X + 15$ value to the value at day X, having studied all the indicators for days $X - 15$. In brief, the algorithm on day X (i.e., 15/6) studies the indicators for days $X - 15$ (i.e., 1/6–15/6) and calculates the closing price of day $X + 15$ (i.e., 30/6) to day X value. If the result is above the unit, and depending on the projection price for the next 15 days, the right choice is to consider buying the share, as the expected price will be higher. In order to avoid limiting the algorithm to a single transaction per company, the magnitude of the change from the decision level (1.0), which corresponds linearly to the sale (or purchase) of more shares, the further the value differs from one. For example, when the projection is 1.1 (an increase of 10% in the value to 15 days), the algorithm will buy ten shares instead of one. At the end of each period, in order to calculate the final profit, every share is liquidated for each company to calculate the final balance. Next, we are going to outline the specific DNN architectures that were used in our experiments.

5.1 Standard FF-ANNs

In a classical case of a FF-ANN, a topology of 4 hidden layers and 50 neurons on each level was chosen. The entry of the neural network consists of 12–16 neurons, the same as the input vector, i.e., the number of derived indices. The output consists of a softmax layer, a kind of weighted average application, which ‘squashes’ the values at the final layer so that each entry is in the interval (0, 1), and all the entries add up to 1. The optimisation function used is the gradient descent, and the model ran for 100 epochs. We notice that with this topology, convergence stops at approximately 12–14 epochs, with the precision ‘clipping’ after that.

5.2 Probabilistic FF-ANNs

For the novel probabilistic FF-ANN, we used the same configuration as with the standard FF-ANN: two FF-ANNs with exactly the same configurations as the ones used in the previous FF-ANN. The first FF-ANN will conduct a pre-classification on the input data. The second will be trained again, also using the potentially correctly-classified data (with the predicted labels, not the ground truth) to augment the training set in order to classify any incorrectly classified data. More details are described in the earlier section.

5.3 RNNs with LSTM modules

The examined RNN consists of either 256 or 384 LSTM neurons in each of the two LSTM levels, followed by a fully-connected level (FF-ANN) with parallel dropout application with probability $p = 0.5$ at each level. The activation function for each neuron chosen was the sigmoid function, whilst the RMSProp algorithm, as presented in Graves (2013) and Mukkamala and Hein (2017), was selected for the optimisation of the model's loss function: binary cross-entropy. The LSTM element held 12 sequential vectors. Thorough experimentation has shown that the model ceased to improve after a few epochs (approximately 10), showing similar performance to all other models, except for the probabilistic recovery neural network.

Furthermore, we analysed the results from the neural network consisting of neurons with memory elements, and the final level of fully-connected traditional neurons. The main objective was to identify whether the use of sequential topology offered by the RNN, together with a forgetting element offered by the LSTM, could enhance the prediction capabilities of DNNs for financial forecasting. Here, we implemented a RNN with LSTM on the Keras frontend, using the TensorFlow backend and an NVidia GTX Titan Xp graphics processing unit card.

The first topology used 256 LSTM neurons in each of the two levels, and the second topology used 384 neurons per layer. In both topologies, a dropout layer was applied after each LSTM level with probability $p = 0.5$. From the 600,000 available training data, 10% of the data was used to construct the validation set.

5.4 Convolutional neural networks

The examined network has two convolutional levels, which are both followed by a max-pooling layer. More specifically, the first layer consists of 32 image filters and the second of 64 image filters. The network is completed by a fully-connected layer (FF-ANN) of 1,024 neurons, an output level with two neurons, as well as our logical outflow, i.e., the increase or decrease in the share price. The novelty here is the way the inputs are constructed. Each input consists of a 12-value vector, as used in the other architectures; however, here, they are temporally grouped to form a 12×12 'image'. In other words, we gather the 12-value vectors extracted for 12 consecutive days and arrange them one after the other to form a 12×12 array, i.e., an image. An example of such images can be seen in Figure 8, where the results shown in the real axis are presented in chronological order (consecutive days) and the imaginary axis is independent of the data points due to image skewing (data augmentation) of the studied dataset. This is an essential step to transform the data into the 2D requirement of CNNs. Another reason is that by using this 2D representation, we attempt to emphasise the temporal connection and correlation between the indices in adjacent days that may influence the trend of a share. Therefore, the convolutional filters of the CNN will attempt to identify this relationship and influence the final trend of a share.

The proposed CNN examines the data images, using 32 2×2 filters at the first convolutional level, whilst the second level uses 64 3×3 filters. Following the second level of max-pooling, there exists a fully-connected level of 1,024 neurons. This layer is necessary to evaluate the features that have been extracted from the convolutional layers. The output level consists of two neurons using softmax which represent the acceptable responses, i.e., up or down. In this topology, the ANN stops essential learning after a few (12–14) epochs. This has led to the application of the dropout technique to prevent over-fitting. It was initially tested with a probability of maintaining $p = 0.8$ of each neuron, and then with a probability of $p = 0.5$.

Figure 8 An input image for the convolutional ANN: 12-value vectors are extracted for 12 consecutive days and are arranged one after the other to form a 12 × 12 array

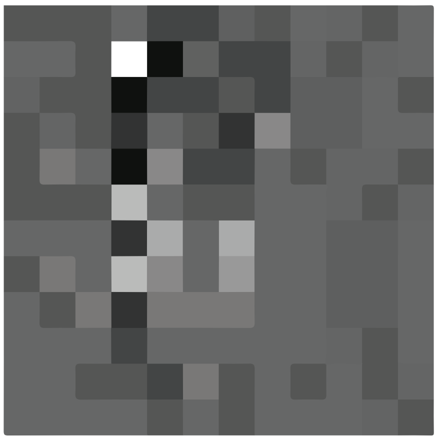
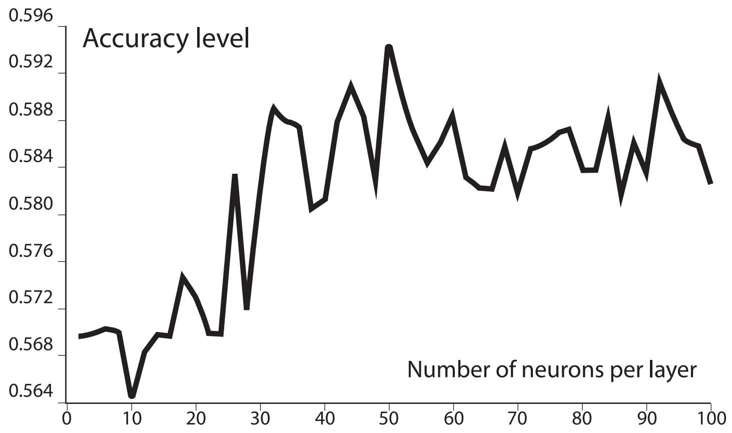


Figure 9 Determining the optimal number of neurons per layer



Notes: A plot of accuracy for various number of neurons used per layer indicates an optimal value of 50.

5.5 *The optimal number of neurons per layer*

Increasing the number of neurons increases the degrees of freedom for the neural network to adapt and classify more complex data. However, increasing the number of neurons increases the danger that the network will over-fit, resulting in degraded performance. The first experiment attempts to identify the optimal number of neurons per layer in the case of a simple feed-forward network. We created a simple FF-ANN with 4 layers but with a variable number of neurons. We examined the accuracy of prediction varying the number of neurons from 2 to 100. Figure 9 depicts the outcome of the experiment. Based on these measurements, it became clear that the optimal choice was 50 neurons per layer since it features the best performance. This marked the best trade-off between accuracy and computational complexity. The derived network belongs to the category of DNNs due to its size and complexity.

6 Data evaluation

In this chapter, we will study the performance of the proposed implementations. The following cases will be analysed and compared: deep vs. non-DNNs, FF-ANNs, feed-forward networks with probabilistic improvement of post-processing, RNNs with memory elements (LSTM), and CNNs. Finally, we will discuss the creation of the dataset and the selection of indices as well as the impact that each had on the precision of the final model.

For performance evaluation, the accuracy of the daily up and down price-movement prediction is evaluated. This metric is very strict since the daily small movements appear to be very random. The baseline accuracy is the ‘toss-the-coin’ probability of 50%. Accuracy consistently above 50% results in gradual money-making, beating the market. State-of-the-art models, such as Tkáč and Verner (2016), currently rarely exceed 54%. This level of accuracy is expected, because anything higher would result in an unstable and highly volatile stock market, rendering the whole idea of long-term commitment and believing/investing in the fate of a company obsolete.

Another method of evaluating predictions takes into account not the up-and-down movement accuracy of the model, but the final relative difference of the predicted and real prices. In this paper, the up/down movement evaluation method will mainly be used, however, the relative price differences evaluation (RPDE) method will also be presented for one company as a reference for other relative models. The company chosen for the RPDE method is ‘Apple Inc.’ (stock market name APPL) because, as it often appears in stock market forecasting.

To test the proposed architectures with another metric, we devised a trading algorithm for evaluation purposes. The algorithm uses the actual stock prices for the given days in order to purchase or sell the stock, as dictated by the neural network. The algorithm buys a company’s share if the network determines that the share price will go up in the next 15 days or sells it if it indicates the opposite. At the end of the test phase, the purchased shares are liquidated to have a final portfolio value that the network has achieved. Along with the final portfolio value, the algorithm keeps in mind the maximum amount invested and the maximum amount credited during the process. This way, we are not only monitoring the final portfolio value that each network topology has produced, but also the final rate of investment return. Below follows a series of experiments that report on the effectiveness of the proposed architectures.

6.1 The optimal number of days for prediction

In this section, we will examine the validity of using 15 days of past share values to estimate the 16 features that are used as input to our network. We used the probabilistic FF-ANN with 16 features as the testing framework since it demonstrated the best performance in our previous tests. We examined four different time spans to calculate the input features: 5, 10, 15 and 20 days. We did not examine for more than 20 days because the stationarity of the input data seemed to collapse for longer intervals. Similarly, we performed 50 independent runs for each time span. The results are outlined in Table 4. We concluded that the optimal time span is 15 days, as originally determined via extensive testing. We could also see that the network underperforms for longer periods since the stationarity of the dataset in these periods decreased.

6.2 FF-ANNs vs probabilistic FF-ANNs

This section will compare the performance of a FF-ANN with the novel proposed probabilistic F-ANN discussed earlier. To compare these two architectures, we will evaluate the accuracy of each topology based on the average of 50 random runs. The statistics of these runs can also demonstrate the consistency of the performance. We used the first 12 indices as input to the network. For the simple-FF network, the results for 50 runs are presented in Figure 10.

Figure 10 Accuracy per run for 50 independent executions for the simple network using 12 indices

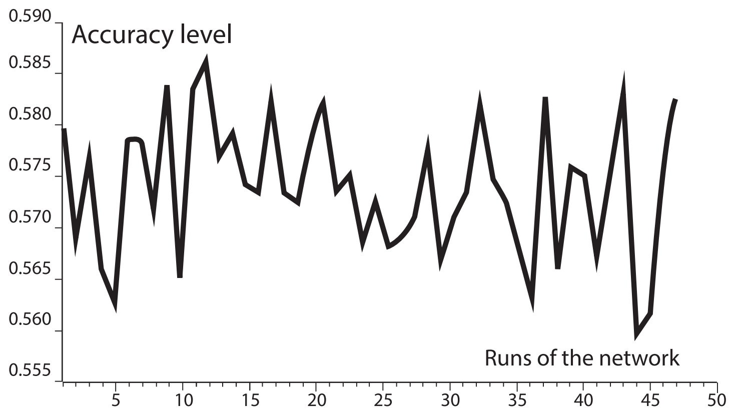


Table 1 presents the results for the simple FF-ANN. The mean accuracy achieved is 57.3%, which is above the toss-the-coin 50%. The standard deviation of accuracy for the 50 independent runs is 0.0081985, which implies that the outcome can be volatile. Using the trading-evaluation algorithm, we gather that the return on investment is 202.4%. Figure 11 depicts the accuracy of the proposed probabilistic FF-ANN for an average value of 58.83%. Table 2 presents the statistical profile for the network's accuracy and the return on investment. The probabilistic post-processing improves accuracy by approximately 0.9%, reaching the overall mean accuracy of 58.3%. The return on investment, however, has changed quite a lot, reaching 262%, offering a higher profit margin and making the new topology much more profitable.

Table 1 Feed-forward neural network tests for stock inputs (12 indices dataset) and 2 benchmark quantities (initial investment portfolio-wallet portfolio, current after transactions portfolio-trading portfolio)

Accuracy	*Wallet portfolio [$]*	*Trading portfolio [$]*
0.579646586	9,576,467.794	–1,376,974.578
0.568807229	10,609,572.73	–6,211,148.116
0.576879518	4,163,715.854	–3,891,246.343
0.565734940	17,233,786.58	–8,779,525.594
0.562618474	7,168,434.151	–1,342,184.974
0.578586345	15,301,159.24	–8,182,107.679
0.578469880	12,661,095.43	–5,055,161.415
0.571497992	9,779,461.713	–37,452,887.14
0.584096386	2,093,257.306	–7,919,887.396
0.564606426	6,345,811.712	–2,574,851.192
0.583710843	829,794.3146	–7,671,279.130
0.586208835	8,970,223.197	–1,864,784.108
0.576722892	9,873,186.899	–13,172,454.25
0.579168675	12,493,731.98	–14,252,039.92
0.574164659	13,715,931.32	–13,919,758.44
0.573385542	1,801,506.764	–6,445,478.138
0.582493976	8,204,882.495	–1,713,094.897
0.573261044	5,158,260.496	–4,340,296.276
0.572417671	11,365,918.72	–11,939,334.52
0.578473896	14,823,387.69	–3,731,525.024
0.582277108	5,059,541.740	–4,886,441.106
0.573461847	7,974,212.882	–3,446,596.427
0.575200803	11,813,165.92	–13,788,153.01
0.568489960	9,354,512.692	–4,426,274.906
0.572670683	7,698,806.315	–3,870,028.970
0.568096386	9,371,875.044	–796,707.5862
0.569088353	196,938.7691	–9,660,378.892
0.571060241	7,512,916.905	–10,745,922.81
0.577666667	12,783,843.13	–14,768,898.43
0.566638554	9713,623.553	–2,444,483.184
0.570963855	14,653,473.94	–14,884,279.76
0.573526104	16,188,439.44	–18,130,494.05
0.581971888	5,716,331.445	–5,342,362.033
0.574827309	7,659,757.631	–3,961,336.931
0.572485944	15,294,969.53	–14,519,065.83
0.567771084	3,416,317.113	–5,444,799.418
0.562955823	11,217,038.63	–4,736,644.478
0.583056225	11,381,068.89	–2,677,231.301
0.565638554	6,755,202.270	–1,611,231.182
0.575891566	8,871,174.699	–2,138,126.135
0.575084337	2,477,339.497	–5,256,613.192

Table 1 Feed-forward neural network tests for stock inputs (12 indices dataset) and 2 benchmark quantities (initial investment portfolio-wallet portfolio, current after transactions portfolio-trading portfolio) (continued)

Accuracy	*Wallet portfolio [$]*	*Trading portfolio [$]*
0.567020080	4,864,080.168	–2,124,374.007
0.575040161	9,439,300.018	–2,058,949.308
0.583048193	17,001,504.56	–40,477,649.09
0.559485944	10,703,287.00	–14,807,256.79
0.561489960	–3,864,657.252	–11,621,219.98
0.575100402	–2,952,916.044	–11,508,377.44
0.582606426	2,680,568.659	–5,580,786.148
0.562362346	12,472,992.69	–6,807,572.784
0.567345892	15,869,208.35	–7,583,452.349
Mean: 0.573824297	Mean: 8,440,652.157	Mean: –8,282,306.282
St. deviation: 0.006651228	St. deviation: 5,007,029.74	St. deviation: 8,012,412.169

Table 2 Proposed probabilistic recovery neural network tests for stock inputs (12 indices dataset) and 2 benchmark quantities (initial investment portfolio-wallet portfolio, current after transactions portfolio-trading portfolio)

Accuracy	*Wallet portfolio [$]*	*Investment portfolio [$]*
0.579646586	9,576,467.794	–1,376,974.578
0.568807229	10,609,572.73	–6,211,148.116
0.576879518	4,163,715.854	–3,891,246.343
0.565734940	17,233,786.58	–8,779,525.594
0.562618474	7,168,434.151	–1,342,184.974
0.578586345	15,301,159.24	–8,182,107.679
0.578469880	12,661,095.43	–5,055,161.415
0.571497992	9,779,461.713	–37,452,887.14
0.584096386	2,093,257.306	–7,919,887.396
0.564606426	6,345,811.712	–2,574,851.192
0.583710843	829,794.3146	–7,671,279.130
0.586208835	8,970,223.197	–1,864,784.108
0.576722892	9,873,186.899	–13,172,454.25
0.579168675	12,493,731.98	–14,252,039.92
0.574164659	13,715,931.32	–13,919,758.44
0.573385542	1,801,506.764	–6,445,478.138
0.582493976	8,204,882.495	–1,713,094.897
0.573261044	5,158,260.496	–4,340,296.276
0.572417671	11,365,918.72	–11,939,334.52
0.578473896	14,823,387.69	–3,731,525.024
0.582277108	5,059,541.740	–4,886,441.106
0.573461847	7,974,212.882	–3,446,596.427
0.575200803	11,813,165.92	–13,788,153.01
0.568489960	9,354,512.692	–4,426,274.906

Table 2 Proposed probabilistic recovery neural network tests for stock inputs (12 indices dataset) and 2 benchmark quantities (initial investment portfolio-wallet portfolio, current after transactions portfolio-trading portfolio) (continued)

Accuracy	*Wallet portfolio [$]*	*Investment portfolio [$]*
0.572670683	7,698,806.315	–3,870,028.970
0.568096386	9,371,875.044	–796,707.5862
0.569088353	196,938.7691	–9,660,378.892
0.571060241	7,512,916.905	–10,745,922.81
0.577666667	12,783,843.13	–14,768,898.43
0.566638554	9,713,623.553	–2,444,483.184
0.570963855	14,653,473.94	–14,884,279.76
0.573526104	16,188,439.44	–18,130,494.05
0.581971888	5,716,331.445	–5,342,362.033
0.574827309	7,659,757.631	–3,961,336.931
0.572485944	15,294,969.53	–14,519,065.83
0.567771084	3,416,317.113	–5,444,799.418
0.562955823	11,217,038.63	–4,736,644.478
0.583056225	11,381,068.89	–2,677,231.301
0.565638554	6,755,202.270	–1,611,231.182
0.575891566	8,871,174.699	–2,138,126.135
0.575084337	2,477,339.497	–5,256,613.192
0.567020080	4,864,080.168	–2,124,374.007
0.575040161	9,439,300.018	–2,058,949.308
0.583048193	17,001,504.56	–40,477,649.09
0.559485944	10,703,287.00	–14,807,256.79
0.561489960	–3,864,657.252	–11,621,219.98
0.575100402	–2,952,916.044	–11,508,377.44
0.582606426	2,680,568.659	–5,580,786.148
0.562362346	12,472,992.69	–6,807,572.784
0.567345892	15,869,208.35	–7,583,452.349
Mean: 0.582610191	Mean: 13,012,888.84	Mean: –10,351,956.39
St. deviation: 0.005227101	St. deviation: 10,782,717.71	St. deviation: 17,897,484.88

6.3 Probabilistic FF-ANN with 16 input features

In this paragraph, we examine the performance of the probabilistic FF-ANN with additional features as inputs. In total, we used an input vector of 16 features. The four additional features tested here are the SP500 index, consumer price index (SP500), long interest rate (SP500), and real price (SP500).

Once we assessed that the probabilistic FF-ANN performed better, we used this architecture to define the performance of the extra four features. Figure 12 depicts the accuracy for each of the 50 independent runs, whilst Table 3 describes the statistical properties on the accuracy and the return on investment. The proposed post-processing topology manages to improve the accuracy scoring over 59% and returning almost 257% of the original capital. The difference in accuracy with similar architecture using only

12 features is important and, justifies the use of a more complex dataset. Note, however, that the standard deviation is almost double that of the 12-indices dataset's, and as a result, only the 12-indices dataset will be used for the evaluation of the other two topologies. Finally, this analysis was executed on a daily basis and we accumulated the results of the monthly index (for 28, 29, 30 or 31 days accordingly). Analytically, on the last trading day of the month (i.e., after the end of the stock market's session), when the updated monthly index was publicly made available (value date), we immediately incorporated it into the dataset.

Figure 11 Accuracy per run for 50 independent executions for the probabilistic network using 12 indices

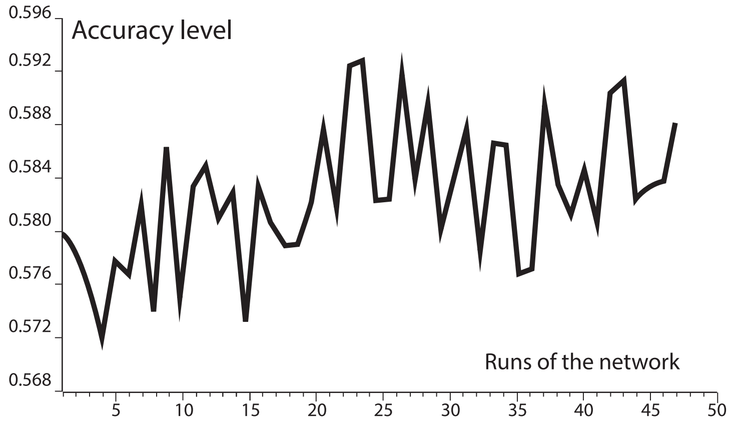
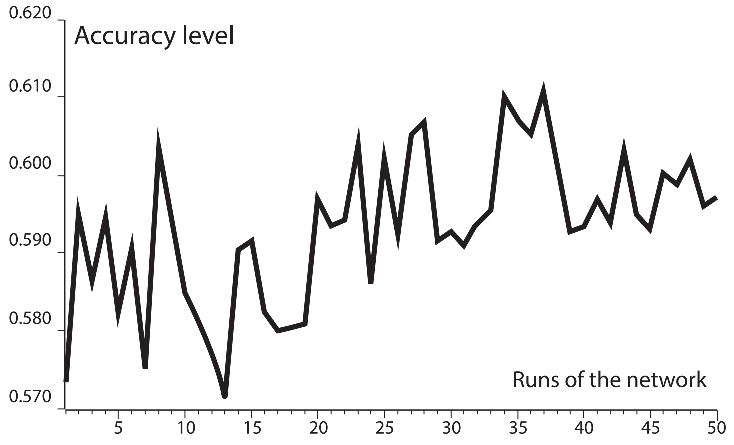


Figure 12 Accuracy per run for 50 independent executions for the probabilistic network using 16 indices



6.4 Convolutional neural networks

Here, we apply CNNs for the first time in the field of finance for the prediction of stock market index data. As mentioned earlier, the data are grouped by 12 days and entered as a 2D array into the neural network to be used for forecasting. This grouping attempts to enforce temporal coherence to the data and lets the CNN exploit the underlying temporal correlations that may exist. As previously explained, the CNN with the proposed architecture stops being substantially trained after 12–15 epochs. We examined the use of dropout with values of $p = 1$, $p = 0.8$, and $p = 0.5$ in an attempt to overcome the early network training termination. The CNN was implemented in Python using the TensorFlow package on the NVidia GTX Titan Xp graphical processing unit card.

Table 3 Proposed probabilistic recovery neural network tests for stock inputs (16 indices dataset) and 2 benchmark quantities (initial investment portfolio-wallet portfolio, current after transactions portfolio-trading portfolio)

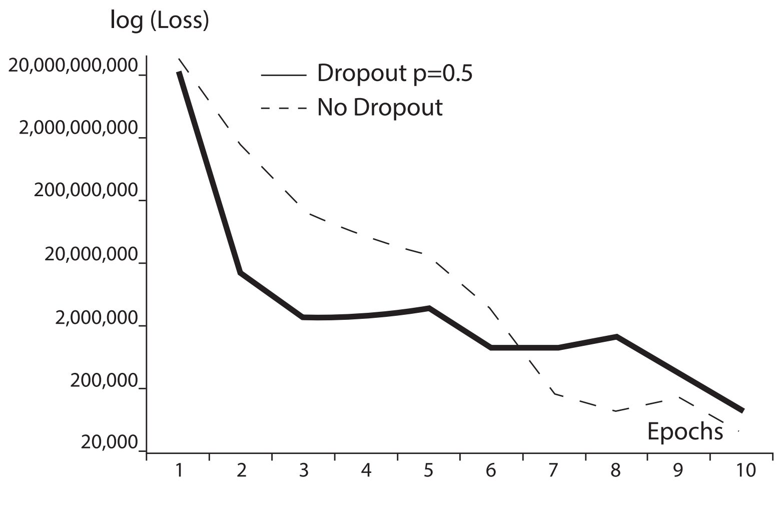
Accuracy	*Wallet portfolio [$]*	*Investment portfolio [$]*
0.573409639	49,692,800.69	–2,157,087.955
0.594943775	13,982,301.67	–15,873,073.71
0.586265060	20,228,797.13	–834,033.6136
0.594477912	6,975,487.039	–3,452,280.708
0.582321285	19,456,706.38	–3,247,303.090
0.590795181	4,972,975.284	–9,531,794.182
0.574710843	4,539,377.824	–3,949,730.872
0.603799197	17,844,692.84	–4,263,499.625
0.594481928	15,419,827.32	–3,491,683.282
0.584650602	10,357,347.95	–4,244,551.251
0.581457831	7,374,924.947	–4,229,400.284
0.577305221	232,725,396.9	–233,488,151.9
0.571317269	9,733,642.410	–3,225,659.117
0.590433735	7,375,111.185	–1,819,369.090
0.591666667	888,992.3039	–3,598,214.399
0.582281124	13,583,725.21	–16,971,746.62
0.579803213	11,271,968.64	–6,263,003.224
0.580493976	1,456,191.595	–2,465,736.967
0.580851406	33,129,766.46	–4,793,294.095
0.596979920	8,099,100.806	–3,754,576.693
0.593457831	6,391,852.159	–7,246,741.340
0.594389558	15,912,814.29	–2,908,281.007
0.604120482	27,712,782.10	–34,121,028.92
0.585506024	1,602,227.210	–6,369,763.190
0.602546185	362,966.1957	–10,832,299.96
0.592068273	11,710,914.01	–2,984,629.613
0.605345382	10,216,137.05	–2,231,771.714
0.606939759	2,253,865.528	–6,771,350.884
0.591485944	26,092,755.28	–10,103,033.80
0.592594378	44,227,714.27	–7,543,559.768
0.590835341	18,351,195.77	–3,281,857.291
0.593742972	21,224,295.94	–9,949,133.497
0.595534137	10,171,473.02	–9,485,553.920
0.610060241	18,416,221.13	–10,217,002.43

Table 3 Proposed probabilistic recovery neural network tests for stock inputs (16 indices dataset) and 2 benchmark quantities (initial investment portfolio-wallet portfolio, current after transactions portfolio-trading portfolio) (continued)

Accuracy	*Wallet portfolio [$]*	*Investment portfolio [$]*
0.607128514	11,664,624.33	–15,217,265.30
0.605204819	1,101,898.581	–6,603,268.221
0.610919679	4,959,502.535	–2,286,375.027
0.600614458	9,436,144.243	–5,601,165.252
0.592799197	22,186,618.37	–14,224,027.54
0.593401606	6,197,371.794	–4,160,122.545
0.596883534	1,238,564.779	–2,674,962.285
0.593847390	23,256,512.62	–13,395,360.86
0.603188755	16,278,427.29	–4,385,055.087
0.594771084	23,239,831.11	–7,335,415.832
0.593084337	13,459,190.48	–1,825,124.494
0.600377510	313,128.6255	–556,732.5636
0.598807229	11,804,650.95	–2,103,231.004
0.602184739	1,230,450.284	–3,237,346.278
0.596116466	6,434,961.056	–3,857,863.017
0.597257028	5,605,214.844	–6,354,547.362
Mean: 0.593153173	Mean: 17,243,268.81	Mean: –10,952,361.21
St. deviation: 0.009604004	St. deviation: 32,853,424.33	St. deviation: 32,618,635.98

It is obvious that the application of dropout does not show any noticeable improvement. It is interesting to note that despite the drop of error per iteration that is reduced by more than half, e.g., 34.205 instead of 88.878 (see Figure 13), the final precision is only slightly improved.

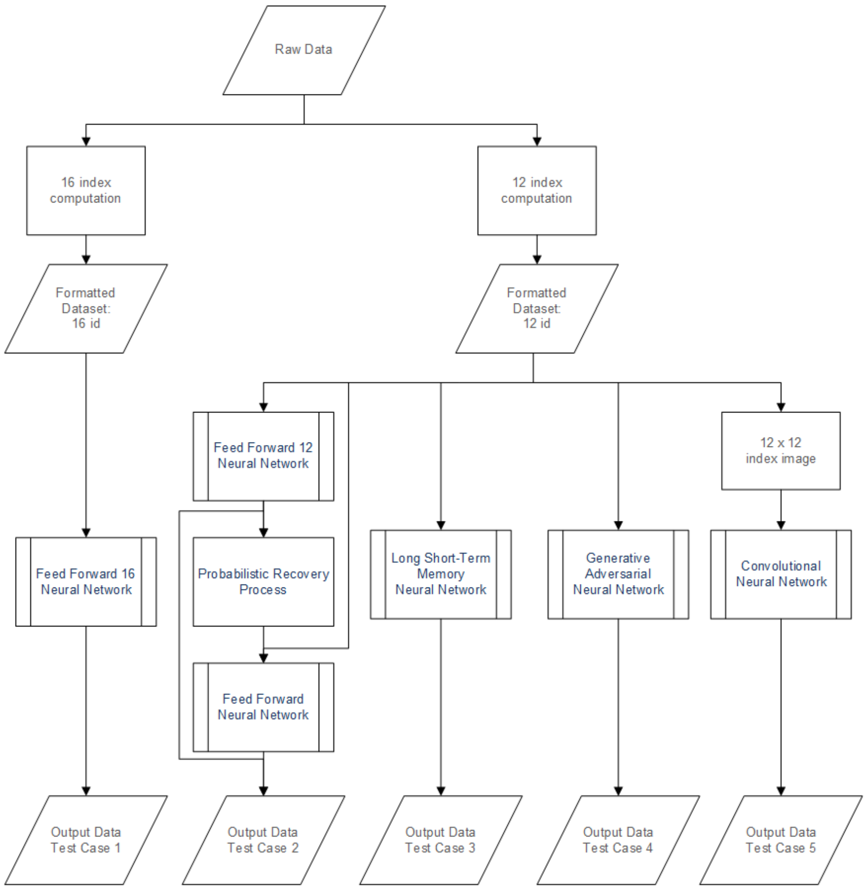
Figure 13 Comparison of loss between methods with and without dropout



6.5 Generative adversarial networks

The most widespread use of a GAN is for realistic image generation, but, this does not mean it can not be used to develop a behaviour stock market model. In our case, we aimed to manage a portfolio that would not only keep its current price but would also be profitable. To achieve this, the typical GAN architecture was changed slightly, as presented in Figure 6, to create a financial dynamic neural network architecture. Firstly, to control the generated data, the random input vector was replaced by the dataset input vector which is part of our latent input space (see Subsection 3.5). Specifically, the generator was trained to produce a decision (buy/sell), in the sampling stage and the input with the decision vector was concatenated. Later, along with the real training data, the data derived from the GAN were presented and evaluated by the discriminator.

Figure 14 Flow diagram of the studied neural network architectures (see online version for colours)



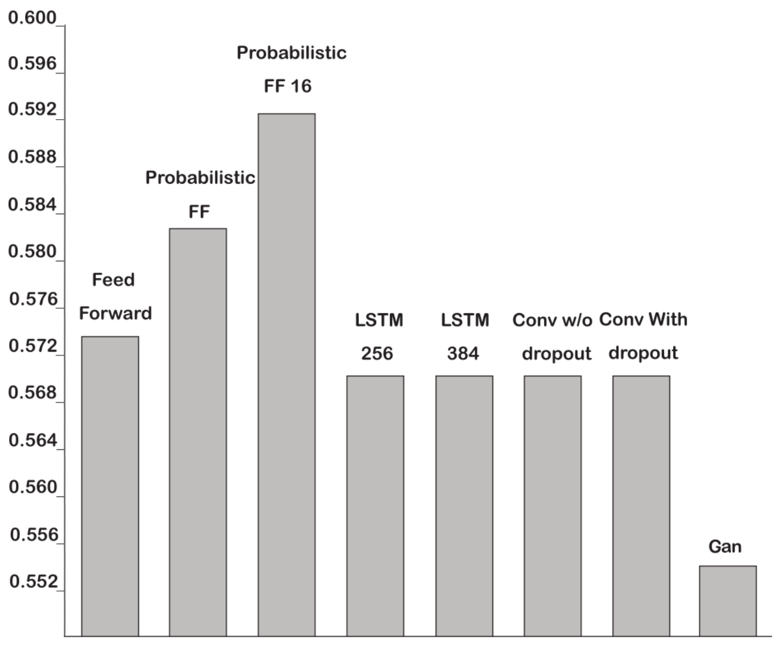
The topology corresponding to the GAN used in our financial data input case is presented in Figure 7. It is noted that a 1:4 ratio is used for the discriminator and generator training step as this was found to be a perfect balance. This ratio was selected after several experiments to determine what is the optimal training mode for the job and the discriminator's training. The generator was an optimal fully-connected model with 4 layers and 50 neurons per layer. These numbers were selected in accordance with previous test cases as the discriminator's part is similar to the CNN model used previously.

Finally, it is noted that further tests have been conducted replacing the CNN with a fully-connected ANN in the discriminator, without noticeable changes in the accuracy rate.

6.6 Overall comparison

In summary, a flow diagram of all the test cases conducted as well as the the accuracy of each of these topologies are presented in Figures 14 and 15 respectively. The probabilistic FF-ANN with 16 features seems to offer the best prediction accuracy, at over 59%.

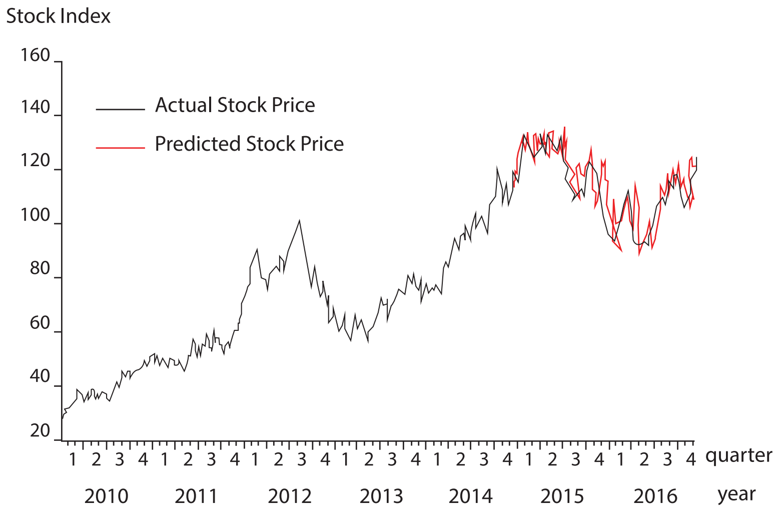
Figure 15 Prediction accuracy comparison among all methods tested



Similar systems, evaluated on the correct per-day prediction of up-or-down stock market movement, are not much better than the 'toss-the-coin' method, ranging from 52 to 54% at most. Schmidt and Hildebrandt (2017) use a deep learning framework and a similar but undisclosed dataset, reporting accuracy that does not exceed 52.8%, whilst assessing both FF models and RNNs. Chen et al. (2017) offer a system modelling how the Dow Jones Index would be affected by daily popular news on Reddit. Two algorithms were mostly used – CART and support vector machines – with CART offering the best performance in terms of mean accuracy at 53.66%.

Our models surpass the aforementioned systems, offering over two times better relative accuracy, assuming the baseline is 50%, all the while modelling a vast number of stocks (501) – a problem much more widespread and demanding than trying to model efficiently only one index or single stock. Apple Inc. is one of the most common targets for stock market prediction. Our model has achieved 93.43% accuracy, modelling 1,247 days and evaluating for 500 days, as shown in Figure 16. Comparable models achieve less than 80% accuracy, rendering them unviable. Moreover, it is noted that during the first 1,247 days the model is idle and trained, whilst in the last 500, it is evaluated and models the real price efficiently. Moreover, after this stock analysis regarding its predicted-actual values and the absolute-relative error margins, we calculated that the mean square error and the root mean squared error are 84.35877379 and 9.18470325 respectively.

Figure 16 Testing case with an Apple stock price (see online version for colours)



Notes: A comparison between the actual stock price and the predicted stock price using the best performing methodology.

7 Conclusions and perspectives

This paper aimed to build a capable dataset as well as a robust model to solve the problem of predicting market movement for high frequency transactions – a problem that classical mathematics cannot solve. Our results showcase the model's capability for high accuracy predictions in large volume and high frequency transactions datasets, using unsupervised learning techniques. Initially, different forms of the dataset were studied to find the most appropriate one because information is power, and without proper data, no topology can solve an ill-composed problem.

Then, when the appropriate dataset was selected, four topologies were studied to find the most appropriate tool to solve the problem more efficiently. The first architecture was a FF-ANN, which gave us an accuracy of approximately 57.36%. The second architecture consisted of a post-processing network that evaluated the results of the FF-ANN, accepted what it deemed to be correct, and reclassified what it depicted was probably wrong. As a result, the accuracy that occurred after the use of the probabilistic post-processing network exceeded 58.83%, an improvement that reached 1.5% on the original coarse network and a higher accuracy than the topologies tested. The third topology tested was the sequential ANN with LSTM elements. Two topologies were tested and, although theoretically better results were expected due to the nature of the problem as sequential neural circuits are ideal for time series processing, the final precision approaches 57.03% regardless of topology, number of neurons, or dropout implementation.

Figure 17 A comparison between the actual stock price and the predicted stock price using the best performing methodology for 50 executions (see online version for colours)

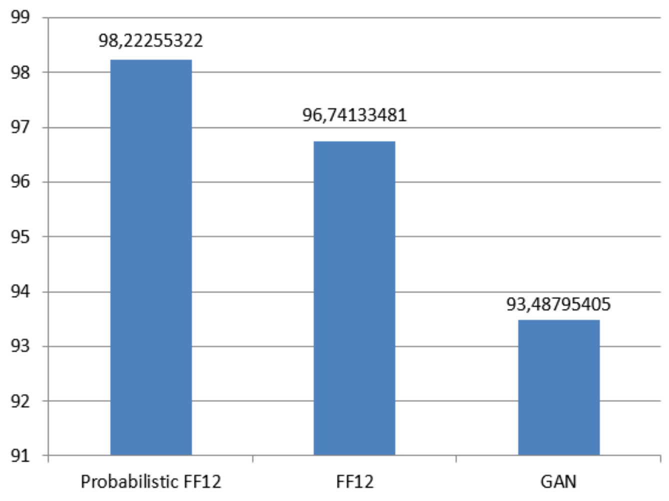
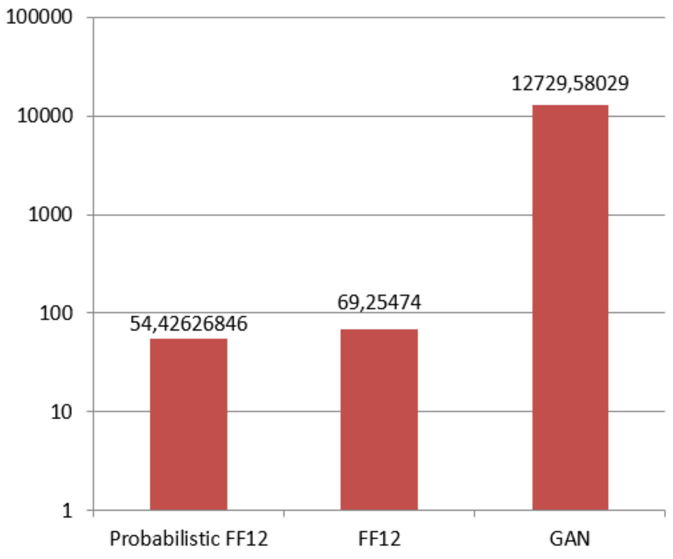


Figure 18 Normalised standard deviation (%) for feed-forward neural network, probabilistic neural network and GAN for 50 executions of the 12 indices dataset (see online version for colours)



For the first time in the field, a convolution neural network was tested. For the proper use of this topology, a sliding window was used with a data aggregation per 12, resulting in the neural network receiving the information in the form of images. Although theoretically not an ideal topology for time series modelling, the final precision was exactly the same as that of the sequential neural network at 57.03%.

Table 4 Accuracy (Acc) results for varying time window sizes

5-day ahead Acc	*10-day ahead Acc*	*15-day ahead Acc*	*20-day ahead Acc*
0.575533576	0.585791924	0.580452510	0.572760312
0.564756967	0.578173000	0.596400502	0.588244884
0.573821163	0.595627653	0.600406928	0.598823411
0.573194566	0.569792184	0.592027108	0.560497362
0.561091804	0.577475675	0.587122691	0.581045678
0.567284813	0.560309422	0.580672691	0.594203154
0.559392267	0.575112517	0.589924598	0.591332273
0.486745648	0.564334107	0.611502309	0.598310743
0.566804136	0.591034776	0.605073896	0.590799666
0.579057110	0.576584648	0.582028313	0.601506384
Average: 0.560768205046311	Average: 0.577423590534156	Average: 0.592561154618474	Average: 0.587752386610811

Table 5 Numerical comparison of feed-forward neural network, probabilistic recovery neural network and GAN for the 12 indices dataset

Neural network test cases	*Feed-forward neural network*	*Probabilistic neural network*	*Generative adversarial neural network*
Accuracy	0.573824297	0.582610191	0.554526766
Standard deviation	0.006651228	0.005227101	1.222549400

Using RPDE, our models achieved well over 93% accuracy in price prediction, outperforming existing models currently on the market, and enjoying over 60% annual return on investment. Additionally, we noticed that the final prediction considers approximately an average 5% reduction regarding possible transaction costs and stock exchange companies' commissions.

The technical novelty of this paper is that the system proposed, contrary to the theory of non-market forecasting (that suggests that one can not make a profit – i.e., accuracy over 52% – by modelling the market) achieves almost 60% accuracy in the upwards-downwards movement of shares. Additionally, our system is characterised by an accuracy rate of slightly over 90%, whereas the latest research in the field using similar methods spans from 70% to 85%.

Furthermore, although GAN is not widely used in the fields of market and stock market predictions, our novel GAN model achieved a 55.453% mean accuracy score and a 1.222 standard deviation over 50 runs. The results of this method are promising, proving further robustness of our constructed dataset and not using daily data of the US Dow Jones Index. However, the authors argue that most conventional models can achieve greater accuracy and lower standard deviation as presented comparatively in

Figures 17, 18 and Table 5. Finally, these results showcase that GANs could be used as a tool to enhance current artificial intelligence tools, both by validating the prediction and by enhancing the overall system dataset. We propose this method not to be used as a standalone feature but rather as a supplementary tool for validation of the 'ground truth' of machine and deep learning trading fine-tuning strategies.

Finally, regarding the future work of this publication, we aim to develop an open-source library MATLAB, Python, and R repository for any user to be able to use via an API, our proposed neural network. Investment-wise, future uses of this method may include using this neural network as a solid tool to comprehend market dynamics and shape possible trading strategies. Specifically, investors may use our system's results as a way to study price patterns and technical indicators for any given stock market. Moreover, correlating forecasting for specific time frames (i.e., given input sliding window size) can be used as a tool to study stock price times in future policy (regulation) events such as rebalancing prices after additions/deletions from specific indexes (such as MSCI's market indexes). Also, an alarming concern is the recent trend of contemporary portfolios to move away from mutual funds and stock picking to shifting towards their capital to index funds. In the near future, will a stock's price embed all the necessary information and trends regarding its current market value? It is important to focus our future system improvements on conducting a comparative study between index funds and individually the group of stocks consisting of them. For example, we are currently working on a method that actively manipulates the topology of the neural networks, fitting the size of each layer to the problem, and using information theory to simulate stock market index behaviour.

References

Ahmed, W.M. (2015) 'On the buying and selling behaviour of investor categories: evidence from Qatar', *International Journal of Economics and Business Research*, Vol. 9, No. 3a, pp.292–315 [online] https://dx.doi.org/10.1504/IJEBR.2015.068551.

Alhazbi, S., Said, A.B. and Al Maadid, A. (2020) 'Using deep learning to predict stock movements direction in emerging markets: the case of Qatar Stock Exchange', *IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp.440–444 [online] https://doi.org/10.1109/ICIoT48696.2020.9089616.

Althelaya, K.A., Mohammed, S.A. and El-Alfy, E.S.M. (2021) 'Combining deep learning and multiresolution analysis for stock market forecasting', *IEEE Access*, Vol. 9, pp.13099–13111 [online] https://doi.org/10.1109/ACCESS.2021.3051872.

Askari, M.Y. and Refae, G.A.E. (2019) 'The rationality of irrational decisions: a new perspective of behavioural economics', *International Journal of Economics and Business Research*, Vol. 17, No. 4, pp.388–401 [online] https://dx.doi.org/10.1504/IJEBR.2019.099972.

Atsalakis, G.S. and Valavanis, K.P. (2009) 'Forecasting stock market short-term trends using a neuro-fuzzy based methodology', *Expert Systems with Applications*, Vol. 36, No. 7, pp.10696–10707 [online] https://doi.org/10.1016/j.eswa.2009.02.043.

Barta, G. and Görcsi, G. (2021) 'Risk management considerations for artificial intelligence business applications', *International Journal of Economics and Business Research*, Vol. 21, No. 1, pp.87–106 [online] http://dx.doi.org/10.1504/IJEBR.2021.112012.

Begenau, J., Farboodi, M. and Veldkamp, L. (2019) 'Big data in finance and the growth of large firms', *Journal of Monetary Economics*, Vol. 97, pp.71–87 [online] https://doi.org/10.1016/j.jmoneco.2018.05.013.

Bishop, C.M. (1995) *Neural Networks for Pattern Recognition*, Oxford University Press [online] http://publications.aston.ac.uk/id/eprint/639/.

Borovykh, A., Bohte, S. and Oosterlee, C.W. (2018) 'Dilated convolutional neural networks for time series forecasting', *Journal of Computational Finance Forthcoming* [online] https://ssrn.com/abstract=3272962.

Bustos, O. and Pomares-Quimbaya, A. (2020) 'Stock market movement forecast: a systematic review', *Expert Systems with Applications*, Vol. 156, p.113464 [online] https://doi.org/10.1016/j.eswa.2020.113464.

Cantabella, M., España, P.M., Ayuso, B., Yáñez, J.A. and Muñoz, A. (2019) 'Analysis of student behavior in learning management systems through a big data framework', *Future Generation Computer Systems*, Vol. 90, pp.262–272 [online] https://doi.org/10.1016/j.future.2018.08.003.

Chaboud, A.P., Chiquoine, B., Hualmarsson, E. and Vega, C. (2014) 'Rise of the machines: algorithmic trading in the foreign exchange market', *Journal of Finance*, Vol. 69, pp.2045–2084 [online] https://doi.org/10.1111/jofi.12186.

Chen, S. and He, H. (2018) 'Stock prediction using convolutional neural network', *IOP Conference Series: Materials Science and Engineering*, Vol. 435, No. 1, p.12026 [online] https://doi.org/10.1088/1757-899X/435/1/012026.

Chen, W.H., Shih, J.Y. and Wu, S. (2006) 'Comparison of support-vector machines and back propagation neural networks in forecasting the six major Asian stock markets', *International Journal of Electronic Finance*, Vol. 1, No. 1, pp.49–67 [online] https://doi.org/10.1504/IJEF.2006.008837.

Chen, W., Zhang, Y., Yeo, C.K., Lau, C.T. and Lee, B.S. (2017) 'Stock market prediction using neural network through news on online social networks', *IEEE International Smart Cities Conference*, pp.1–6 [online] https://doi.org/10.1109/ISC2.2017.8090834.

Chong, E., Han, C. and Park, F.C. (2017) 'Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies', *Expert Systems with Applications*, Vol. 83, pp.187–205 [online] https://doi.org/10.1016/j.eswa.2017.04.030.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B. and Bharath, A.A. (2018) 'Generative adversarial networks: an overview', *IEEE Signal Processing Magazine*, Vol. 35, No. 1, pp.53–65 [online] https://doi.org/10.1109/MSP.2017.2765202.

Erbas, B.C. and Stefanou, S.E. (2009) 'An application of neural networks in microeconomics: input-output mapping in a power generation subsector of the US electricity industry', *Expert Systems with Applications*, Vol. 36, No. 2, pp.2317–2326 [online] https://doi.org/10.1016/j.eswa.2007.12.062.

Erdogdu, E. (2016) 'Asymmetric volatility in European day-ahead power markets: a comparative microeconomic analysis', *Energy Economics*, Vol. 56, pp.398–409 [online] https://doi.org/10.1016/j.eneco.2016.04.002.

Gazi, T. and Gazis, A. (2021) 'Humanitarian aid in the age of COVID-19: a review of big data crisis analytics and the General Data Protection Regulation', *International Review of the Red Cross*, Vol. 102, No. 913, pp.75–94 [online] https://doi.org/10.1017/S1816383121000084.

Gazis, A. and Katsiri, E. (2019) 'Web frameworks metrics and benchmarks for data handling and visualization', *Theoretical Computer Science and General Issues – Lecture Notes in Computer Science: International Symposium on Algorithmic Aspects of Cloud Computing*, Vol. 11409, pp.137–151 [online] https://doi.org/10.1007/978-3-030-19759-9_9.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014) 'Generative adversarial nets', *Advances in Neural Information Processing System*, Vol. 27, pp.2672–2680 [online] https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

Gopal, N. and Senthilkumar, K.S. (2020) 'Predicting bitcoin prices-ANN approach', *International Journal of Electronic Finance*, Vol. 10, Nos. 1–2, pp.67–78 [online] https://doi.org/10.1504/IJEF.2020.110296.

Graves, A. (2013) *Generating Sequences with Recurrent Neural Networks*, arXiv, 1308.0850 [online] https://arxiv.org/abs/1308.0850.

Gravvanis, G.A., Salamanis, A.I. and Filelis-Papadopoulos, C.K. (2019) 'Advanced parametric methods for short-term traffic forecasting in the era of big data', *Machine Learning Paradigms*, Vol. 149, pp.199–231 [online] https://doi.org/10.1007/978-3-319-94030-4_9.

Henrique, B.M., Sobreiro, V.A. and Kimura, H. (2018) 'Stock price prediction using support vector regression on daily and up to the minute prices', *The Journal of Finance and Data Science*, Vol. 4, No. 3, pp.183–201 [online] https://doi.org/10.1016/j.jfds.2018.04.003.

Hongping, H., Tang, L., Shuhua, Z. and Haiyan, W. (2018) 'Predicting the direction of stock markets using optimized neural networks with Google Trends', *Neurocomputing*, Vol. 285, pp.188–195 [online] https://doi.org/10.1016/j.neucom.2018.01.038.

Hu, Z., Zhao, Y. and Khushi, M. (2021) 'A survey of forex and stock price prediction using deep learning', *Applied System Innovation*, Vol. 4, No. 1, p.9 [online] https://doi.org/10.3390/asi4010009.

Huang, X. (2018) 'Macroeconomic news announcements, systemic risk, financial market volatility, and jumps', *Journal of Futures Markets*, Vol. 38, pp.513–534 [online] https://doi.org/10.1002/fut.21898.

Huang, Y., Capretz, L.F. and Ho, D. (2019) 'Neural network models for stock selection based on fundamental analysis', *IEEE Canadian Conference of Electrical and Computer Engineering*, pp.1–4 [online] https://doi.org/10.1109/CCECE.2019.8861550.

Hudson, R., McGroarty, F. and Urquhart, A. (2017) 'Sampling frequency and the performance of different types of technical trading rules', *Finance Research Letters*, Vol. 22, pp.136–139 [online] https://doi.org/10.1016/j.frl.2016.12.015.

Ismail, M.S., Noorani, M.S.M., Ismail, M., Razak, F.A. and Alias, M.A. (2020) 'Predicting next day direction of stock price movement using machine learning methods with persistent homology: evidence from Kuala Lumpur Stock Exchange', *Applied Soft Computing*, Vol. 93, p.106422 [online] https://doi.org/10.1016/j.asoc.2020.106422.

Iuhasz, G., Tirea, M. and Negru, V. (2012) 'Neural network predictions of stock price fluctuations', *IEEE International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp.505–512 [online] https://doi.org/10.1109/SYNASC.2012.7.

Jagwani, J., Gupta, M., Sachdeva, H. and Singhal, A. (2018) 'Stock price forecasting using data from Yahoo finance and analysing seasonal and nonseasonal trend', *IEEE International Conference on Intelligent Computing and Control Systems*, pp.462–467 [online] https://doi.org/10.1109/ICCONS.2018.8663035.

Jeon, S., Hong, B. and Chang, V. (2018) 'Pattern graph tracking-based stock price prediction using big data', *Future Generation Computer Systems*, Vol. 80, pp.171–187 [online] https://doi.org/10.1016/j.future.2017.02.010.

Kaeppel, J. (2008) *Seasonal Stock Market Trends: The Definitive Guide to Calendar-Based Stock Market Trading*, No. 438, John Wiley and Sons Trading, ISBN: 978-0-470-27043-1 [online] https://www.shorturl.at/tEJV1.

Kara, Y., Boyacioglu, M.A. and Baykan, O.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*, Vol. 38, No. 5, pp.5311–5319 [online] https://doi.org/10.1016/j.eswa.2010.10.027.

Karpathy, A., Abbeel, P., Brockman, G., Chen, P., Cheung, V., Duan, R., Goodfellow, I., Kingma, D., Ho, J., Houthooft, R., Salimans, T., Schulman, J., Sutskever, I. and Zaremba, W. (2016) *Generative Models* [online] https://openai.com/blog/generative-models/.

Kewat, P., Sharma, R., Singh, U. and Itare, R. (2017) 'Support vector machines through financial time series forecasting', *IEEE International conference of Electronics, Communication and Aerospace Technology*, Vol. 2, pp.471–477 [online] https://doi.org/10.1109/ICECA.2017.8212859.

Khan, W., Ghazanfar, M.A., Azam, M.A., Karami, A., Khaled, H.A. and Alfakeeh, A.S. (2020) 'Stock market prediction using machine learning classifiers and social media, news', *Journal of Ambient Intelligence and Humanized Computing*, pp.1–24 [online] https://doi.org/10.1007/s12652-020-01839-w.

Khanboubi, F., Boulmakoul, A. and Tabaa, M. (2019) 'Impact of digital trends using IoT on banking processes', *Procedia Computer Science*, Vol. 151, pp.77–84 [online] https://doi.org/10.1016/j.procs.2019.04.014.

Kirilenko, A.A. and Lo, A.W. (2013) 'Moore's law versus Murphy's law: algorithmic trading and its discontents', *Journal of Economic Perspectives*, Vol. 27, No. 2, pp.51–72 [online] https://dx.doi.org/10.2139/ssrn.2235963.

Kishore, A., Kumar, A. and Dang N. (2020) 'Enhanced image restoration by GANs using game theory', Vol. 173, pp.225–233 [online] https://doi.org/10.1016/j.procs.2020.06.027.

Kock, A.B. and Teräsvirta, T. (2016) 'Forecasting macroeconomic variables using neural network models and three automated model selection techniques', *Econometric Reviews*, Vol. 35, Nos. 8–10, pp.1753–1779 [online] https://doi.org/10.1080/07474938.2015.1035163.

Kokkinos, F. and Potamianos, A. (2017) *A Structural Attention Neural Networks for Improved Sentiment Analysis*, arXiv, 1701.01811 [online] https://arxiv.org/abs/1701.01811.

Kraus, M. and Feuerriegel, S. (2017) 'Decision support from financial disclosures with deep neural networks and transfer learning', *Decision Support Systems*, Vol. 104, pp.38–48 [online] https://doi.org/10.1016/j.dss.2017.10.001.

Kumar, M. and Yadav, N. (2011) 'Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey', *Computers and Mathematics with Applications*, Vol. 62, No. 10, pp.3796–3811 [online] https://doi.org/10.1016/j.camwa.2011.09.028.

Kusuma, R.M.I., Ho, T.T., Kao, W.C., Ou, Y.Y. and Hua, K.L. (2019) *Using Deep Learning Neural Networks and Candlestick Chart Representation to Predict Stock Market*, arXiv, 1308.0850 [online] https://arxiv.org/abs/1308.0850.

Labach, A., Salehinejad, H. and Valaee, S. (2019) *Survey of Dropout Methods for Deep Neural Networks*, arXiv, 1904.13310 [online] https://arxiv.org/abs/1904.13310.

Lee, T.K., Cho, J.H., Kwon, D.S. and Sohn, S.Y. (2019) 'Global stock market investment strategies based on financial network indicators using machine learning techniques', *Expert Systems with Applications*, Vol. 117, No. 1, pp.228–242 [online] https://doi.org/10.1016/j.eswa.2018.09.005.

Li, D., Wang, Y., Madden, A., Ding, Y., Tang, J., Sun, G.G., Zhang, N. and Zhou, E. (2019) 'Analyzing stock market trends using social media user moods and social influence', *Journal of the Association for Information Science and Technology*, Vol. 70, No. 9, pp.1000–1013 [online] https://doi.org/10.1002/asi.24173.

Li, M. and Wang, D. (2017) 'Insights into randomized algorithms for neural networks: practical issues and common pitfalls', *Information Sciences*, Vol. 382, No. 383, pp.170–178 [online] https://doi.org/10.1016/j.ins.2016.12.007.

Li, X., Chang, D., Ma, Z., Tan, Z.H., Xue, J.H., Cao, J., Yu, J. and Guo, J. (2020) 'OSLNet: deep small-sample classification with an orthogonal softmax layer', *IEEE Transactions on Image Processing*, Vol. 29, pp.6482–6495 [online] https://doi.org/10.1109/TIP.2020.2990277.

Li, Y. and Ma, W. (2010) 'Applications of artificial neural networks in financial economics: a survey', *IEEE International Symposium on Computational Intelligence and Design*, pp.211–214 [online] https://doi.org/10.1109/ISCID.2010.70.

Lipton, Z.C., Berkowitz, J. and Elkan, C. (2015) *A Critical Review of Recurrent Neural Networks for Sequence Learning*, arXiv, 1506.00019 [online] https://arxiv.org/abs/1506.00019.

Liu, S., Zhang, X., Wang, Y. and Feng, G. (2020) 'Recurrent convolutional neural kernel model for stock price movement prediction', *PLoS ONE*, Vol.l 15, No. 6, p.e0234206 [online] https://doi.org/10.1371/journal.pone.0234206.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y. and Alsaadi, F.E. (2017) 'A survey of deep neural network architectures and their applications', *Neurocomputing*, Vol. 234, pp.11–26 [online] https://doi.org/10.1016/j.neucom.2016.12.038.

Lo, A.W. (2016) *Moore's Law vs. Murphy's Law in the Financial System: Who's Winning?*, BIS Working Paper, No. 564, pp.564.1–564.38 [online] https://ssrn.com/abstract=2789737.

Lo, A.W. and Mackinlay, A.C. (1999) *A Non-Random Walk Down Wall Street*, Princeton University Press, Princeton, Oxford [online] https://doi/10.2307/j.ctt7tccx.

Long, J., Chen, Z., He, W., Wu, T. and Ren, J. (2020) 'An integrated framework of deep learning and knowledge graph for prediction of stock price trend: an application in Chinese Stock Exchange market', *Applied Soft Computing*, Vol. 91, p.106205 [online] https://doi.org/10.1016/j.asoc.2020.106205.

Luc, P., Couprie, C., Chintala, S. and Verbeek, J. (2016) *Semantic Segmentation using Adversarial Networks*, arXiv, 1611.08408 [online] https://arxiv.org/abs/1611.08408.

Malakar, P., Balaprakash, P., Vishwanath, V., Morozov, V. and Kumaran, K. (2018) 'Benchmarking machine learning methods for performance modeling of scientific applications', *IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, pp.33–44 [online] https://doi.org/10.1109/PMBS.2018.8641686.

Maltoudoglou, L., Boutalis, Y. and Loukeris, N. (2015) 'A fuzzy system model for financial assessment of listed companies', *IEEE International Conference on Information, Intelligence, Systems and Applications*, IEEE, pp.1–6 [online] https://doi.org/10.1109/IISA.2015.7388049.

Maniatopoulos, A., Gazis, A. and Mitianoudis, N. (2020a) *U.S. Stock Market Data – Dow Jones (501 Companies, 2010–2016)*, Mendeley Data v1 [online] https://doi.org/10.17632/hfzvhd2f5p.1.

Maniatopoulos, A., Gazis, A., Palikaras, V.P. and Mitianoudis, N. (2020b) 'Artificial neural network performance boost using probabilistic recovery with fast cascade training', *NAUN International Journal of Circuits, Systems and Signal Processing*, Vol. 14, pp.847–854 [online] https://doi.org/10.46300/9106.2020.14.110.

Manojlovic, T. and Stajduhar, I. (2015) 'Predicting stock market trends using random forests: a sample of the Zagreb Stock Exchange', *IEEE International Convention on Information and Communication Technology, Electronics and Microelectronics*, pp.1189–1193 [online] https://doi.org/10.1109/MIPRO.2015.7160456.

Merello, S., Ratto, A.P., Oneto, L. and Cambria, E. (2019) 'Ensemble application of transfer learning and sample weighting for stock market prediction', *IEEE International Joint Conference on Neural Networks*, pp.1–8 [online] https://doi.org/10.1109/IJCNN.2019.8851938.

Mingyue, Q. and Yu, S. (2016) 'Predicting the direction of stock market index movement using an optimized artificial neural network model', *PLoS ONE*, Vol. 11, No. 5, p.e0155133 [online] https://doi.org/10.1371/journal.pone.0155133.

Mingyue, Q., Cheng, L. and Yu, S. (2016) 'Application of the artifical neural network in predicting the direction of stock market index', *International Conference on Complex, Intelligent, and Software Intensive Systems*, pp.219–223 [online] https://doi.org/10.1109/CISIS.2016.115.

Moghaddam, A.H., Moghaddam, M.H. and Esfandyari, M. (2016) 'Stock market index prediction using artificial neural network', *Journal of Economics, Finance and Administrative Science*, Vol. 21, No. 41, pp.89–93 [online] https://doi.org/10.1016/j.jefas.2016.07.002.

Mostafa, M.M. (2010) 'Forecasting stock exchange movements using neural networks: empirical evidence from Kuwait', *Expert Systems with Applications*, Vol. 37, No. 9, pp.6302–6309 [online] https://doi.org/10.1016/j.eswa.2010.02.091.

Mukkamala, M.C. and Hein, M. (2017) 'Variants of RMSProp and Adagrad with logarithmic regret bounds', *ACM Proceedings of the International Conference on Machine Learning*, Vol. 70, pp.2545–2553 [online] https://dl.acm.org/doi/10.5555/3305890.3305944.

Nakano, M., Takahashi, A. and Takahashi, S. (2018) 'Bitcoin technical trading with artificial neural network', *Physica A: Statistical Mechanics and its Applications*, Vol. 510, pp.587–609 [online] https://doi.org/10.1016/j.physa.2018.07.017.

Ntakaris, A., Magris, M., Kanniainen, J., Gabbouj, M. and Iosifidis, A. (2018) 'Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods', *Journal of Forecasting*, Vol. 37, pp.852–866 [online] https://doi.org/10.1002/for.2543.

Nunes, M., Gerding, E., McGroarty, F. and Niranjan, M. (2019) 'A comparison of multitask and single task learning with artificial neural networks for yield curve forecasting', *Expert Systems with Applications*, Vol. 119, pp.362–375 [online] https://doi.org/10.1016/j.eswa.2018.11.012.

Nuseir, M.T. (2018) 'How big data is used in expanding marketing activities', *International Journal of Economics and Business Research*, Vol. 16, No. 4, pp.466–475 [online] https://dx.doi.org/10.1504/IJEBR.2018.095342.

Oliveira, F.A., Nobre, C.N. and Zarate, L.E. (2013) 'Applying artificial neural networks to prediction of stock price and improvement of the directional prediction index – case study of PETR4, Petrobras, Brazil', *Expert Systems with Applications*, Vol. 40, No. 18, pp.7596–7606 [online] https://doi.org/10.1016/j.eswa.2013.06.071.

Önder, E., Bayır, F. and Hepsen, A. (2013) 'Forecasting macroeconomic variables using artificial neural network and traditional smoothing techniques', *Journal of Applied Finance and Banking*, Vol. 3, No. 4, pp.73–104 [online] https://dx.doi.org/10.2139/ssrn.2264379.

Parsva, P. (2020) 'Investigating the impact of trade shocks on production in Iranian manufacturing industries', *International Journal of Economics and Business Research*, Vol. 20, No. 4, pp.391–406 [online] http://dx.doi.org/10.1504/IJEBR.2020.111081.

Pavlidis, N.G., Tasoulis, D.K. and Vrahatis, M.N. (2003) 'Financial forecasting through unsupervised clustering and evolutionary trained neural networks', *IEEE 2003 Congress on Evolutionary Computation*, Vol. 4, pp.2314–2321 [online] https://doi.org/10.1109/CEC.2003.1299377.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M.P., Shyu, M.L., Chen, S.C. and Iyengar, S.S. (2018) 'A survey on deep learning: algorithms, techniques, and applications', *ACM Computer Surveys*, Vol. 51, No. 5, p.92 [online] https://doi.org/10.1145/3234150.

Priyadarshini, R., Barik, R.K., Panigrahi, C., Dubey, H. and Mishra, B.K. (2020) *Deep Learning and Neural Networks: Concepts, Methodologies, Tools, and Applications*, pp.654–666 [online] http://doi:10.4018/978-1-7998-0414-7.

Qiu, M., Song, Y. and Akagi, F. (2016) 'Application of artificial neural network for the prediction of stock market returns: the case of the Japanese stock market', *Chaos, Solitons and Fractals*, Vol. 85, pp.1–7 [online] https://doi.org/10.1016/j.chaos.2016.01.004.

Ramezanian, R., Peymanfar, A. and Ebrahimi, S.B. (2019) 'An integrated framework of genetic network programming and multi-layer perceptron neural network for prediction of daily stock return: an application in Tehran Stock Exchange market', *Applied Soft Computing*, Vol. 82, p.105551 [online] https://doi.org/10.1016/j.asoc.2019.105551.

Rosenblatt, F. (1962) *Principles of Neurodynamics, Perceptrons and The Theory of Brain Mechanisms*, VG-1196-G-8, Cornell Aeronautical Laboratory, Buffalo, NY [online] https://cds.cern.ch/record/239697.

Sazli, M.H. (2006) 'A brief review of feed-forward neural networks', *Communications Faculty of Science University of Ankara*, Vol. 50, No. 1, pp.11–17 [online] https://doi.org/10.1501/commua1-2_0000000026.

Schmidt, B. and Hildebrandt, A. (2017) 'Next-generation sequencing: big data meets high performance computing', *Drug Discovery Today*, Vol. 22, No. 4, pp.712–717 [online] https://doi.org/10.1016/j.drudis.2017.01.014.

Scholtus, M., Dijk, D. and Frijns, B. (2014) 'Speed, algorithmic trading, and market quality around macroeconomic news announcements', *Journal of Banking and Finance*, Vol. 38, pp.89–105 [online] https://doi.org/10.1016/j.jbankfin.2013.09.016.

Selim, H. (2009) 'Determinants of house prices in Turkey: hedonic regression versus artificial neural network', *Expert Systems with Applications*, Vol. 36, No. 2, pp.2843–2852 [online] https://doi.org/10.1016/j.eswa.2008.01.044.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E.A., Menon, V.K. and Soman, K.P. (2017) 'Stock price prediction using LSTM, RNN and CNN-sliding window model', *IEEE International Conference on Advances in Computing, Communications and Informatics*, pp.1643–1647 [online] https://doi.org/10.1109/ICACCI.2017.8126078.

Sermpinis, G., Karathanasopoulos, A., Rosillo, R. and Fuente D. (2019) 'Neural networks in financial trading', *Annals of Operations Research* [online] https://doi.org/10.1007/s10479-019-03144-y.

Sezer, O.B. and Ozbayoglu, A.M. (2018) 'Algorithmic financial trading with deep convolutional neural networks: time series to image conversion approach', *Applied Soft Computing*, Vol. 70, pp.525–538 [online] https://doi.org/10.1016/j.asoc.2018.04.024.

Sharma, D., Misra, V. and Pathak, J.P. (2021) 'Emergence of behavioural finance: a study on behavioural biases during investment decision-making', *International Journal of Economics and Business Research*, Vol. 21, No. 2, pp.223–234 [online] http://dx.doi.org/10.1504/IJEBR.2021.113140.

Shen, J. and Shafiq, M.O. (2020) 'Short-term stock market price trend prediction using a comprehensive deep learning system', *Springer Journal of Big Data*, Vol. 7, p.66 [online] https://doi.org/10.1186/s40537-020-00333-6.

Song, J. (2019) *Predicting Individual Stock Returns using Optimized Neural Networks*, Aalto University [online] http://urn.fi/URN:NBN:fi:aalto-201907144311.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) 'Dropout: a simple way to prevent neural networks from overfitting', *Journal of Machine Learning Research*, pp.1929–1958 [online] https://dl.acm.org/doi/abs/10.5555/2627435.2670313.

Stoean, C., Paja, W., Stoean, R. and Sandita, A. (2019) 'Deep architectures for long-term stock price prediction with a heuristic-based strategy for trading simulations', *PLoS ONE*, Vol. 14, No. 10, p.e0223593 [online] https://doi.org/10.1371/journal.pone.0223593.

Symeonidis, S., Effrosynidis, D. and Arampatzis, A. (2018) 'A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis', *Expert Systems with Applications*, Vol. 110, pp.298–310 [online] https://doi.org/10.1016/j.eswa.2018.06.022.

Temponeras, G.S., Alexandropoulos, S.N., Kotsiantis, S.B. and Vrahatis, M.N. (2019) 'Financial fraudulent statements detection through a deep dense artificial neural network', *International Conference on Information, Intelligence, Systems and Applications*, pp.1–5 [online] https://doi.org/10.1109/IISA.2019.8900741.

Teräsvirta, T., Dijk, D. and Medeiros, M.C. (2005) 'Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: a re-examination', *International Journal of Forecasting*, Vol. 21, No. 4, pp.755–774 [online] https://doi.org/10.1016/j.ijforecast.2005.04.010.

Terna, P. (1992) *Artificial Neural Networks: Microeconomic Experiments by Neural Networks*, pp.1339–1342, Elsevier [online] https://doi.org/10.1016/B978-0-444-89488-5.50109-3.

Thakur, G.S.M., Bhattacharyya, R. and Mondal, S.S. (2016) 'Artificial neural network based model for forecasting of inflation in India', *Fuzzy Information and Engineering*, Vol. 8, No. 1, pp.87–100 [online] https://doi.org/10.1016/j.fiae.2016.03.005.

Tkáč, M. and Verner, R. (2016) 'Artificial neural networks in business: two decades of research', *Applied Soft Computing*, Vol. 38, pp.788–804 [online] https://doi.org/10.1016/j.asoc.2015.09.040.

Tsekouras, G.E., Trygonis, V., Maniatopoulos, A., Rigos, A., Chatzipavlis, A., Tsimikas, J., Mitianoudis, N. and Velegrakis, A.F. (2018) 'A Hermite neural network incorporating artificial bee colony optimization to model shoreline realignment at a reef-fronted beach', *Neurocomputing*, Vol. 280, pp.32–45 [online] https://doi.org/10.1016/j.neucom.2017.07.070.

Vargas, M.R., Lima, B.S.L.P. and Evsukoff, A.G. (2017) 'Deep learning for stock market prediction from financial news articles', *IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications*, pp.60–65 [online] https://doi.org/10.1109/CIVEMSA.2017.7995302.

Vargas, M.R., Lima, B.S.L.P. and Evsukoff, A.G. (2018) 'Deep learning for stock market prediction using technical indicators and financial news articles', *IEEE International Joint Conference on Neural Networks*, pp.1–8 [online] https://doi.org/10.1109/IJCNN.2018.8489208.

Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X. and Wang, F.Y. (2017) 'Generative adversarial networks: introduction and outlook', *IEEE/CAA Journal of Automatica Sinica*, Vol. 4, No. 4, pp.588–598 [online] https://doi.org/10.1109/JAS.2017.7510583.

Wei, H., Nakamori, Y. and Wang, S.Y. (2005) 'Forecasting stock market movement direction with support vector machine', *Computers and Operations Research*, Vol. 32, No. 10, pp.2513–2522 [online] https://doi.org/10.1016/j.cor.2004.03.016.

Welch, T.F. and Widita, A. (2019) 'Big data in public transportation: a review of sources and methods', *Transport Reviews*, Vol. 39, No. 6, pp.795–818 [online] https://doi.org/10.1080/01441647.2019.1616849.

Worasucheep, C. (2016) 'A stock price forecasting application using neural networks with multi-optimizer', *IEEE International Workshop on Computational Intelligence and Applications (IWCIA)*, pp.63–68 [online] https://doi.org/10.1109/IWCIA.2016.7805750.

Yu, Y., Si, X., Hu, C. and Zhang, J. (2019) 'A review of recurrent neural networks: LSTM cells and network architectures', *Neural Computation*, Vol. 31, No. 7, pp.1235–1270 [online] https://doi.org/10.1162/neco_a_01199.

Zaremba, W., Sutskever, I. and Vinyals, O. (2014) *Recurrent Neural Network Regularization*, arXiv, 1409.2329 [online] https://arxiv.org/abs/1409.2329.

Zhang, L., Wang, S. and Liu, B. (2018) 'Deep learning for sentiment analysis: a survey', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 8, p.e1253 [online] https://doi.org/10.1002/widm.1253.

Zhong, X. and Enke, D. (2017) 'A comprehensive cluster and classification mining procedure for daily stock market return forecasting', *Neurocomputing*, Vol. 267, pp.152–168 [online] https://doi.org/10.1016/j.neucom.2017.06.010.

Zhong, X. and Enke, D. (2019) 'Predicting the daily return direction of the stock market using hybrid machine learning algorithms', *Financial Innovation*, Vol. 5, No. 24 [online] https://doi.org/10.1186/s40854-019-0138-0.

Zhou, B. (2019) *Deep Learning and the Cross-Section of Stock Returns: Neural Networks Combining Price and Fundamental Information*, SSRN [online] https://dx.doi.org/10.2139/ssrn.3179281.