How random are intraday stock prices? Evidence from deep learning

Citation for published version:
Ibikunle, G & Möws, B 2017 'How random are intraday stock prices? Evidence from deep learning' pp. 1-59.

How Random are Intraday Stock Prices? Evidence from Deep Learning

GBENGA IBIKUNLE* BENJAMIN MÖWS†

Abstract Standard methods and theories in finance are ill equipped to capture complex data interactions presented in financial prediction problems. Deep learning approaches however offer more useful insights into these complex big data interactions. In this paper, using deep-layered feedforward neural networks, which applies econometrically constructed gradients, we learn and exploit time-shifted correlations among S&P 500 stocks to predict intraday and daily stock price movements for target stocks with only other stocks' lagged prices as inputs. Our findings show that time-shifted correlations can be exploited to predict stock prices; our model is also consistent in volatile markets.

Keywords: Machine Learning, Deep Learning, Artificial Intelligence, Asset Pricing, Big Data, Stock Price Prediction, Decision Analytics

* University of Edinburgh and European Capital Markets Cooperative Research Centre, Pescara: University of Edinburgh Business School, 29 Buccleuch Place, Edinburgh EH8 9JS, United Kingdom; e-mail: Gbenga.Ibikunle@ed.ac.uk; phone: +441316515186.

† AI & Machine Learning Group, Baillie Gifford & Co: Calton Square, 1 Greenside Row, Edinburgh EH1 3AN, United Kingdom

1. Introduction

Machine learning involves using data to train a model in order to now use that model to make predictions from new data; in this paper, we deploy a form of machine learning known as deep learning. Machine/deep learning approaches to stock price prediction rely on the assumption that stock price time series are systematically linked, such that previous prices contain substantial relevant information that could be used to predict future price trends (White, 1988). However, this assumption stands in direct violation of the efficient-market hypothesis, which describes stock market mechanics as being largely *informationally* efficient (Fama, 1965, 1970), and even more so in a high frequency trading environment prevalent in most developed markets today (see Brogaard et al., 2014). As an example, if the semi-strong form of the efficient-market hypothesis holds in a market, the only source of changes in that market's stock prices should be new and unpredictable information, as markets would have already reflected all previously available information. This notion of informational efficiency is consistent with the random walk hypothesis, which states that stock markets follow a random walk and are thus inherently unpredictable (Kendall and Bradford Hill, 1953; Cootner, 1964; Malkiel, 1973). Consequently, if stock markets merely following a random walk, it would be impossible to forecast price trends in a manner that results in selected portfolios outperforming the market over long periods of time and without a proportionately higher risk exposure.

In this paper, we postulate that price series in historical stock market data contain time-shifted correlations that can be successfully exploited with deep-layered feed-forward neural network architectures by using trend approximations as features (inputs). This can be used in making above-average price trend predictions without the data of the target stock as an input, while taking directional trend distributions for time intervals into account. Our hypothesis is linked to Ho and Stoll's (1983) microstructure model, showing the connection between quote shifts in a stock and inventory changes in other stocks. They show that quote shifts in stock *a*, which is in reaction to a transaction in stock *b* is based on $cov(R_a, R_b)/\sigma^2(R_b)$. This portfolio view of stock trading is consistent with commonly deployed diversification strategies in finance and has led to the rise of instruments such as exchange traded funds (ETFs), which offer cheap means of risk diversification. Indeed, it is instructive that the most liquid financial instrument, the SPDR, is an ETF.

We find evidence of the presence of time-shifted correlations in S&P 500 stocks in contradiction to both the random walk hypothesis and the efficient-market hypothesis in all three forms. We also find evidence of the viability of using deep-layered neural networks for trend predictions in inter-correlated time series. Our experiments outperform the predefined baselines for strict statistical key performance indices (KPIs). Furthermore, predictions of one stock's trend changes based on other stocks' price trend gradients in the preceding time step show an improved accuracy for larger time intervals, with average and maximum accuracies of 56.02% and 63.95% respectively for one-day predictions. Our findings are novel in that we exploit correlations of a target stock with other stocks in the same economy, without recourse to industry classifications, in making predictions about the target stock's price evolution. The results are consistent in financial crisis situations and demonstrate that our framework is able to exploit existing and inter-temporal correlations in a highly volatile market environment.

We ensure that our evidence holds up to scientific scrutiny by creating a high-quality set of features to train the models. The dataset employed is cleaned and pre-processed in a manner allowing for a perfect alignment of different stocks' observations for all time steps/intervals. We also ensure that the finalised models are shown to learn and successfully act on non-random correlations with above-average predictions of trend changes. Validation measures also confirm that the models outperform predetermined baselines that exclude the simple learning of distributions or frequencies, and adhere to statistical key performance indices. With our methodological approach, we also demonstrate the utility of linear regression derivatives as inputs for time series-based investigations in deep learning.

The growing interest in research dealing with the usage of artificial neural networks for stock market prediction is facilitated by the availability of large historical stock market trading data/messages. Such data usually takes the form of a time series and thus classical approaches to time series analysis are currently widespread within the investment industry (Clarke et al., 2001). This configuration, together with the existence of related hypotheses, makes the prediction of stock price changes based on historical data a good use case for trend forecasting in complex and potentially inter-correlated time series.

Although a small number of papers on deep learning models for stock price prediction have been published in recent years, compelling and thorough evidence for the feasibility is still outstanding. Deep learning recently started to be applied to stock price time series in order to improve simple strategies such as momentum trading, with results indicating the feasibility of such methods (Takeuchi and Lee, 2013). Only two existing studies could be directly compared to this paper; these are Takeuchi and Lee (2013) and Batres-Estrada (2015). Both papers employ deep-layered neural network models for a binary month-wise trend prediction of target stocks, based on historical stock market data of the preceding 12 months, with resulting accuracies of 53.36% and 52.89% respectively (see also Heaton et al., 2016; Krauss et al., 2017). Our approach is different since we address the prediction of up- and downward evolution of the trend gradient instead of the target gradient's sign. Furthermore, we concentrate on intraday intervals, which are more relevant in today's high-tech stock market environments. Specifically, we predict stock price movements over one-day, one-hour and half-hour time intervals, while previous research based their analysis on months-long intervals, or as it is in the case of Krauss et al. (2017), day intervals only. The accuracies we find with our approach for one-day and one-hour predictions are higher than those reported by Takeuchi and Lee (2013) and Batres-Estrada (2015) with 56.02% and 53.95% respectively. Our modelling approach also proves capable of extracting required market prediction information in high volatility situations such as a financial markets crisis.

While research on gradients of regression lines performed on stock price intervals is sparse, the utilisation of directional derivatives of wavelets has been successfully introduced in a wide number of research areas such as natural language processing (see Gibson et al., 2013). Mierswa (2004) uses, among other features, the gradients of linear regressions of the frequency spectrum over a moving window as input features for audio classification, explicitly treating the data as multivariate time series. Although a decision tree and a support vector machine are used to evaluate the viability of the selected features, this represents an instance of other research utilising such linear regression derivatives over time intervals as features. In another approach to time series classification, Górecki and Łuczak (2013) build on earlier research by Keogh and Pazzani (2001) on the addition of derivatives to dynamic time warping, where the latter is a method to measure the similarity of temporal sequences with potentially different speeds (Berndt and Clifford, 1994). The proposal of using a distance metric based on the discrete derivatives of different time series is later successfully used in an experimental implementation for a *k*-Nearest Neighbours algorithm (*k*-NN) classification (Górecki and Łuczak, 2014).[2] Generally, research on first derivatives for classification tasks in time series as features for machine learning is sparse. This is also the case even if not viewed in the narrower context of step-wise linear regression gradients over set intervals to search for time-shifted complex correlations in a large number of time series with deep-layered artificial neural networks. This gap in the literature allows this paper to spearhead applied research in this direction, with potential implications for a wider utilisation of this methodology based on deep learning with feedforward neural network models for time-shifted correlations.

Our successful application validates the approach of using deep-layered feedforward neural networks for exploiting time-shifted and highly complex correlations between time series in the area of trend prediction. For that reason, this paper furthers the understanding of deep learning in this specific context. One of the main concerns for an effective application of deep-layered neural networks is the choice and implementation of feature engineering, which often consumes large parts of a machine learning project's outlay and relies on domain knowledge for the identification of good data representations (Najafabadi et al., 2015). As linear regressions on time series are simple measurements of trends, such regressions hold the potential of being used as input features extracted from the respective time series. They can be used in the input layer of a feedforward neural network, the results have to be further reduced to a vector per training example while maintaining a rich-enough representation, for example, as the gradients computed through the first derivatives of linear regressions. The gradient in such a case does not represent the value of a time series at a certain point, but the strength of the upward or downward movement as approximated by the regression. It is expected that the gradients of such simple trend approximations contain enough information to retain complex correlations between time series at different points in time, and deep-layered feedforward neural networks are able to extract this information.

Changes in stock markets are fuelled by human decisions based on beliefs about a stock's future performance. Such beliefs are also influenced by investors anticipating the actions of other investors, especially in a period of high uncertainty. Examples include the sharp fall in the prices of airline stocks in the aftermath of the September 11 attacks, and the

[2] See also Baggenstoss, (2015); *k*-NN is a non-parametric approach for classification and regression.

negative effects of acknowledgements of a CEO's deteriorated health (Drakos, 2004; Perryman et al., 2010). This makes markets inherently noisy and prone to fluctuations via overreactions and dynamic reinforcement (Chen et al., 1986). Such inconsistencies in the valuation of stock prices has been the subject of a long-standing academic debate centred on the efficient market and random walk hypotheses, i.e. whether such time-shifted correlations in the stock markets exist at all. Should such correlations be present in historical information, they must also be detectable despite potentially poor data quality, and through stock trading noise.

The application of the proposed approach regarding the learning of time-shifted correlations between time series to stock market data in a sense is an empirical test of both the efficient market and random walk hypotheses. Our results deliver rigorously tested empirical evidence against the latter hypothesis. This is because the assumption of stock prices over time as random walks effectively excludes the possibility of exploitable information in historical stock market data. Thus our results are inconsistent with Sitte and Sitte (2002), who argue in favour of the existence of a random walk specifically for S&P 500 stocks due to the inability of artificial neural networks to extract any information resulting in over-average predictions for those stocks. The results are however consistent with previous, albeit weak, evidence for the absence of a random walk in financial time series via the use of artificial neural networks as presented by Darrat and Zhong (2000). The consistency of the efficient-market hypothesis with the random walk hypothesis also implies that our findings are in direct contradiction with the efficient-market hypothesis, which is widely supported by the preponderance of finance academic research (see as examples, Fama, 1970; Doran et al., 2010).

The remainder of this paper is set out as follows: Section 2 discusses the progress made on predicting stock returns and prices and discusses the literature on deep neural networks and the functionality of deep learning models in finance; Section 3 presents the data and methodology, Section 4 discusses the results and findings while Section 5 concludes.

2. Predicting stock returns

Technical analysis involves making stock trading decisions on the basis of historical stock market data. The assumption behind its utilisation in the investment industry is that above-average risk-adjusted returns are possible when using past time series of stock information. While this assumption is inconsistent with the random walk hypothesis and all forms of the efficient-market hypothesis, Clarke et al. (2001) show that this practice is widespread in the investment industry. A meta-analysis by Park and Irwin (2004) also shows that the majority of studies on the topic of technical analysis report a profitability that undercuts the arguments of the efficient-market hypothesis. White (1988) hypothesised early that artificial neural networks could be successfully used to deliver empirical evidence against all the three forms of the efficient-market hypothesis, reporting an R^2 value of 0.175 when using a simple feedforward network of five previous days of IBM stock prices as inputs for a regression task (see also Saad, 1998). Zhang et al. (1997) find that artificial neural networks are especially suited to forecasting due to their unique characteristics, such as arbitrary function mapping, non-linearity and adaptability. Skabar and Cloete (2002) also employ a neural network model with just one hidden layer trained on both a collection of randomly generated data and a small subset of historical stock prices, and find a statistically significant superior return when stock market information is used. Research on artificial neural networks for stock market prediction does, however, remain sparse over the last two decades.

However, momentum trading strategies have been receiving a lot of attention in finance research in recent times. The apparent ability of momentum-based strategies to outperform the market are viewed as a premier anomaly within the framework of the efficient-market hypothesis (see Fama and French, 2008). Takeuchi and Lee (2013), exploiting the suggested efficacy of momentum trading, is, to our knowledge, the first published research on deep learning for stock market prediction. Drawing on the work of Hinton and Salakhutdinov (2006) focused on the construction of auto-encoders via stacked restricted Boltzmann machines for dimensionality reduction and feature learning, Takeuchi and Lee (2013) predict stock movements using historical stock market data of only the respective target stocks from a large set of NYSE stocks. With an average accuracy of 53.36%, their model delivers evidence of above-average returns by using features learned from the preceding 12-month period to predict the trend for the 13th month. This study thus serves as a baseline for subsequent research endeavours in stock price prediction through the use of deep learning. Takeuchi and Lee's (2013) approach is different from ours in that we examine price predictability for a target stock by using only alternate stocks' data.

Following the work of Takeuchi and Lee (2013), Batres-Estrada (2015) constructs a deep belief network composed of stacked restricted Boltzmann machines, followed by a feedforward artificial neural network with one hidden layer. The input and objectives are similar, with the previous 12 months' worth of a stock's log-returns as the input used to predict the 13th month's trend in a binary fashion, with the addition of daily log-returns for each day of a respective month. This approach results in an accuracy of 52.89% for the test set, which outperforms naïve baselines and a simple logistic regression, and also yields results that are comparable to Takeuchi and Lee (2013). Dixon et al. (2016) also implement a feedforward artificial neural network with five hidden layers for trinary classification, differing in an output that represents little or no change from the previously cited studies. Using data of CME-listed commodities and foreign exchange futures in five-minute intervals to generate a variety of engineered features like moving correlations, a single model is trained instead of a separate model for each target instrument resulting in an average accuracy of 42.0% for the investigated three-class prediction task. It should, however, be noted that no cross-validation is carried out, which would further validate the results for economic conclusions.[3] Even more recently, Krauss et al. (2017) investigate the effectiveness of deep neural networks (DNN), gradient-boosted-trees (GBT) and random forests (RAF) in the detection of statistical arbitrage. Their results indicate that when combining one each of DNN, GBT and RAF, one may obtain returns exceeding 0.45% per day before taking transaction costs into account.

Apart from the sparse literature on deep learning for time series-based stock market prediction, text-based prediction approaches, using machine learning models, are also gaining traction and have emerged as the predominant alternative during the past few years. The notion of using news articles, with new information, as opposed to historical market data, to predict stock prices was introduced by Lavrenko et al. (2000) and is a common baseline for subsequent research. A system devised by Schumaker and Chen (2009a), named *AZFinText*, which employs wide-spread news coverage, results in a directional accuracy of 57.1% for the best-

[3] Cross-validation is the splitting of the dataset into, e.g., 5 sets of approximately the same size. A model is then trained on 4 of the sets and tested on the fifth, alternating which sets are trained on and which is used for testing, so five tests are carried out over all instances of the data, and the results for the five models are then averaged. This is the "gold standard", so to speak, in machine learning, as it takes care of the possibility of testing models on a subset unrepresentative of the whole dataset. Cross-validation makes for more reliable and stable results; therefore, we apply the approach to our analysis in this paper (see Section 3.3.2).

performing model. The approach involves a support vector machine with a proper-nouns scheme instead of a simple bag-of-words approach in combination with a stock's current price as inputs, over a five-week period. A criticism of the approach is that five weeks-worth of information could fail to constitute a rigorous test of performance. In addition, it proves to be only successful within a twenty-minute time frame, which falls under the margin of earlier research concluding that the majority of market responses to new information experiences a time lag of approximately ten minutes (see Patell and Wolfson, 1984). Subsequent research, however, shows that AzFinText can outperform established quantitative funds (see Schumaker and Chen, 2009b).

Ding et al. (2015) propose the use of a neural tensor network to learn event embedding from financial news articles in order to feed that information into a deep convolutional neural network for a two-class prediction of a stock price's future movement. For this system, an accuracy of 65.9% is reported for 15 different S&P 500 stocks and daily trend predictions. No clear indication, however, is given as to how the reported stocks are selected. Related research by Fehrer and Feuerriegel (2015) employs recursive auto-encoders to extract sentiments from financial news headlines and companies' financial disclosure statements, resulting in an accuracy of 56.5% for the test set and predictions of stock price movements after a financial disclosure statement.

2.1. Background to our approach: deep neural networks

Artificial neurons are the fundamental building blocks of deep-layered feedforward neural networks, which are employed in this study. They were first proposed for solving computational problems by McGulloch and Pitts (1943) within the scope of thresholds for logical calculations. The basic idea is that a certain level of activation is necessary to make an artificial neuron fire instead of remaining dormant. Perceptrons are the next step in this evolution. Devised by Rosenblatt (1958), perceptrons are algorithms that implement a linear classification for binary distinctions and are thus the simplest example of a feedforward neural network. The mathematical formulation that takes place for a named perceptron can be summarised as follows:

$$f(\mathbf{x}) = \begin{cases} 1 \; if \; \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 \; else \end{cases} \tag{1}$$

Here, **w** and **x** denote vectors in ℝ, with **x** being the vector of inputs to an artificial neuron, whereas **w** is the vector of the respective weights for each separate input. b denotes a bias term which represents the artificial neuron's firing threshold, and *f(***x***)* is a Heaviside step function, i.e. a function that outputs 1 for a positive argument and 0 for a negative argument. The dot product of **w** and **x** can be formulated as:

$$\mathbf{w} \cdot \mathbf{x} = \sum_{n=1}^{N} w_i \, x_i \tag{2}$$

Feedforward neural networks are directed, acyclic graphs which use a set of artificial neurons to funnel inputs in one direction towards the outputs. In their commonly used form, such models are fully connected between neighbouring layers of artificial neurons, whereas no connections exist over multiple layers. Due to their antecedents, artificial neural networks are often still called single- or multi-layer perceptrons despite the term denoting a model consisting of just one artificial neuron (depending on their number of hidden layers). In this paper, however, the naming as perceptrons is mentioned only in the given context of a historical overview of the broader topic; the models are forthwith referred to as artificial neural networks. Figure 1 depicts a simple feedforward artificial neural network with no hidden layers.

INSERT FIGURE 1 ABOUT HERE

The input layer represents, in this form, the input vector used in Equations (1) and (2); in this case, values for four variables. The output layer represents the result that is obtained from running the values of the input layer through the model. In this basic form, the artificial neural network is equivalent to a linear regression, as each input is multiplied by a weight to obtain the corresponding output. In other forms, the weights are depicted as layers instead, but the former representation will be applied throughout this paper in order to guarantee a consistent reading process for all sections. Other types of neural network models exist for example, various kinds of recurrent neural networks and convolutional neural networks.[4]

Activation functions are utilised by artificial neurons in these models, allowing inputs to be transformed by using weights and, in the common case, a bias term. Apart from the Heaviside step function earlier stated, non-linear activation functions allow for the solution of non-trivial problems, as outputs are not constrained to logical values. Similarly, linearly increasing activation functions require a large number of artificial neurons for non-linear separation tasks, which makes them computationally challenging and expensive. Instead, commonly used activation functions are meant to increase in their output at first, but then gradually approach their limit in an asymptotic manner for higher values. A classic example of such a function is the sigmoid function depicted in Figure 2.

INSERT FIGURE 2 ABOUT HERE

In training artificial neural networks, sigmoid functions are a term applied to the special case of the logistic function shown in Figure 2, with a steepness value of $k = 1$ and a midpoint of $x_0 = 0$. The sigmoid function is calculated as follows:

$$sigm(\mathbf{x})_j = \frac{1}{1+e^{-k\cdot(x_j-x_0)}} \tag{3}$$

It is important to note that values for the sigmoid function levels out at 0 on the lower end, which can lead to a fast saturation of weights at the top layers of multi-layered artificial neural networks (Glorot and Bengio, 2010). An alternative is the use of the hyperbolic tangent function, which is similar to the sigmoid function, but is centred on 0 instead of 0.5, with a lower limit of –1 and the same upper limit of 1 for its values:

[4] The learning process discussed in this paper is restricted to the supervised version, i.e. the training of a model with already correctly labelled data. Other types of learning, such as unsupervised learning for unlabelled datasets, and reinforcement learning, find applications in a wide range of research areas as well.

$$tanh(\mathbf{x})_j = \frac{sinhx_j}{coshx_j} = \frac{e^{x_j}-e^{-x_j}}{e^{x_j}+e^{-x_j}} = \frac{1-e^{-2x_j}}{1+e^{-2x_j}} \tag{4}$$

Other often-used activation functions for training artificial neural networks include radial basis functions and rectified linear units (see Broomhead and Lowe, 1988; Nair and Hinton, 2010). Other proposals for activation functions specifically target the goal of a reduced computational cost, e.g. the squash function introduced by Elliott (1993). The softmax function also deserves a mention as an activation function; it serves as a widely-used way to interpret the outputs of neural network models used for classification tasks as probabilities (Bishop, 2006). The formula for this function, which is wide-spread in its application as a last layer of such models, is:

$$softm(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{n=0}^{N} e^{x_n}}, s.t.\, j \in \{1, 2, \ldots, N\} \tag{5}$$

The notable difference to the other functions earlier enumerated is that the utilisation of all inputs from the previous layer results in values between 0 and 1 for the softmax layer adding up to 1. These outputs can be expressed as probabilities of mutually exclusive classes, i.e. used as percentages of a 100% total for further computations. Hidden layers are additional layers between the output and the input layers that are shown in Figure 1, with one hidden layer.[5] Figure 3 depicts a simple feedforward artificial neural network with four inputs, a single hidden layer, and two outputs, which could be used for a binary classification problem.

INSERT FIGURE 3 ABOUT HERE

Backpropagation of error was developed as a method to time-efficiently train multi-layered artificial neural networks in the 1970s (Werbos, 1974). By using a predefined loss function's gradient w.r.t. all weights in a neural network model for optimisation methods, such

[5] The main advantage of using hidden layers is that the artificial neurons of such layers can process the full output of the previous layer, which turns the linear separations that a neural network model with no hidden layers implements into a non-linear process, allowing for greater differentiation capabilities.

as stochastic gradient descent, efficient training of multi-layered models becomes feasible (see also Rumelhart et al., 1986; Nielsen, 2015). In this process, loss functions are used to attach a value to the total error under a certain set of weights between layers. Using the example of the quadratic cost function, the total error for this case can be calculated with the following equation:

$$E = \frac{1}{2}\sum_i \sum_j \left(\hat{y}_{j,i} - y_{j,i}\right) \tag{6}$$

where j indexes the output units and i indexes the pairs of training examples and corresponding outputs, $\hat{y}$ and y denote the calculated outputs and actual labels respectively. In the forward pass of the input through the network model, the values for the neurons in each layer are calculated with the last layer's outputs, processed through the activation function, and the respective connection's weights and the layer's bias, as previously described. In the backward pass, the weights and the bias are then updated. Gradient descent is a common optimisation method, and gradient-based optimisers allow for the use of backpropagation. Weights and biases of a layer are updated as follows, with $w_{i,j}$ as a weight, b_1 as the layer's bias, and η as the chosen learning rate:

$$w_{j,i} = w_{j,i} - \eta \frac{\partial E}{\partial w_{j,i}} \tag{7}$$

$$b_l = b_l - \eta \frac{\partial E}{\partial b_l} \tag{8}$$

These expressions require the computation of the error w.r.t. a single weight or bias. Using the chain rule, the error can be propagated backwards through the neural network model, which gives the name to the method. For weights, the formula is:

$$\frac{\partial E}{\partial w^l_{j,i}} = \sum_{m_L, m_{L-1}, \dots, m_l} \frac{\partial C}{\partial a^L_{m_L}} \frac{\partial a^L_{m_L}}{\partial a^{L-1}_{m_{L-1}}} \frac{\partial a^{L-1}_{m_{L-1}}}{\partial a^{L-2}_{m_{L-2}}} \frac{\partial a^L_{m_L}}{\partial a^{L-1}_{m_{L-1}}} \cdots \frac{\partial a^L_{m_{l+1}}}{\partial a^l_j} \frac{\partial a^l_j}{\partial w^l_{j,i}} \tag{9}$$

The case for computing the error w.r.t. a layer's bias is analogous to the above formula. Here, $w_{j,i}^{l}$ denotes a single weight in a specific layer l, with L indicating the final layer and a_{x}^{l} denoting the output of neuron x via the neuron's activation function in layer l. Put simply, the rate of change of the error is calculated w.r.t. a single weight, i.e. every connection between two artificial neurons in two adjacent layers has a rate represented by the gradient of a neuron's output w.r.t. the preceding neuron's output. For a path through the model, the product of this path's rates is the path's own rate.

2.2. The functionality of deep learning models

In recent years, artificial neural networking has emerged as a subject of increased public interest in machine learning due to the increasing ease of training deep-layered models with advanced computing equipment. Although deep learning, describing a high number of processing layers mostly used for deep-layered neural network models, has been criticised as not being more than a marketing term for long-established machine learning methods, its usage is now firmly established in academia (see Wlodarczak et al., 2015). For deep-layered feedforward artificial neural networks, these models' graph structures are identical to Figure 3, with the exception of a number of additional hidden layers. The primary advantage of such model architectures is their high non-linearity, which allows for the automatic identification of complex relationships in data. According to Glorot and Bengio (2010) deep-layered feedforward neural networks have the rather unique ability to extract features from features learned by previous hidden layers. This attribute reduces the need for time-intensive feature engineering, otherwise known as model selection.[6] While there are many varieties of deep neural network models, such as convolutional networks and deep belief networks, sufficiently deep feedforward models without such complexities attained the then-best performance of 99.75% accuracy on the MNIST handwritten digit database (Cireçan et al., 2010).

Despite their advantages, training deep-layered models brings difficulties that are addressed by refining the methods for the models described in the preceding section. Stochastic

[6] Glorot and Bengio (2010) also criticise the use of the sigmoid function in hidden layers, as its non-zero mean is shown to decelerate the learning process, and support the use of zero-mean activation functions like the hyperbolic tangent function.

gradient descent deals with the problem encountered in computing the loss function's gradients for all training, i.e. the computational expensiveness of the process leading to a slowdown in the training of a model. Stochastic gradient descent involves approximating the total error via the gradients for a random sample of training inputs. This changes Equations (7) and (8), with $x_1, x_2, \ldots, x_m$ as the sample and E_{x_k} as the cost for each data point from the sample to:

$$w_{j,i} = w_{j,i} - \frac{\eta}{m} \sum_k \frac{\partial E_{x_k}}{\partial w_{j,i}}, s.t.\ k \in \{1, 2, \ldots, N\} \tag{10}$$

$$b_l = b_l - \frac{\eta}{m} \sum_k \frac{\partial E_{x_k}}{\partial b_l}, s.t.\ k \in \{1, 2, \ldots, N\} \tag{11}$$

Momentum is very important for the training of deep learning architectures (see Sutskever et al., 2013), the purposes of which are to prevent the model from remaining at a local minimum and to accelerate the step size in the so-called shallow valleys.[7] If applied to stochastic gradient descent, with η denoting the amount of friction for the momentum and Δw representing the last iteration's weight update, Equation (10) is transformed to:

$$w_{j,i} = w_{j,i} - \frac{\eta}{m} \sum_k \frac{\partial E_{x_k}}{\partial w_{j,i}} + \eta \Delta w_{j,i}, s.t.\ k \in \{1, 2, \ldots, N\} \tag{12}$$

Overfitting, a machine learning model's tendency to incorporate noise and random error from the training set, thus leading to a larger generalisation error, is a concern here. Generalisation error, while not subject to being calculated for all possible unseen data, is approximated via a split of the available data into a training set and a test set, which serves as an empirical example of unseen data. It indicates, for a poor performance on the test set in relation to the training set, the presence of overfitting. In order to prevent overfitting, regularisation becomes necessary, a simple example of which is ‘early stopping’. By splitting the data three-fold into an additional validation set, the model's performance on data that is not part of the training is assessed after each epoch. If the accuracy on the validation set stagnates

[7] Shallow valleys are phases in which the steepest direction remains the same or similar for multiple iterations, but without a pronounced steepness.

over a predefined number of epochs, the training is terminated. Other, more sophisticated approaches include ℓ_1 and ℓ_2 regularisation for a sparsity-based solution.[8]

2.3. Trend analysis of financial time series

Clarke et al. (2001) and Gehrig and Menkhoff (2006) find that technical analysis, despite the permanence of the efficient-market hypothesis, is wide-spread in today's investment industry, although the term refers to simpler approaches in most of the cases. Exponential moving average, an infinite impulse response-based approach, is one of the dominant techniques used as a lagged indicator for stock trend forecasting. It is identical to exponential smoothing, a term more commonly used in the general study of time series, and can be calculated in a recursive manner as follows, with *i* as the time step indicator starting at 1 and α as the smoothing factor with $\alpha \in (0,1)$:

$$\begin{aligned} EMA_0 &= x_0 \\ EMA_i &= \alpha x_i + (1-\alpha)EMA_{i-1}, s.t. i > 0 \end{aligned} \tag{13}$$

Perceived patterns are among the other, less investigated methods that are used by technical analysts, e.g. the head-and-shoulders pattern. The latter is used as an indicator for a trend shift, using the negative value of the height denoted in Figure 4 as the target price for a trade initiated at the breakout point, which marks the pattern's completion. The lack of statistical research on such patterns has been criticised by Neftci (1991), noting that there is a disparity between the rigour of academic time series analysis and the decision-making of traders. Subsequent studies by Osler and Chang (1999) and Lo et al. (2000) show indicators of

[8] There are other mathematical considerations. For example, the Universal Approximation Theorem states that feedforward networks with one hidden layer act as approximators for continuous functions on closed subsets of the Euclidean space $\mathbb{R}^n$. Initially, the theorem was proven for three hidden layers by Irie and Miyake (1988), followed by a proof for one hidden layer and the sigmoid function by Cybenko (1989). Hornik (1991) thereafter concluded this process by showing that arbitrary non-constant activation functions fit the criteria of approximators. The reason for deep-layered models being used instead is that the theorem makes no statement about the learnability itself, and the necessary numbers of neurons and training examples are only given as finite respectively. In practical applications, deep-layered models have been shown to perform better on complex problems, although that does not invalidate the theorem itself.

applications for select currencies in foreign exchange markets, concluding that such patterns may hold some practical value.

INSERT FIGURE 4 ABOUT HERE

As earlier stated, stock price changes are, at their core, driven by human beliefs about the future performance of a stock; this aggregate belief stock is correlated with the sum of beliefs that market participants think other participants hold about future price evolution. To this extent, investment decisions quantify beliefs about the beliefs of other investors in the future, which is a process that can be continued iteratively into the future. Thus the influence of new information on investment decisions, and a variety of methods of varying sophistication are used to make stock price predictions, making stock time series inherently noisy. This property of stock market prices makes stock markets an interesting and challenging example of real-world time series created by a global conglomeration of human decisions. Hence, stock series are ripe for the application of deep learning methods, especially, since traditional methods and theories in finance appear ill-equipped for capturing complex data interactions presented in financial prediction problems.

3. Data and Methodology

3.1. Data

We obtain three sets of high frequency transactions data sampled at varying intraday intervals from the Thomson Reuters Tick History (TRTH) database. The variation in sampling allows for experiments over differing intraday time intervals. Each set of data contains the Reuters Identification Code (RIC) for each stock in the sample, transaction date, transaction time to the nearest millisecond, high/best bid price, low/best ask price, volume, and average transaction price.[9] The RICs for all stocks contained in the sample are listed in Appendix B.

The first dataset is sampled from 2011-04-04 to 2016-04-01, covering approximately five years' worth of stock transactions price information in 1-hour intervals for the S&P 500

[9] Three prices are obtained for robustness: (1) volume weighted average price, (2) average price and (3) last transaction price for the time interval; our inferences are unchanged irrespective of the price variable employed.

stocks, with 6,049,849 transactions for 505 stocks in total. The second dataset spans the same time frame and stocks sampled at 5-minute intervals, resulting in 47,853,642 transactions. The final dataset is also sampled at 1-hour intervals; however, sampling begins from 1996-04-04, which results in a larger dataset covering approximately 20 years and with 65,183,368 transactions, including the financial crisis period of 2007/2008.

The datasets show instances of missing values for the price variables in some stocks, especially earlier in the largest of the datasets, thus suggesting that there are no transactions recorded for those intervals. We follow Chordia et al. (2001) by replacing the missing values with the same-column entries of the preceding index or the index with the next non-missing value, depending on whether the former belongs to the same stock as identified by the X.RIC value. In order to generate feature vectors that can be used as inputs, the time stamps have to be aligned perfectly. Thus, missing observations are forbidden, unless that interval's observations are missing for every other stock in the sample. We therefore write a code in the R software in order to secure a time-wise alignment of observations for different stocks by substituting missing rows.[10] All observations with inexplicable and incomplete timestamps are also eliminated. Following the data cleaning process, the datasets are checked for a subset of stocks satisfying the requirement of being consistently present over a sufficiently large portion of the dataset's time frame divisible by the chosen number of time steps for the subsequent gradient calculation. The datasets' price information is extracted and transformed into a feature matrix with row-wise time steps and column-wise stocks. The final sets of cleaned data contain 449 stocks.

3.2. Statistical feature engineering

Feature engineering describes the manual selection and, if necessary, transformation of given datasets into new data that better represents the features needed for a chosen task (see

[10] The full R software code of the algorithm that we create is presented in Appendix A. The primary goal behind the design of the above approach to data alignment is to speed up the code execution, as naïve procedures with loops over the full datasets and copies of a full matrix for every missing observation, as well as a vectorised implementation of the latter, result in infeasible time estimates. By acting solely on time vector comparisons, with matrix shifts in the case of local time vector incompatibility, only for a specific pre-split stock's matrix, and operating on a pre-assigned matrix of sufficient dimensionality to allow for insertions instead of appending values in-process, a sufficient speed for datasets of the given scale in comparably short time frames is realised.

Ng, 2012). For this paper, a simple approach, linear regressions, is used to approximate the trends over given time intervals. Linear regressions assume a linear relationship between the regressand y_i and the regressors x_i. They follow the form below, with i denoting an observation, β_0 as the intercept, and $\epsilon_i$as the unobserved error term:

$$y_i \;=\; \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots + \beta_p x_{i,p} \;=\; x_i^T \beta + \epsilon_i \,, s.t.\; i \in \{1\,,2\,,\ldots,n\} \qquad (14)$$

Simple linear regressions are least-squares estimators of such models with one explanatory variable to fit a line that minimises the squared sum of the residuals. They take the form of a minimisation problem to find the intercept β_0 and the slope β_1:

$$\min_{\beta_0,\beta_1} Q(\beta_{0,}\beta_1)\,, s.t.\, Q(\beta_{0,}\beta_1) \;=\; \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_i)^2 \qquad (15)$$

By running a linear regression over each time series and time interval separately, and by taking the first derivative of the resulting equation, the trend gradients for single stocks and time intervals are obtained. Given aligned stock prices for *N* points in time and a chosen step size *s*, the resulting feature vector generated from a stock's price has the length *N*/*s*. Depending on the time frame covered by the dataset, limits apply to the size of intervals that can be chosen to still obtain a viable size for the training set. For the 5-year hourly interval dataset, gradients are computed for a time step size of 8, covering a whole trading day with 1,242 gradients per stock for 449 stocks. For the 5-year dataset with data in 5-minute intervals, two sets of gradients are computed: The first set covers a time step size of 12, resulting in 7,361 one-hour gradients for 449 stocks, whereas the second set covers a time step size of 6, with 14,725 gradients for each half hour and 449 stocks.[11] For the 20-year dataset with hourly values, daily gradients with a step size of 8 were computed for the years from 2003, preceding the global financial crisis, to and including 2008, resulting in 2,131 gradients for 298 stocks.

[11] As the code that implements the linear regression cuts the respective dataset to a length that allows for the computation over the prescribed amounts of values, the second set for half-hour gradients is slightly larger than double the first set.

3.3. Training the deep learning models

3.3.1. Libraries and programming environment

We choose Python for the implementation of the deep learning experiments, because it is a multi-purpose language with sufficient mathematical capabilities through extensions such as NumPy (see van Rossum, 1995). Python has become established as one of the primary programming languages for deep learning in libraries like Theano (Bergstra et al., 2011). Keras, also a highly modular library for neural networks written in Python, which is able to incorporate either Theano or, more recently, Google's TensorFlow as its basis (see Chollet, 2015), is chosen to build the experimental models. This is also due to its suitability for the fast prototyping of artificial neural networks.[12] We implement the experimental code using version 2.7.11 of Python, with IPython Notebook in its version 4.2.1 as the programming environment to allow for gradual code execution and an easily accessible overview of scrollable outputs, e.g. for training epochs.

3.3.2. Experimental setup and data splits

Figure 5 depicts a schematic overview of the experimental setup used in this paper. For the number n_{+1} of stocks that are made usable during the data cleansing and pre-processing, gradients of the price trends for each separate stock are computed in the feature engineering step as earlier described. The n gradients for one time step t_{-1} are then used as inputs to a feedforward artificial neural network that is fully connected for adjacent layers in order to predict whether the gradient of the left-out (n_{+1})st stock changes up- or downwards w.r.t. its gradient in the preceding time step t_{-1}. This setup ensures that the experiments test for time-shifted correlations between stocks instead of using a stock's own historical price information; thus, the data of the stock that is to be predicted is not part of the model's input. This is a major distinguishing aspect of this paper from preceding research on time series-based stock market prediction.

INSERT FIGURE 5 ABOUT HERE

[12] Theano is preferred over TensorFlow due to its position as an established machine learning library and the resulting variety of guidelines for its proper usage (Bahrampour et al., 2015).

As the experiments aim to find general correlations between stocks, five-fold cross-validation is used in order to reduce the variability of results, and frugally exploit the datasets (see Giovanni and Elder, 2010). After each five-way split and sorting into a training set of 80% of the dataset for the respective fold, another 25% of the training set is partitioned off as the validation set for the aforementioned early stopping procedure. Hence, a 60-20-20 split is used for each fold and stock. The experiments are run for all $n + 1$ stocks by looping over an index i for all column-wise stock gradients and splitting the matrix into the target gradients for stock i and the inputs for the rest of the columns. The time intervals are then shifted one step by clipping the first row of the input matrix and the last value of the output vector. The output vector is subsequently replaced by a binary one-hot representation indicating whether the gradients for each successive time interval for stock i are larger or smaller than for the preceding interval. Consistent with Takeuchi and Lee (2013) and Ding et al. (2015), two output nodes are chosen.

3.3.3. Input normalisation and regularisation

Normalising the inputs is necessary in order to address geometrical biases, distribute the importance of values equally and ensure that all values are situated in the same range in order to make them comparable for an efficient learning process. The training examples split from the dataset are normalised element-wise using min-max scaling as follows:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{16}$$

As earlier stated, regularisation approaches are a valuable way to address issues with overfitting, i.e. poor generalisation due to a specialisation on unneeded idiosyncrasies of the training set. Early stopping and ℓ_2 regularisation are used to prevent overfitting and unnecessary complexity, whereas momentum is applied in order to prevent stochastic gradient descent from terminating in small-spaced local minima. In addition, a dynamic learning rate decay is utilised to find a minimum along the optimiser's descent path:

$$x^* = rate \cdot \frac{1}{1+decay \cdot epoch} \quad (17)$$

3.3.4. Parameter tuning and model complexity

Commenting on the inherent vagueness of parameter tuning with regards to real-world applications, Snoek et al. (2012) concisely summarise the problems associated with the terrain by stating that machine learning algorithms:

"[...] frequently require careful tuning of model hyperparameters, regularization terms, and optimization parameters. Unfortunately, this tuning is often a "black art" that requires expert experience, unwritten rules of thumb, or sometimes brute-force search." (Snoek et al., 2012).

While such a statement sounds bleak, the parameters and hyperparameters that have to be set can be determined, or at least approximated.[13] Preliminary test runs show a rise in accuracy for up to five hidden layers, after which the learning process is hindered by the model's complexity in relation to the available number of training examples and thus provide no further improvement in accuracy. In order to achieve comparability in the models' performances over different experiments with all datasets, 400 nodes per hidden layer is identified as the level slightly below the number of inputs for the experiments, since the number of stocks used is 449 as stated above.

Preliminary tests run with 20 randomly chosen stocks for each of the three datasets with half-hour, one-hour and one-day gradients show the smallest test set error for this size of hidden layers, as measured in increments of 50 nodes for up to 800 nodes. In order to address potential memory issues, a mini-batch size of 100 is chosen, and each model is trained for 50 epochs, with early stopping as a regularisation measure as described in Section 2.2. Hyperbolic tangent functions serve as activation functions, with sigmoid functions at the output layer, and the model's weights are initialised as scaled samples from a Gaussian distribution (see He et al.

[13] The choices we make in this paper follow a combination of established scientific reasoning and preliminary experimentation, as well as experience in the application of deep learning architectures, and simple heuristics. These are employed in slowly increasing the model's complexity; further details can be found (Nielsen, 2015).

2015).[14] In addition, ℓ_2 regularisation is added to the regularisation process and chosen over ℓ_1 regularisation. This decision is driven by the encouragement to use all inputs to a certain degree, as a complex interdependence of the studied time series is assumed due to the failures of past approaches to identify simpler correlations. Through the introduction of decaying weights, and with the parameter λ defining the trade-off between the loss function and the penalty for larger weights, the previously introduced notation for unaltered gradient descent in Equation (7) is extended to:

$$w_{j,i} = w_{j,i} - \eta \frac{\partial E}{\partial w_{j,i}} - \eta \lambda w_{j,i} \tag{18}$$

Hyperbolic tangent functions from Equation (4) serve as activation functions, with sigmoid functions from Equation (3) at the output layer. The former choice is due to the discussed concerns regarding weight saturation, while the latter function is chosen over the softmax function due to the interpretability of the results as independent probabilities, and also because these results are not required to integrate to 1 as inputs for subsequent methods. The model's weights are initialised as scaled samples from a zero-mean Gaussian distribution to address the potential of vanishing or exploding gradients, with a standard deviation of $\sqrt{2/n_l}$, an initial bias of 0, and with n_l denoting the number of connections in a layer, allowing for an easy adaptation to future experiments with rectified linear units (Glorot and Bengio, 2010; He et al. 2015).

3.4. Further experiments and high volatility

3.4.1. Complexity reduction via bottleneck layers

In the context of the given problem, an interesting question is that of its general complexity, i.e. to find out to what number of variables the relevant data necessary for an acceptable accuracy can be reduced. In order to give an indication, a bottleneck layer consisting of a small number of neurons is inserted into the models for the daily predictions based on the

[14] Parameters and hyperparameters for the artificial neural networks used in the experiments have to be chosen and subsequently fixed for the experimental implementation.

5-year dataset with hourly values. This process is implemented for 1, 3, 5 and 10 nodes to see how the bottleneck's size influences the accuracy. This approach is favoured over using autoencoders (see Heaton et al., 2016) as a precedent to using their compression layer as inputs for a full model, because autoencoders learn a goalless reduction of their inputs.[15] For a bottleneck layer in the same model, the latter is forced to learn a compressed representation directed at a representation that is suited to the target predictions at hand, funnelling the model's learning process through the nodes of the respective bottleneck. Figure 6 shows a modification of the model depicted in Figure 3, featuring an exemplary additional bottleneck layer for complexity tests with one node marked **b**:

INSERT FIGURE 6 ABOUT HERE

3.4.2. Performance during a financial crisis

Previous studies applying machine learning approaches to stock market prediction are not tested for above-average results in selectively volatile market situations. However, Schumaker and Chen (2009a), who develop the AZFinText system, note such a test as a suggestion for further research. For time series-based prediction, high-volatility environments are more problematic contexts than for text analysis-based systems like AZFinText. This is because they rely solely on stock market information that is effectively in a state of unrest in that scenario. In order to test the gradient-based approach and the model implementation for such environments, data from 2003 to, and including, 2008 is extracted from the 20-year dataset with hourly values and used to compute daily gradients for the contained stock price information for 298 stocks. No cross-validation is performed for these experiments, as the test set has to represent a phase of enhanced volatility in the market. In order to achieve this, a test set from July 2007 to the end of 2008 is split from the set of gradients, with the previous 4.5 years serving as training data. This setup also more closely resembles an applicable prediction

[15] While similar to using a bottleneck, auto-encoders recreate their input as their output through a reduction layer without a target outside of the input recreation, whereas our chosen method reduces the complexity via the bottleneck layer with the explicit goal of making that reduction suitable for the computational task.

system, as only past data is used instead of identifying general correlations between combinations of different time periods through cross-validation.

Due to the large fluctuations in the dataset, and given that some stocks remain more stable than others during the financial crisis of 2007/2008, a higher variance of accuracies is expected. An accuracy above the baseline, which is expected to be higher than random chance due to the general negative trend in that time period, would deliver evidence for the persistence of correlations in high-volatility scenarios such as global financial markets crises.

3.5. Reliability and robustness analysis

For a validation of the results, baselines that address the market behaviour and the distributions of target vectors are necessary in order to find statistically significant evidence. The focus of this paper on time-shifted correlations between stocks in relation to economic theory, manifested in excluding the target stock data in the models' inputs, sets this work apart from research that aims to find the best-possible stock market predictions instead of correlations in stock market prices. For research focussed purely on prediction accuracy, baselines that represent naïve regression or classification approaches, or basic machine learning methods, are more suitable. In this paper, the threat of coincidences is partly approached through cross-validation, but a model's accuracy must also lie significantly above the accuracies of one-value and random predictions.

Another area of concern relates to the accuracy of random mock predictions; we address this by creating for each stock and fold in each model, a randomly shuffled copy of the predictions and thereafter test them against the correct targets in addition to the predictions themselves. These result in mock predictions with a class distribution identical to the actual predictions. Such copies can be employed in testing whether the model learns the distribution of the two output classes in the training set, which would result in very similar accuracies for the actual and mock predictions when compared to the test set's correct targets.

Another issue that needs to be contended with is that of the models learning to predict the dominant class of the respective training set. In order to address this potential issue, two targets are created for each stock and model, each containing exclusively one of the two classes. A model that learns more actionable information from its inputs than the dominant class of the training set needs to perform better on the test set than on both of those one-class mock targets when compared to the correct targets.

Finally, the average accuracies for each model over all stocks are expressed as the standard method to assess the predictive power of each model. These accuracies do not, however, give an indication as to whether the predictions' variations are too large to be considered successful in the context of this paper. For this reason, the accuracies for the three baseline mock predictions are also given, as well as the lower bound of each confidence interval. In addition, the p-values for the predictions' accuracies via an upper-tail test are calculated for each of the three baselines and an additional baseline that contains the highest accuracy among the three baselines for each stock, i.e. for each model. The null hypothesis H_0 in each case is that the predictions' accuracies are not significantly larger than the corresponding baseline, with a very strict significance level of $\alpha = 0.001$.

4. Results and Discussions

4.1. Primary experiments

In Panels A, B and C of Figure 7 we present box and whisker plots to visualise the results of the one-day, one-hour and half-hour gradient interval results respectively. For every experiment, the accuracies for the model and the baselines are given, as well as the p-values w.r.t. the means and the minimal difference for a 99.9% confidence interval. In Table 1 and subsequent tables, class 1 is the prediction that stock trends will change downwards, and class 2 is the prediction that stock trends will change upwards.

INSERT FIGURE 7 ABOUT HERE

In Figure 7, Panel A shows that the accuracy of 56.02% listed in Panel A of Table 1 lies significantly above all baselines, both for the means as measured by the p-values and the medians as indicated by the box plots, with neither the notches nor the boxes themselves overlapping. The first and third quartiles are, however, spread wider for the accuracy of the model. Panel B (Figure 7) shows the plot for the one-hour gradient intervals. With an accuracy of 53.95%, the model's accuracies exhibit the same increased variability as for one-day gradients, albeit at a smaller rate. The baselines' accuracies are centred more closely on 50% (see also Panel B of Table 1), which is consistent with the overall smaller spread of the accuracies for both the model and the baselines, trading accuracy for narrowness. Figure 7’s Panel C, showing the plot for the half-hour interval gradients suggests a model accuracy of 51.70%. Firstly, the distribution of the model's accuracies is also skewed towards lower values, i.e. more variability above the median. Secondly, there appears to be a pattern emerging here when all three plots are considered together. The model accuracies are lower for smaller gradient time intervals. However, the narrower boxes for the baselines also imply that the quartile ranges of the accuracy values are narrower, evidencing decreasing variability as time intervals considered become shorter.

INSERT TABLE 1 ABOUT HERE

The usage of p-values for validation purposes has to be viewed with caution, as criticisms of their often incorrect use have risen in recent years. (see Wasserstein and Lazar, 2016). Accordingly, the p-values presented in Table 1 are given in combination with other metrics such as the lower boundary for differences in means given a 99.9% confidence interval, notched box and whisker plots for median differences and quartile distributions, as well as accuracies for both the models on the one hand and the separate baselines on the other. In combination, these metrics deliver arguably strong evidence in support of the economic arguments implied in our hypothesis, i.e. that price series in historical stock market data contain time-shifted correlations that can be successfully exploited with deep-layered feedforward neural networks. This exploitation can result in above-baseline price trend predictions without even using the data of the target stock.

Notably, larger time intervals for gradient calculations and predictions based on the latter result in higher average accuracies, but with a trade-off in the form of an increased spread of the accuracies, i.e. a larger variance. It therefore seems to be easier for the models to learn correlations between gradients and make corresponding predictions for larger time steps, although more volatility w.r.t. to the variance is incorporated. A natural explanation for these differences is the presence of more noise in short-time stock observations, indicating that noise is smoothed out for regressions over larger intervals. Such noise events, such as bid-ask bounce, are microstructural in nature and are thus more prevalent in higher frequency data.

Although our results are in direct contradiction to two of the well-established hypotheses in modern finance, the findings presented are not exactly far-fetched. While meta-analyses should always be interpreted cautiously due to the possibility of publication biases, Park and Irwin (2004) find that a majority of the published research dealing with technical analysis for stock market prediction reports findings that indicate a critical view of the efficient market hypothesis in its strong form. As research on deep learning for time series-based stock market prediction is still very sparse, there are two existing studies that we can directly compare to this study; these are Takeuchi and Lee (2013) and Batres-Estrada (2015). Both papers use deep-layered neural network models for a binary month-wise trend prediction of target stocks, based on historical stock market data of the preceding 12 months, with resulting accuracies of 53.36% and 52.89% respectively. A direct comparison is still an approximation, as this paper addresses the prediction of up- and downward changes in the trend gradient instead of the target gradient's sign, but the binary prediction of either targets is comparable in their perceived difficulty and exclusion by the efficient-market hypothesis and the random walk hypothesis.

A further distinction of our study is the time frame on which the stock price predictions are based. This paper is based on one-day, one-hour and half-an-hour predictions instead of months as is the case with existing research. Focusing on higher frequency intervals is critical given the evolution of global financial markets in recent times, as trading now occurs in large volumes at sub-second frequencies and at much lower trade sizes (see Chordia et al., 2011). Thus, investment windows of even significantly large funds have shortened over time.

Despite the more limited features (mainly price gradients) employed in this paper, the accuracies for one-day and one-hour predictions surpass both Takeuchi and Lee (2013) and Batres-Estrada (2015), who obtain 56.02% and 53.95% respectively. The approach we use in this paper also proves capable of extracting additional information in high-volatility scenarios with imbalanced trend targets, albeit with a higher variance in the levels of accuracy than in non-volatile market environments. In summary, our approach outperforms both examples of binary trend prediction using past stock market time series, without the utilisation of information about the target stock itself, since the aim is to show time-shifted correlations of stock prices. It can therefore be inferred that our findings could potentially be applied for profitable stock trading.

4.2. Complexity and volatile environments

4.2.1. Results for models with bottlenecks

In order to show the effect of bottlenecks sizes, the box plots for four cases are depicted in Figure 8. The models' accuracies, as shown in Table 2, significantly increase with the increases in bottleneck sizes, with the maximum (10 bottleneck nodes) resulting in an accuracy slightly below the full model without a bottleneck. The quartile ranges of the box plots, remaining approximately symmetrical, also increase with a higher number of bottleneck nodes. The inclusion of a bottleneck layer in the neural network model should hinder its performance, with the number of nodes forming the bottleneck being the deciding factor. However, if judged via the notches of the box plots, the step-wise increases from one to three, then five and finally ten nodes each time lead to a statistically significant rise in performance. While a one-node bottleneck results in an average accuracy of 51.07%, the result for a ten-node bottleneck, with 55.03%, differs only by 0.99% from the accuracy of the same model and dataset without a bottleneck layer.

INSERT FIGURE 8 AND TABLE 2 ABOUT HERE

The possibility of the model not learning anything new after the introduction of bottleneck(s), i.e. the bottlenecked model performance being identical to a model with less hidden layers, has to be taken into account, but can be safely dismissed due to the accuracies for 3, 5 and 10 nodes being notably different from each other. The results suggest that a large portion of the information can be compressed in ten weighted variables half-way through the model, which gives a rough indication of the overall complexity of the prediction problem itself.

4.2.2. Robustness in a crisis/high volatility environment

Figure 9 presents plots for accuracies computed in a high volatility/crisis environment. Table 3 also shows the key performance estimates for the same scenario. Specifically, the results are for the trading environment during the global financial crisis of 2007/2008. The results show a large spread of the accuracies for different stocks. Notably, the medians and mean accuracies for one-class predictions show that the price trends more often change downwards during this modelled volatile scenario. In addition, the distribution of the model's accuracies is also skewed below the median; for example, the accuracies are spread higher upwards from the median, and the interquartile ranges are wider than for non-crisis scenarios.

INSERT FIGURE 9 AND TABLE 3 ABOUT HERE

While the average accuracies for predicting exclusively down- or upwards trend changes do not differ by more than 0.9% for the primary experiments, this difference grows to 11.90% for the crisis data. The model's high accuracy of 61.13% can be partly explained by this difference, as the mean accuracy for predicting exclusively negative gradient changes is 55.95% for the model's predictions, as shown in Table 3. The latter results are consistent with the general downwards-oriented trend of the whole market during a financial crisis, when confidence in the economy is expected to have fallen. Nevertheless, the additional accuracy of the model, along with the accuracy of the randomised mock predictions being below the exclusive predictions for negative trend changes, demonstrates that the model is able to exploit existent correlations in high-volatility environments such as the 2007/2008 financial crisis period. Thus, our findings are robust in the presence of extreme market stress.

5. Conclusion

In this paper, we test the hypothesis that stock price data contains time-shifted correlations that can be successfully exploited through the application of deep-layered feed-forward neural networks. Specifically, by using trend approximations as features (inputs) we can make higher than average price trend predictions without using the data of the target stock in the inputs. Our findings deliver evidence that there exists time-shifted correlations between the price behaviour of S&P 500 stocks. This finding contradicts the essence of the random walk hypothesis and the semi-strong and strong forms of the efficient market hypothesis while underscoring the viability of applying deep-layered neural networks for trend prediction in cross-correlated financial time series. Several robustness analytical steps are taken, including testing the model in an environment of extreme price volatility, all of which firmly support the primary findings that stock price data contains time-shifted correlations that can be successfully exploited. In addition, the levels of prediction accuracies we obtain for the S&P 500 stocks in our sample outperform the predefined baselines for strict statistical key performance indices.

Furthermore, we find that predictions of one stock's trend changes based on other stocks' price trend gradients in the preceding time interval improves in accuracy for larger time intervals, with average and maximum accuracies of 56.02% and 63.95% respectively for one-day predictions, all in excess of stated baseline predictions. The estimates retain large parts of their accuracy for a minimum of 10 nodes for mid-model bottleneck layers, and show equally above-baseline predictions in high-volatility market scenarios, albeit with the trade-off of a higher variance for different stocks. The large difference in accuracies for the model's predictions and the baselines cannot be explained by the shift to a skewed distribution in favour of negative trend changes, given that the training data does not contain such an imbalance.

The findings in this paper has implications for modern finance theory. It delivers vigorous evidence against the random walk and efficient market hypotheses. However, the postulations of the efficient market hypothesis may be adapted to allow for the findings as presented here. While indeed all three forms of the efficient-market hypothesis are inconsistent with the evidence, tweaking the weak form of the efficient market hypothesis could lead to it being consistent with our findings. Specifically, this would be for a case where the hypothesis allows for prediction methods that reliably outperform the market and can only be implemented by a sufficiently small number of investors so as not to result in a new aggregate equilibrium for a typical modern economy. With a negligible amount of capital involved in the context of the whole market, some agents, such as select quantitative hedge funds or individuals, could consistently realise above-average returns, thus reducing the weak form efficient market hypothesis to a context-based version. A time-specific weak form efficient market hypothesis would in effect acknowledge that it does not apply to the overall market, but does for a majority of the trading stakeholders due to restrictions regarding the methodology and the capital involved in deploying such strategies.

Finally, the results in this paper demonstrate the value of deep learning approaches to econometrics and show the utility of linear regression derivatives as features for time series-based investigations, thus offering a simple trend indicator with a high predictive value in order to further the understanding of highly complex time series correlations.

References

Baggenstoss, P. M. (2015), "Derivative-augmented features as a dynamic model for time-series", *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO 2015)*, pp. 958-962

Bahrampour, S.; Ramakrishnan, N.; Schott, L.; Shah, M. (2015), "Comparative study of deep learning software frameworks", *working paper*

Batres-Estrada (2015), "Deep learning for multivariate financial time series", *Master's thesis, KTH Royal Institute of Technology*

Bergstra, J.; Bastien, F.; Breuleux, O.; Lamblin, P.; Pascanu, R.; Delalleau, O.; Desjardins, G.; Warde-Farley, D.; Goodfellow, I.; Bergeron, A.; Bengio, Y. (2011), "Theano: Deep learning on GPUs with Python", *Journal of Machine Learning Research*, Vol. 1, No. 1, pp. 1-48

Berndt, D. J.; Clifford, J. (1994), "Using dynamic time warping to find patterns in time series", *AAAI Workshop on Knowledge Discovery in Databases*, pp. 229-248

Bicchetti, D.; Maystre, N. (2013), "The synchronized and long-lasting structural change on commodity markets: Evidence from high frequency data", Algorithmic Finance*, Vol. 2, No. 3-4, pp. 233-239*

Bishop. C. (2006), "Pattern recognition and machine learning", *New York: Springer-Verlag New York*

Brogaard, J.; Hendershott, T.; Riordan, R. (2014), "High-frequency trading and price discovery", Review of Financial Studies, Vol. 27, No. 8, pp. 2267-2306

Broomhead, D.s.; Lowe, D. (1988), "Multivariable functional interpolation and adaptive networks", *Complex Systems,* Vol. 2, No. 1, pp. 321-355

Chen, N.; Roll, R.; Ross, S. A. (1986), "Economic forces and the stock market", *The Journal of Business*, Vol. 59, No. 3, pp. 383-403

Chollet, F. (2015), "Keras", *GitHub*, available at: https://github.com/fchollet/keras (accessed 2016-10-27)

Chordia, T.; Roll, R; Subrahmanyam, A. (2001), Market Liquidity and Trading Activity. *The Journal of Finance*, Vol. 56, No. 2, pp. 501-530.

Cireçan, D. C.; Meier, U.; Gambardella, L. M.; Schmidhuber, J. (2010), "Deep, big, simple neural nets for handwritten digit recognition", *Neural Computation*, Vol. 22, No. 12, pp. 3207-3220

Clarke, J.; Jandik, T.; Mandelker, G. (2001), "The efficient markets hypothesis", *Expert Financial Planning: Advice from Industry Leaders*, pp. 126-141

Cootner, P. H. (1964), "The random character of stock market prices", *Cambridge, MA: M.I.T. Press*

Cybenko, G. (1989), "Approximations by superpositions of sigmoidal functions", *Mathematics of Control, Signals, and Systems*, Vol. 2, No. 4, pp. 303-314

Darrat, A. F.; Zhong, M. (2000), "On testing the random-walk hypothesis: A model-comparison approach", *The Financial Review,* Vol. 35, No. 1, pp. 105-124

Ding, X.; Zhang, Y.; Liu, T.; Duan, J. (2015), "Deep learning for event-driven stock prediction", *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 2327-2333

Dixon, M. F.; Klabjan, D.; Bang, J. H. (2016), "Classification-based financial markets prediction using deep neural networks", *working paper*

Doran, J. S.; Peterson, D. R.; Wright, C. (2010), "Confidence, opinions of market efficiency, and investment behavior of finance professors", *Journal of Financial Markets*, Vol. 13, No. 1, pp. 174-195

Drakos, K. (2004), "Terrorism-induced structural shifts in financial risk: Airline stocks in the aftermath of the September 11th terror attacks", *European Journal of Political Economy*, Vol. 20, No. 2, pp. 435-446

Elliott, D. L. (1993), "A better activation function for artificial neural networks", *technical report*

Fama, E. F. (1965), "The behaviour of stock-market prices", *Journal of Business*, Vol. 38, No. 1, pp. 34-105

Fame, E. F. (1970), "Efficient capital markets: A review of theory and empirical work", *The Journal of Finance*, Vol. 25, No. 2, pp. 383-417

Fama, E. F.; French, K. R. (2008), "Dissecting anomalies", *The Journal of Finance*, Vol. 63, No. 1, pp. 1653-1678

Fehrer, R.; Feuerriegel, S. (2015), "Improving decision analytics with deep learning: The case of financial disclosures", *working paper*

Gehrig, T.; Menkhoff, L. (2006), "Extended evidence on the use of technical analysis in foreign exchange", *International Journal of Finance & Economics*, Vol. 11, No. 1, pp. 327-338

Gibson, J.; Van Segbroeck, M.; Ortega, A.; Georgiou, P.; Narayanan, S. (2013), "Spectro-temporal directional derivative features for automatic speech recognition", *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH 2013)*, pp. 872-875

Giovanni, S.; Elder, J. F. (2010), "Ensemble methods in data mining: Improving accuracy through combining predictions", *San Rafael: Morgan & Claypool Publishers*

Glorot, X.; Bengio, Y. (2010), "Understanding the difficulty of training deep feedforward neural networks", *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 249-256

Górecki, T.; Łuczak, M. (2013), "Using derivatives in time series classification", *Data Mining and Knowledge Discovery*, Vol. 26, No. 2, pp. 310-331

Górecki, T.; Łuczak, M. (2014), "First and second derivatives in time series classification using DTW", *Communications in Statistics - Simulation and Computation,* Vol. 43, No. 9, pp. 2081-2092

He, K.; Zhang, X.; Ren, S.; Sun, J. (2015), "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification", *Proceedings of the 15th IEEE International Conference on Computer Vision (ICCV15)*

Heaton, J. B.; Polson, N. G.; Witte, J. H. (2016), "Deep learning for finance: deep portfolios", *Applied Stochastic Models in Business and Industry,* forthcoming, available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2838013

Hinton, G. E.; Salakhutdinov, R. R. (2006), "Reducing the dimensionality of data with neural networks", *Science*, Vol. 313, pp. 504-507

Ho, T. S. Y.; Stoll, H. R. (1983), "The Dynamics of Dealer Markets Under Competition", *The Journal of Finance*, Vol. 38, No. 4, pp. 1053-1074.

Hornik, K. (1991), "Approximation capabilities of multilayer feedforward networks", *Neural Networks*, Vol. 4, No. 2, pp. 251-257

Irie, B.; Miyake, S. (1988), "Capabilities of three-layer perceptrons", *Proceedings of the 2nd IEEE International Conference on Neural Networks*, pp. 641-648

Kamstra, M. J.; Kramer, L. A.; Levi, M. D.; Wermers, R. (2015), "Seasonal asset allocation: Evidence from mutual fund flows", *Journal of Financial and Quantitative Analysis*, forthcoming, available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1907904

Kendall, M. G.; Bradford Hill, A. (1953), "The analysis of economic time series part I: Prices", *Journal of the Royal Statistical Society*, Vol. 116, No. 1, pp. 11-34

Keogh, E. J.; Pazzani, M. J. (2001), "Derivative dynamic time warping", *Proceedings of the 1st SIAM International Conference on Data Mining (SIAM01),* pp. 285-289

Krauss, C., Do, X. A. & Huck, N. (2017) "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500", *European Journal of Operational Research,* Vol. 259, No. 2, pp. 689-702

Lavrenko, V.; Schmill, M.; Lawrie, D.; Ogilvie, P.; Jensen, D.; Allan, J. (2000), "Language models for financial news recommendation", *Proceedings of the 9th International Conference on Information and Knowledge Management,* pp. 389-396

Lee, D.; Zhang, S.; Fischer, A.; Bengio, Y. (2015), "Difference target propagation", *Lecture Notes in Artificial Intelligence: Machine Learning and Knowledge Discovery in Databases*, Vol. 9284, No. 1, pp. 498-515

Lo, A. W.; Mamaysky, H.; Wang, J. (2000), "Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation", *The Journal of Finance*, Vol. 55, No. 4, pp. 1705-1765

Malkiel, B. G.; Fama, E. F. (1970), "Efficient capital markets: A review of theory and empirical work", *The Journal of Finance*, Vol. 25, No. 2, pp. 383-417

Malkiel, B. G. (1973), "A random walk down wall street", *New York: W. W. Norton & Company, Inc.*

McGulloch, W.; Pitts, W. (1943), "A logical Calculus of ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics,* Vol. 5, No. 4, pp. 115-133

Mierswa, I. (2004), "Automatic feature extraction from large time series", *Proceedings of the 28th Annual Conference of the Gesellschaft für Klassifikation e.V.*, pp. 600-607

Nair, V.; Hinton, G. E. (2010), "Rectified linear units improve restricted Boltzmann machines", *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, paper ID: 432

Najafabadi, M. M.; Villanustre, F. V.; Khoshgoftaar, T. M; Seliya, N.; Wald, R.; Muharemagic, E. (2015), "Deep learning applications and challenges in big data analytics", *Journal of Big Data*, Vol. 2, No. 1, pp. 1-21

Neftci, S. N. (1991), "Naive trading rules in financial markets and Wiener-Kolmogorov Prediction Theory", *The Journal of Business*, Vol. 64, No. 4, pp. 549-571

Ng, A. (2012), "Machine learning and AI via brain simulations", *Lectures of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*

Nicholson, F. (1968), "Price ratios in relation to investment results", *Financial Analysts Journal*, Vol. 24, No. 1, pp. 105-109

Nielsen, M. A. (2015), "Neural networks and deep learning", *Determination Press*

Osler, C. L.; Chang, P. H. K. (1999), "Methodical madness: Technical analysis and the irrationality of exchange-rate forecasts", *Economic Journal*, Vol. 109, No. 458, pp. 636-661

Park, C. H,.; Irwin, S. H. (2004), "The profitability of technical analysis: A review", *AgMAS Project Research Reports*, No. 37487

Patell, J.; Wolfson, M. (1984), "The intraday speed of adjustment of stock prices to earnings and dividend announcements", *Journal of Financial Economics,* Vol. 13, pp. 223-252

Perryman, A. A.; Butler, F. C.; Martin, J. A.; Ferris, G. R. (2010), "When the CEO is ill: Keeping quiet or going public?", *Business Horizons*, Vol. 53, No. 1, pp. 21-29

R Core Team (2014), "R: A language and environment for statistical computing", *R Foundation for Statistical Computing*, available at: http://www.r-project.org/ (accessed 2016-10-19)

Rosenberg, B.; Reid, K.; Lanstein, R. (1985), "Persuasive evidence of market inefficiency", *The Journal of Portfolio Management*, Vol. 11, No. 1, pp. 9-16

Rosenblatt, F. (1958), "The perceptron: A probabilistic model for information storage and organization in the brain", *Psychological Review*, Vol. 65, No. 6, pp. 386–408

RStudio Team (2015), "RStudio: Integrated development for R", *RStudio Inc.*, available at: http://www.rstudio.com/ (accessed 2016-10-19)

Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. (1986), "Learning representations by back-propagating errors", *Nature,* Vol. 323, No. 9, pp. 533-536

Saad, E. W. (1998), "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks", *IEEE Transactions on Neural Networks*, Vol. 9, No. 1, pp. 1456-1470

Schumaker, R. P.; Chen, H. (2009a), "Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFinText System", *ACM Transactions on Information Systems*, Vol. 27, pp. 1-19

Schumaker, R. P.; Chen, H. (2009b), "A quantitative stock prediction system based on financial news", *Information Processing and Management*, Vol. 45, pp. 571-583

Sitte, R.; Sitte, J. (2002), "Neural networks approach to the random walk dilemma of financial time series", *Applied Intelligence*, Vol. 16, No. 3, pp. 163-171

Skabar, A., Cloete, I. (2002), "Neural networks, financial trading and the efficient markets hypothesis", *Proceedings of the 24th Australasian Conference on Computer Science*, Vol. 4, pp. 241-249

Snoek, J.; Larochelle, H.; Adams, R. P. (2012), "Practical Bayesian optimization of machine learning algorithms", *Advances in Neural Information Processing Systems 25,* pp. 2951-2959

Takeuchi, L.; Lee, Y. (2013), "Applying deep learning to enhance momentum trading strategies in stocks", *working paper*

Tippmann, S. (2015), "Programming tools: Adventures with R", *Nature*, Vol. 517, No. 1, pp. 109-110

van Rossum, G. (1995), "Python tutorial", *technical report,* CWI report CS-R9526

Wasserstein, R. L.; Lazar, N. A. (2016), "The ASA's statement on p-values: Context, process, and purpose", *The American Statistician*, Vol. 70, No. 2, pp. 129-133

Werbos, P. J. (1974), "Beyond regression: New tools for prediction and analysis in the behavioural sciences", *PhD thesis, Harvard University*

White, H. (1988), "Economic prediction using neural networks: The case of IBM daily stock returns", *Proceedings of the IEEE International Conference on Neural Networks*, pp. 451-459

Wlodarczak, P.; Soar, J.; Ally, M. (2015), "Multimedia data mining using deep learning", *Proceedings of the 5th Int. Conference on Digital Information Processing and Communications*, pp. 190-196

Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. (2013), "On the importance of initialization and momentum in deep learning", *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pp. 1139-1147

Zhang, G.; Patuwo, B. E.; Hu, M. J. (1997), "Forecasting with artificial neural networks: The state of the art", *International Journal of Forecasting,* Vol. 14, No. 1, pp. 35-62

Figure 1: Feedforward neural network with no hidden layers

The depicted model features four nodes in the input layer, i.e. four input variables, and one node in the output layer. Subject to the weights of the connections between the nodes, this model therefore implements a classic linear regression.

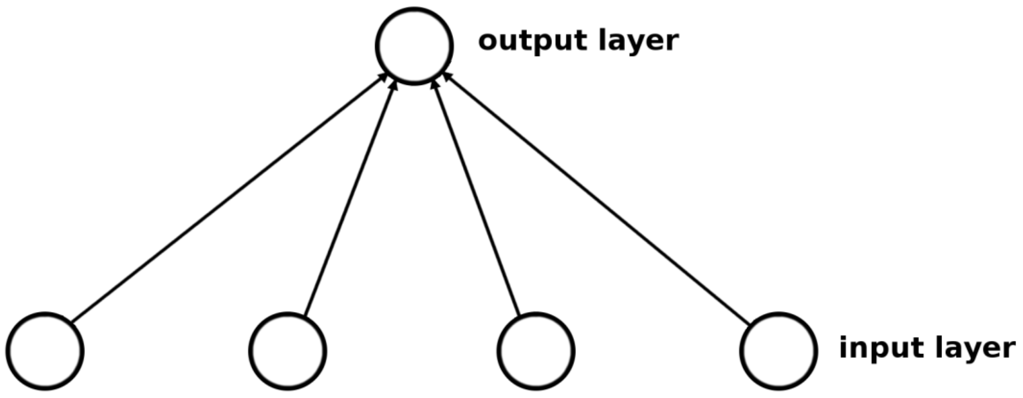


Figure 2: The sigmoid function

The non-linear nature of the sigmoid that allows for a more differentiated output than, e.g. a binary Heaviside step function, as well as the sufficient approximation of biological neuron's firing behaviour, are primary reasons to use it for neural network models.

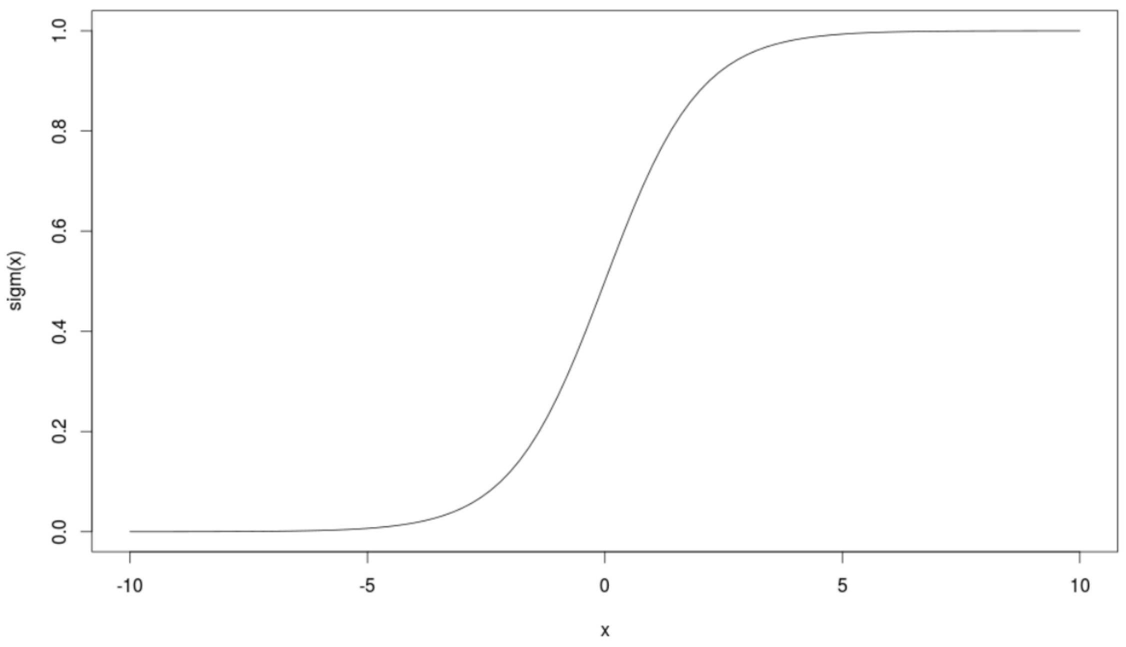


Figure 3: Feedforward neural network with one hidden layer

As opposed to Figure 2, which also shows four input nodes, the depicted model features an additional hidden layer with three nodes for a more complex information extraction, and two output nodes, e.g. for a binary classification task.

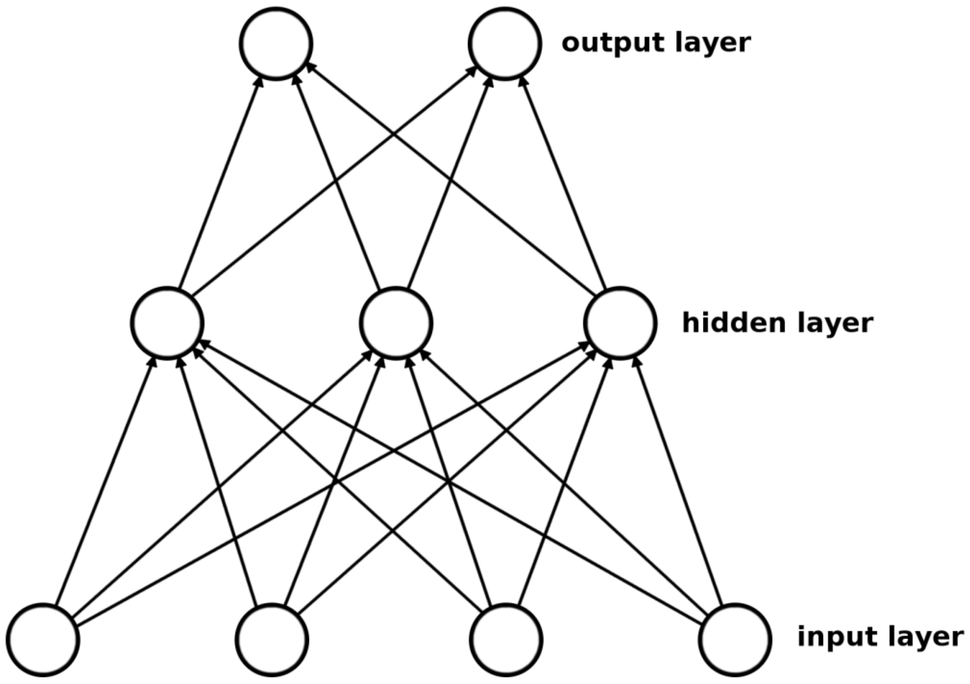


Figure 4: Head-and-shoulders pattern in stock market data

Head-and-shoulders pattern in stock market data is not a mathematically developed decision aid. However, the pattern is widely used by technical analysts as a guideline for investments, as it is perceived to follow a predictable scheme as a rule-of-thumb within the stock market.

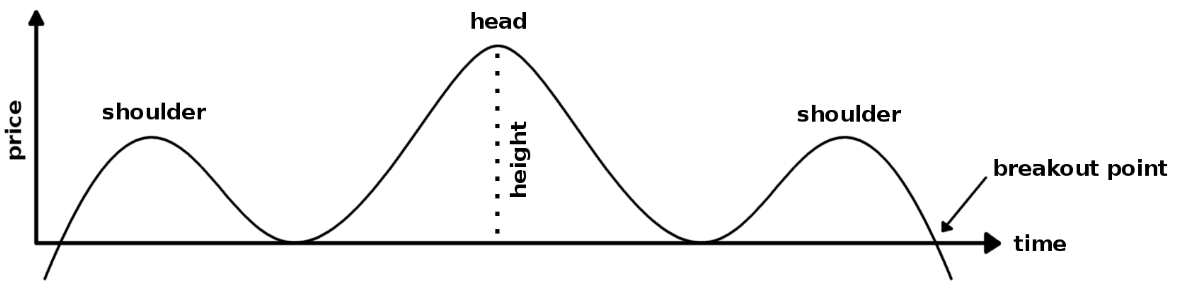


Figure 5: Model setup for the experiments

A simple linear regression is performed on each time step for each stock's price series, and the first derivative of the resulting regression is then taken as a trend strength indicator. For a respective stock, all stock's gradients for the preceding time step except for information about the target stock are then used as input features for a feed-forward artificial neural network with 5 hidden layers and two output nodes for a binary classification task. The model is then, with separate experiments for each stock, trained to predict the up- or downward change of the target stock's trend gradient in the next time step.

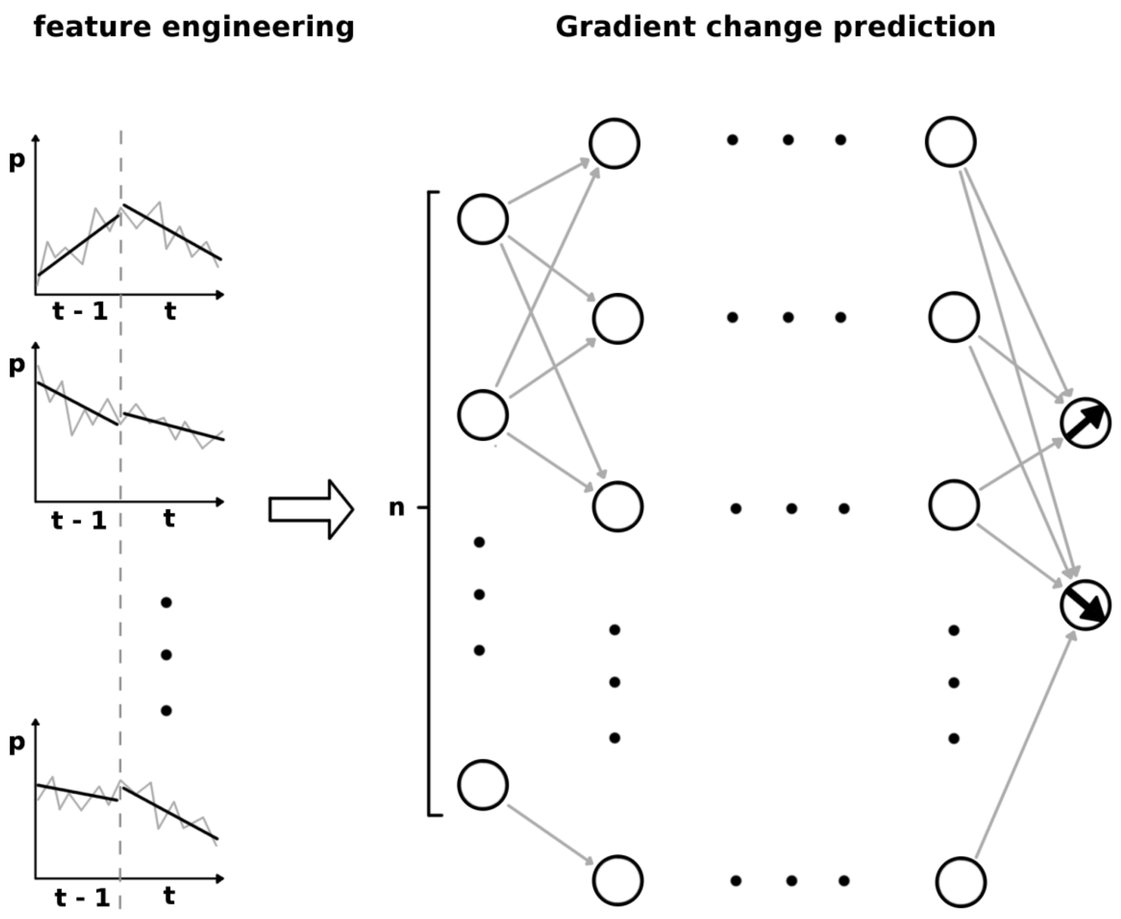


Figure 6: Model with a one-neuron bottleneck layer

The depicted model inserts a one-node bottleneck in a deep feed-forward neural network model for binary classification. The reason for this insertion is to force the model to learn a goal-oriented reduction of the model's information by "squeezing" the necessary information through a narrow representation mid-way.

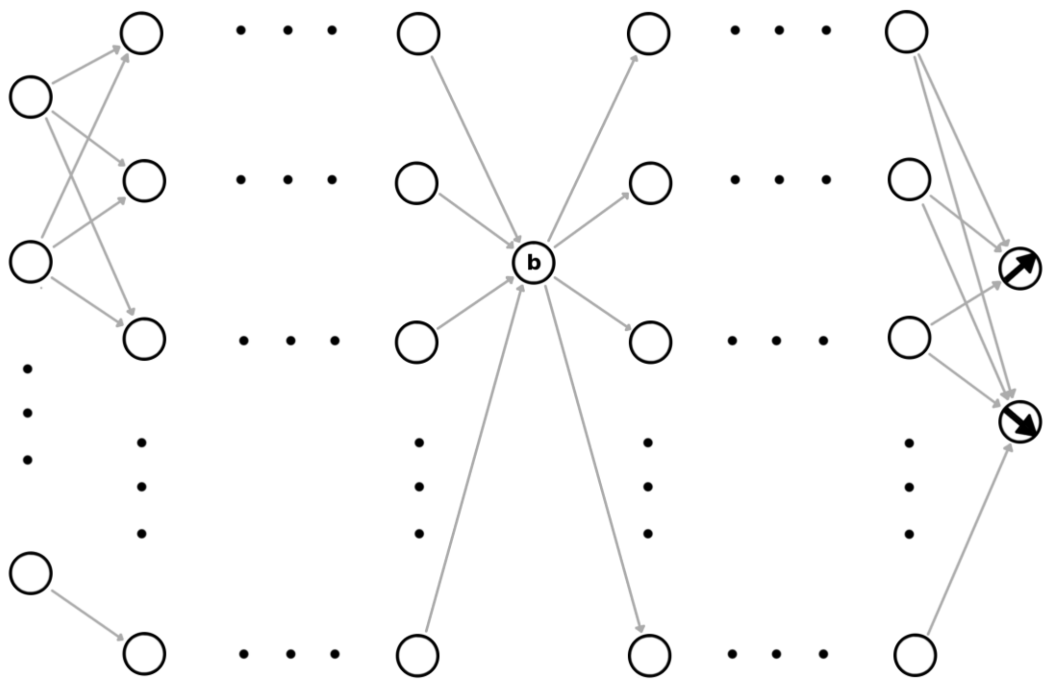


Figure 7: Notched box-and-whiskers plots for accuracies over time intervals

The Figure shows the notched box-and-whiskers plots for accuracies of various time intervals. Panels A, B and C presents the plots for one day, one hour and half-hour intervals respectively. A statistically significant difference in means at 95% confidence is indicated if notches are non-overlapping. The lower and upper ends of a box indicate the first and third quartile, while the median is depicted as a horizontal bar. The whiskers show the lowest and highest data point that is within 1.5-times the interquartile range of the first and third quartile:

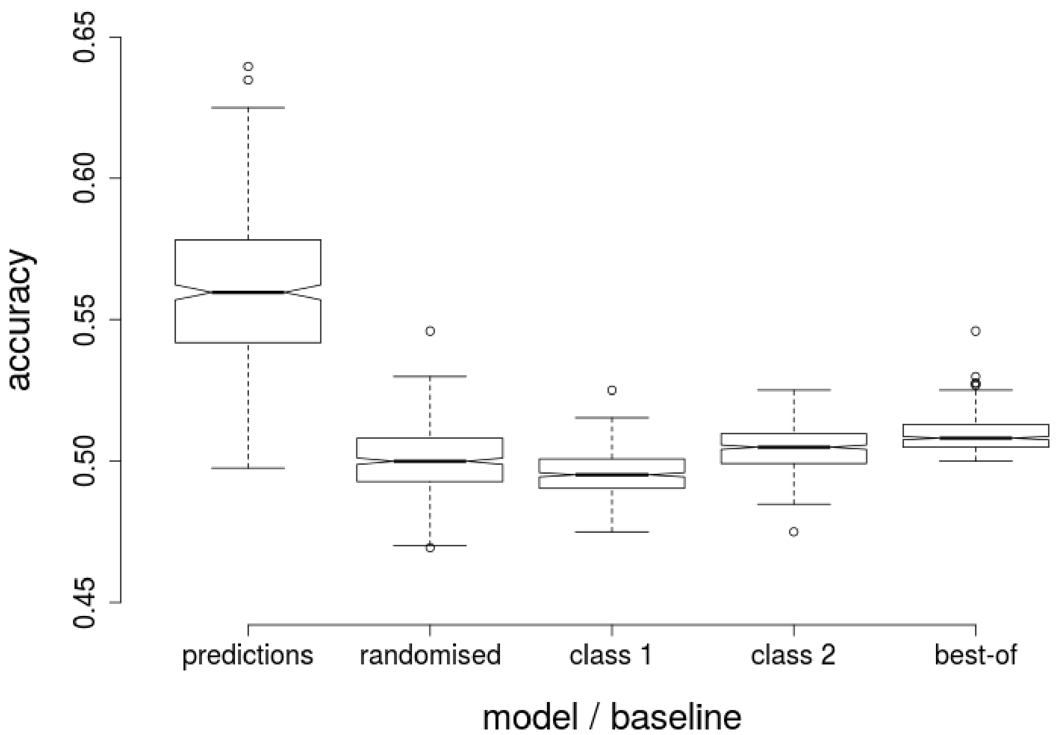
〖whisker〗 _upper = min(max(data) ,Q_3+1.5·(Q_1-Q_3))

〖whisker〗 _lower = max(min(data) ,Q_1-1.5·(Q_1-Q_3))

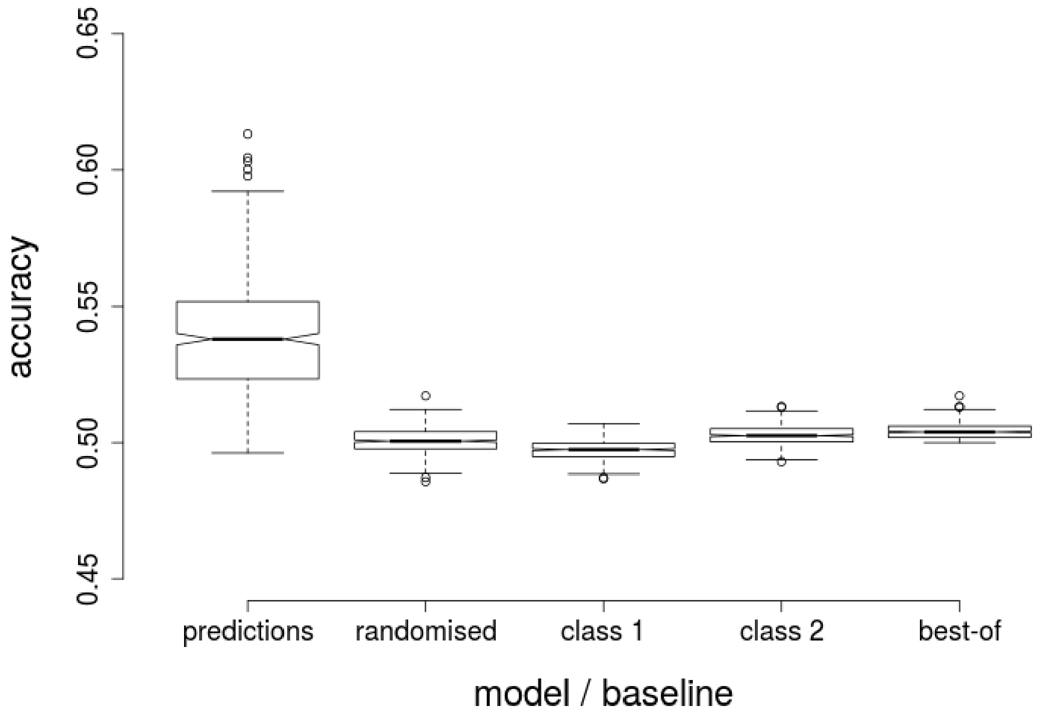
Outliers are shown above or below the whiskers, and non-overlapping notches for two boxes indicate a statistically significant median difference at 95% confidence. Welch's t-test is used to achieve a higher reliability for unequal variances.

Here, class 1 is the prediction that stock trends will change downwards, and class 2 is the prediction that stock trends will change upwards, whereas randomised represents the same prediction distribution as the model's predictions sampled randomly in order to test for simple distribution learning. The best-of features the respective highest accuracy of the three baselines, i.e. class 1, class 2 and randomised to result in a fourth baseline that is, by default, at least 50%.

Panel A



Panel B



Panel C

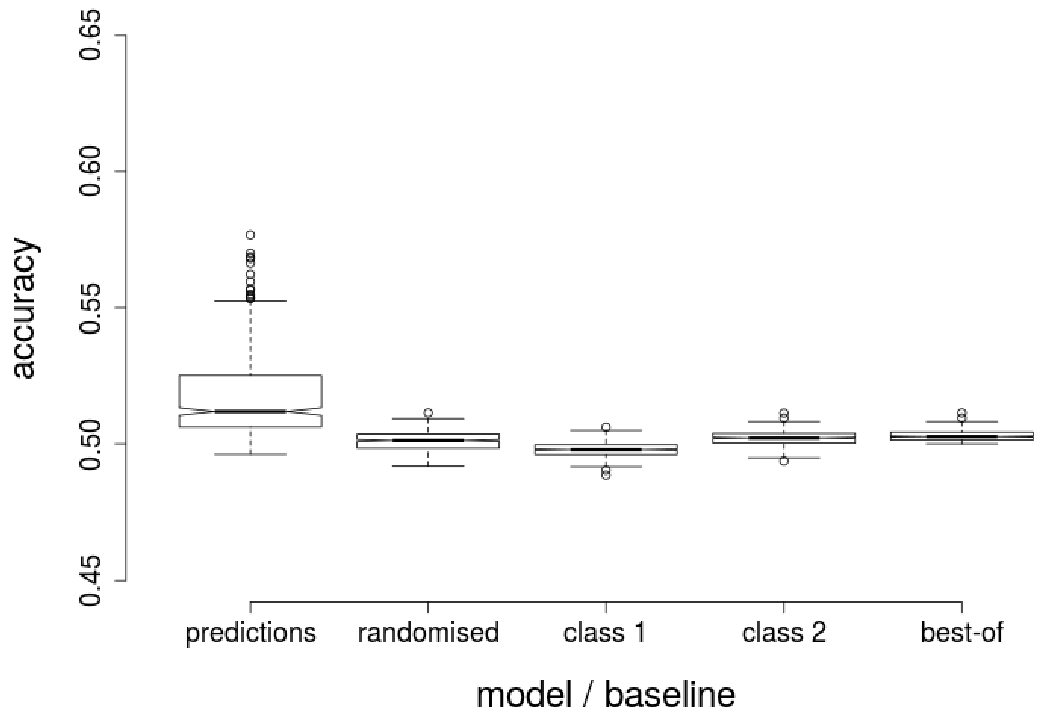

accuracy
0.45
0.50
0.55
0.60
0.65
predictions
randomised
class 1
class 2
best-of
model / baseline

Figure 8: Notched box-and-whiskers plots for accuracies for different bottleneck sizes within a feed-forward neural network model

A statistically significant difference in means at 95% confidence is indicated if notches are non-overlapping. As can be seen from the figure, the accuracy rises significantly with each of the increments in the respective bottleneck's size, with a reduction of the model's information to 10 variables nearly reaching the accuracy of a model without a bottleneck.

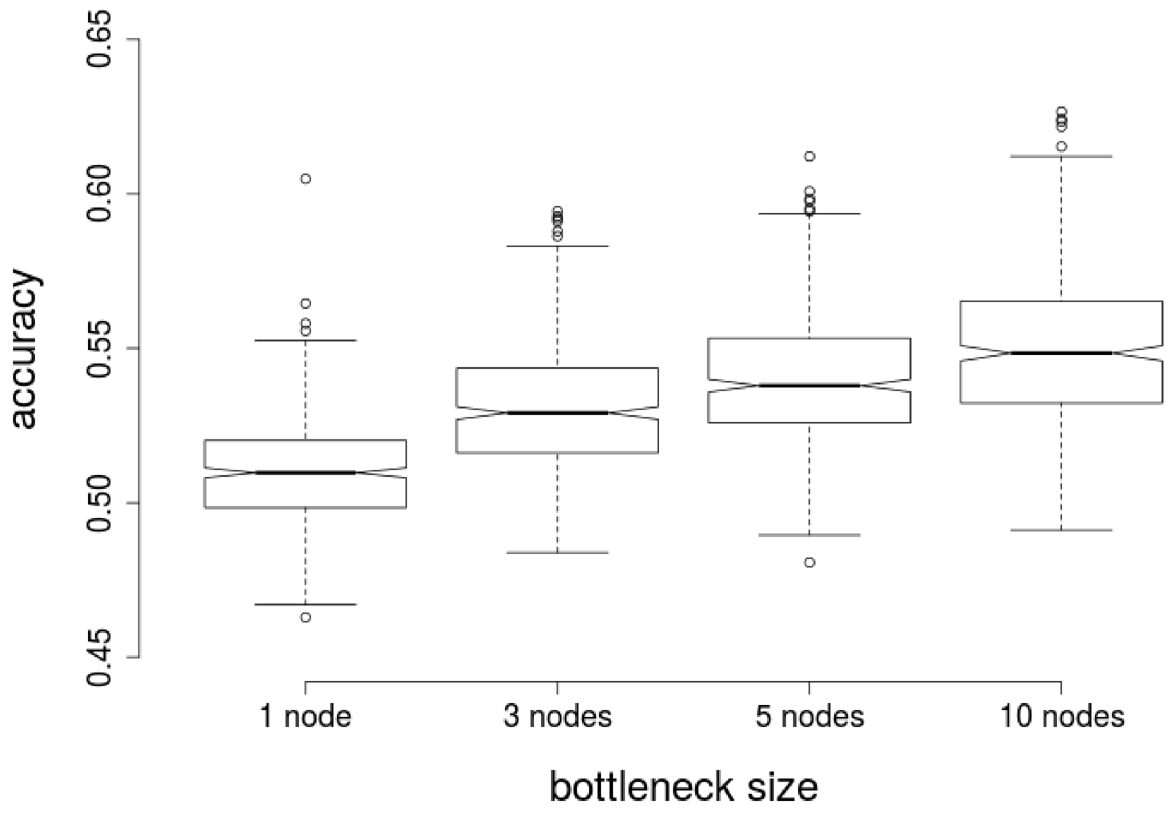


Figure 9: Notched box-and-whiskers plots for accuracies in a high-volatility scenario

A statistically significant difference in means at 95% confidence is indicated if notches are non-overlapping. Here, class 1 is the prediction that stock trends will change downwards, and class 2 is the prediction that stock trends will change upwards, whereas randomised represents the same prediction distribution as the model's predictions sampled randomly in order to test for simple distribution learning. The best-of features the respective highest accuracy of the three baselines, i.e. class 1, class 2 and randomised to result in a fourth baseline that is, by default, at least 50%.

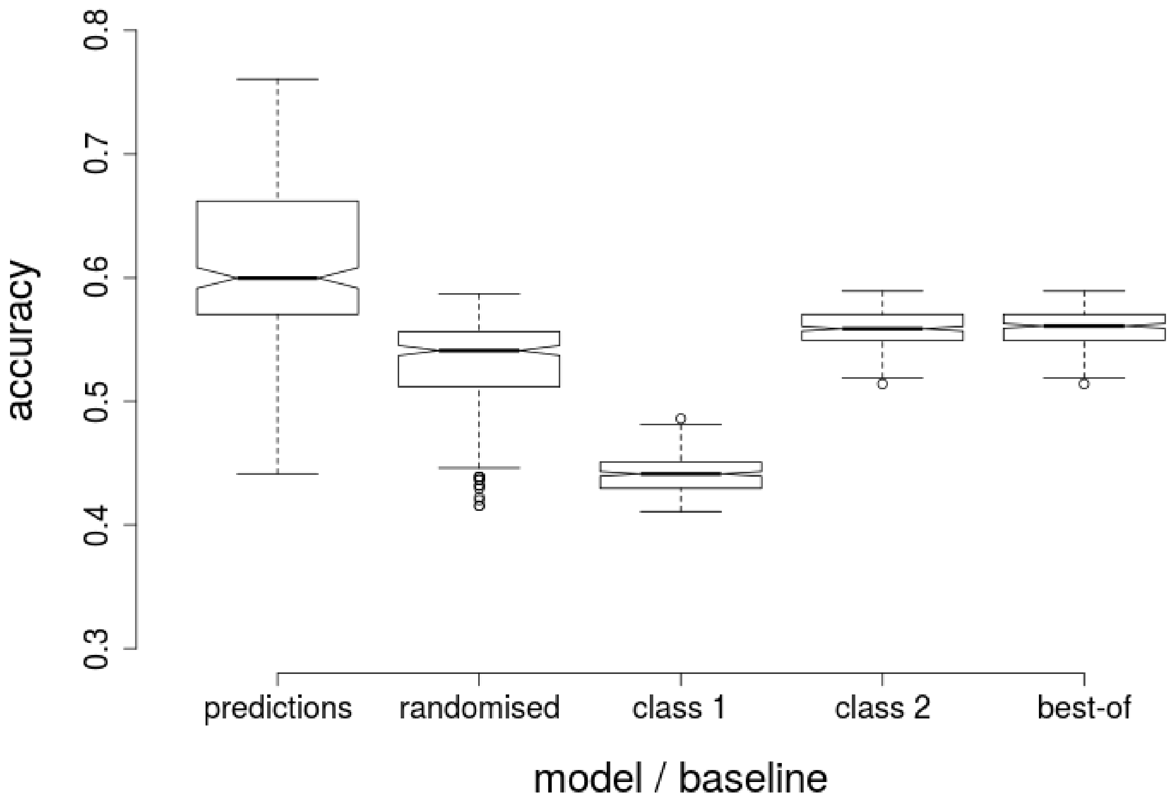


Table 1: Statistical KPIs for various time intervals

449 S&P 500 stocks are used to predict directional trend changes for each stock, with only the respective other 448 stocks' trends from the preceding time interval as input features. In the table, accuracies, p-values and the minimal difference w.r.t. the significant difference in means are given. Class 1 is the prediction that stock trends will change downwards, and class 2 is the prediction that stock trends will change upwards, whereas randomised represents the same prediction distribution as the model's predictions sampled randomly to test for simple distribution learning. The best-of features the respective highest accuracy of the three baselines, i.e. class 1, class 2 and randomised to result in a fourth baseline that is, by default, at least 50%. Panels A, B and C presents the estimates for one day, one hour and half-hour intervals respectively.

Panel A

accuracies of predictions				
model	randomised	class 1	class 2	best-of
~ 0.5602	~ 0.5002	~0 .4955	~ 0.5045	~ 0.5092
tests against baselines				
	randomised	class 1	class 2	best-of
p-value	p < 0.001	p < 0.001	p < 0.001	p < 0.001
min. diff.	~ 0.0599	~ 0.0607	~ 0.0518	~ 0.0471

Panel B

accuracies of predictions				
model	randomised	class 1	class 2	best-of
~ 0.5395	~ 0.5008	~ 0.4973	~ 0.5027	~0.5043
tests against baselines				
	randomised	class 1	class 2	best-of
p-value	p < 0.001	p < 0.001	p < 0.001	p < 0.001
min. diff.	~ 0.0356	~ 0.0392	~ 0.0338	~ 0.0322

Panel C

accuracies of predictions				
model	randomised	class 1	class 2	best-of
~ 0.5170	~ 0.5011	~ 0.4979	~ 0.5021	~0.5030

tests against baselines				
	randomised	class 1	class 2	best-of
p-value	p < 0.001	p < 0.001	p < 0.001	p < 0.001
min. diff.	~ 0.0137	~ 0.0169	~ 0.0127	~ 0.0118

Table 2: Statistical KPIs for different bottleneck sizes

449 S&P 500 stocks were used to predict directional trend changes for each stock, with only the respective other 448 stocks' trends from the preceding time interval as input features. The accuracy rises with each of the increments in the respective bottleneck's size, with a reduction of the model's information to 10 variables nearly reaching the accuracy of a model without a bottleneck, i.e. no forced compression.

accuracies of predictions				
1 node	3 nodes	5 nodes	10 nodes	no bottleneck
~ 0.5107	~ 0.5309	~ 0.5395	~ 0.5503	~0.5602

Table 3: Statistical KPIs for high-volatility environments

449 of the S&P 500 stocks were used to predict directional trend changes for each stock, with only the respective other 448 stocks' trends from the preceding time interval as input features. In the table, accuracies, p-values and the minimal difference w.r.t. the significant difference in means are given. Class 1 is the prediction that stock trends will change downwards, and class 2 is the prediction that stock trends will change upwards, whereas randomised represents the same prediction distribution as the model's predictions sampled randomly to test for simple distribution learning. The best-of features the respective highest accuracy of the three baselines, i.e. class 1, class 2 and randomised to result in a fourth baseline that is, by default, at least 50%.

accuracies of predictions				
model	randomised	class 1	class 2	best-of
~ 0.6113	~ 0.5301	~ 0.4405	~ 0.5595	~0.5607
tests against baselines				
	randomised	class 1	class 2	best-of
p-value	$p < 0.001$	$p < 0.001$	$p < 0.001$	$p < 0.001$
min. diff.	~ 0.0677	~ 0.1589	~ 0.0399	~ 0.0388

Appendix A

Main function for the substitution of missing instances:

```
# Function to account for missing time stamps
missinginstances <- function(data, stocks.num, valid.times, ) {
  data.list <- list();
  for(i in 1:stocks.num) {
    data.list[[i]] <- data[(data$X.RIC == unique.ric[i]), ];
  }
  count.list <- list();
  for(i in 1:stocks.num) {
    count.list[[i]] <- data.list[[i]]$Time.L.[1];
  }
  count.wrong <- unlist(count.list);
  count.wrong <- which(count.wrong != valid.times[1]);
  for(i in count.wrong) {
    data.start <-
        as.integer(substring(data.list[[i]]$Time.L.[1], 1, 2));
    valid.start <- as.integer(substring(valid.times[1], 1, 2));
    start.diff <- data.start - valid.start;
    data.list[[i]][(1 + start.diff):(dim(data.list[[i]])[1]
        + start.diff), ]
        <- data.list[[i]][1:dim(data.list[[i]])[1], ];
    for(j in 1:start.diff) {
      data.list[[i]][j, ] <- data.list[[i]][(1 + start.diff), ]
      data.list[[i]]$Time.L.[j] <- valid.times[j];
    }
  }
  for(i in 1:stocks.num) {
    data.list[[i]] <- fillinstances(data.list[[i]], valid.times);
    data.list[[i]] <- na.omit(data.list[[i]]);
  }
  return(data.list);
}
```

Auxiliary function for missinginstances():

```
# Function to detect and fill missing instances
fillinstances <- function(data, valid.times) {
  ruler <- dim(data)[1];
  input.length <- ruler;
  data <- rbind(data, data);
  data.length <- dim(data)[1];
  data$Type[(input.length + 1):data.length] <- NA;
  times <- data$Time.L.;
  times.new <- rep(valid.times, length.out=input.length);
  id.times <- FALSE;
  while(!(id.times == TRUE)) {
    loc.diff <- min(which((times == times.new) == FALSE));
    if((loc.diff == Inf) || is.na(data$Type[loc.diff])) {
      id.times <- TRUE;
    } else {
      data[(loc.diff + 1):(ruler + 1), ] <- data[loc.diff:ruler, ];
      hold.time <- times.new[loc.diff];
      if(data$X.RIC[loc.diff] == data$X.RIC[loc.diff - 1]) {
        data[loc.diff, ] <- data[loc.diff - 1, ];
      } else {
        data[loc.diff, ] <- data[loc.diff + 1, ];
      }
      data$Time.L.[loc.diff] <- hold.time;
      ruler <- ruler + 1;
      times <- data$Time.L.;
    }
  }
  data <- data[1:(min(which(is.na(data$Type))) - 1), ];
  data <- data[, !(names(data) %in% 'Type')];
  return(data);
}
```

<u>Input explanation:</u>
Data is a matrix without *NA* values containing a dataset with a structure as described in Section 3.1.2. stocks.num is the number of unique stocks represented in data, and valid.times is a vector of valid successive time stamps of the same length as the number of rows in data. The output of missinginstances() is a list with one list place for each unique stock in data that contain matrices that can be aligned perfectly.

Appendix B

Reuters Instrument Codes for the primary and bottleneck experiments:

AA.N	AAPL.OQ	AAP.N	ABC.N	ABT.N	ACN.N
ADBE.OQ	ADM.N	ADP.OQ	ADSK.OQ	ADS.N	AEE.N
AEP.N	AES.N	AET.N	AFL.N	AIG.N	AIV.N
AIZ.N	AKAM.OQ	ALL.N	ALXN.OQ	AMAT.OQ	AME.N
AMG.N	AMGN.OQ	AMP.N	AMT.N	AMZN.OQ	A.N
AN.N	AON.N	APA.N	APC.N	APD.N	APH.N
ARG.N	ATVI.OQ	AVB.N	AVGO.OQ	AVY.N	AWK.N
AXP.N	AZO.N	BAC.N	BA.N	BAX.N	BBBY.OQ
BBT.N	BBY.N	BCR.N	BDX.N	BEN.N	Bfb.N
BHI.N	BIIB.OQ	BK.N	BLK.N	BLL.N	BMY.N
BRKb.N	BSX.N	BWA.N	BXP.N	CAG.N	CAH.N
CA.OQ	CAT.N	CBG.N	CB.N	CBS.N	CCE.N
CCI.N	CCL.N	CELG.OQ	CERN.OQ	CF.N	CHD.N
CHK.N	CHRW.OQ	CI.N	CINF.OQ	CL.N	CLX.N
CMA.N	CMCSA.OQ	CME.OQ	CMG.N	CMI.N	CMS.N
C.N	CNC.N	CNP.N	COF.N	COG.N	COH.N
COL.N	COP.N	COST.OQ	CPB.N	CRM.N	CSCO.OQ
CTAS.OQ	CTL.N	CTSH.OQ	CTXS.OQ	CVC.N	CVS.N
CVX.N	CXO.N	DAL.N	DD.N	DE.N	DFS.N
DG.N	DGX.N	DHI.N	DHR.N	DISCA.OQ	DISCK.OQ
DIS.N	DLTR.OQ	D.N	DNB.N	DO.N	DOV.N
DOW.N	DPS.N	DRI.N	DTE.N	DUK.N	DVA.N
DVN.N	EBAY.OQ	ECL.N	ED.N	EFX.N	EIX.N
EL.N	EMC.N	EMN.N	EMR.N	ENDP.OQ	EOG.N
EQIX.OQ	EQR.N	EQT.N	ESRX.OQ	ESS.N	ETFC.OQ
ETN.N	ETR.N	EW.N	EXC.N	EXPD.OQ	EXPE.OQ
EXR.N	FAST.OQ	FCX.N	FDX.N	FE.N	FFIV.OQ
FIS.N	FISV.OQ	FITB.OQ	FLIR.OQ	FL.N	FLR.N
FLS.N	FMC.N	F.N	FRT.N	FSLR.OQ	FTI.N
GD.N	GE.N	GGP.N	GILD.OQ	GIS.N	GLW.N
GME.N	GM.N	GOOG.OQ	GPC.N	GPN.N	GPS.N
GRMN.OQ	GS.N	GWW.N	HAL.N	HAR.N	HAS.OQ
HBAN.OQ	HBI.N	HCA.N	HCN.N	HCP.N	HD.N
HES.N	HIG.N	HOG.N	HOLX.OQ	HON.N	HOT.N

HP.N	HPQ.N	HRB.N	HRL.N	HRS.N	HSIC.OQ
HST.N	HSY.N	HUM.N	IBM.N	ICE.N	IFF.N
ILMN.OQ	INTC.OQ	INTU.OQ	IPG.N	IP.N	IRM.N
IR.N	ISRG.OQ	ITW.N	IVZ.N	JBHT.OQ	JCI.N
JEC.N	JNJ.N	JNPR.N	JPM.N	JWN.N	KEY.N
KIM.N	KLAC.OQ	KMB.N	KMI.N	KMX.N	K.N
KO.N	KR.N	KSS.N	KSU.N	LEG.N	LEN.N
LH.N	LLL.N	LLTC.OQ	LLY.N	LM.N	LMT.N
L.N	LNC.N	LOW.N	LRCX.OQ	LUK.N	LUV.N
LYB.N	MAC.N	MA.N	MAS.N	MAT.OQ	MCD.N
MCHP.OQ	MCK.N	MCO.N	MDT.N	MET.N	MHK.N
MJN.N	MKC.N	MLM.N	MMC.N	MMM.N	M.N
MO.N	MON.N	MOS.N	MRK.N	MRO.N	MSFT.OQ
MSI.N	MS.N	MTB.N	MU.OQ	MUR.N	MYL.OQ
NBL.N	NDAQ.OQ	NEE.N	NEM.N	NFLX.OQ	NFX.N
NI.N	NKE.N	NLSN.N	NOC.N	NOV.N	NRG.N
NSC.N	NTAP.OQ	NTRS.OQ	NUE.N	NVDA.OQ	NWL.N
NWSA.OQ	NWS.OQ	OI.N	OKE.N	OMC.N	O.N
ORLY.OQ	OXY.N	PAYX.OQ	PBCT.OQ	PBI.N	PCAR.OQ
PCG.N	PCLN.OQ	PDCO.OQ	PEG.N	PEP.N	PFE.N
PFG.N	PG.N	PHM.N	PH.N	PKI.N	PLD.N
PM.N	PNC.N	PNR.N	PNW.N	PPG.N	PPL.N
PRU.N	PSA.N	PVH.N	PWR.N	PXD.N	PX.N
QCOM.OQ	RAI.N	RCL.N	RF.N	RHI.N	RHT.N
RIG.N	RL.N	R.N	ROK.N	ROP.N	ROST.OQ
RRC.N	RSG.N	RTN.N	SBUX.OQ	SCG.N	SCHW.N
SEE.N	SE.N	SHW.N	SIG.N	SJM.N	SLB.N
SLG.N	SNA.N	SNDK.OQ	SO.N	SPG.N	SPLS.OQ
SRCL.OQ	SRE.N	STI.N	STJ.N	STT.N	STX.OQ
STZ.N	SWK.N	SWKS.OQ	SWN.N	SYK.N	SYMC.OQ
SYY.N	TAP.N	TDC.N	TEL.N	TE.N	TGT.N
TIF.N	TJX.N	TMK.N	TMO.N	T.N	TROW.OQ
TRV.N	TSCO.OQ	TSN.N	TSO.N	TSS.N	TWC.N
TWX.N	TXT.N	TYC.N	UAL.N	UA.N	UDR.N
UHS.N	ULTA.OQ	UNH.N	UNM.N	UNP.N	UPS.N
URBN.OQ	URI.N	USB.N	UTX.N	VAR.N	VFC.N

VLO.N	VMC.N	V.N	VNO.N	VRSK.OQ	VRSN.OQ
VRTX.OQ	VTR.N	VZ.N	WAT.N	WEC.N	WFC.N
WHR.N	WMB.N	WM.N	WMT.N	WU.N	WY.N
WYN.N	WYNN.OQ	XEC.N	XEL.N	XL.N	XOM.N
XRAY.OQ	XRX.N	YHOO.OQ	YUM.N	ZION.OQ	

Reuters Instrument Codes for the high-volatility experiments:

AA.N	AAP.N	ABC.N	ABT.N	ACN.N	ADM.N
ADS.N	AEE.N	AEP.N	AES.N	AET.N	AFL.N
AGN.N	AIG.N	AIV.N	ALL.N	AME.N	AMG.N
AMT.N	A.N	AN.N	APA.N	APC.N	APD.N
APH.N	ARG.N	AVB.N	AVY.N	AXP.N	AZO.N
BAC.N	BA.N	BAX.N	BBT.N	BBY.N	BCR.N
BDX.N	BEN.N	Bfb.N	BHI.N	BK.N	BLK.N
BLL.N	BMY.N	BRKb.N	BSX.N	BWA.N	BXP.N
CAG.N	CAH.N	CAT.N	CB.N	CCE.N	CCI.N
CCL.N	CHD.N	CHK.N	CI.N	CL.N	CLX.N
CMA.N	CMS.N	C.N	CNP.N	COF.N	COG.N
COH.N	COL.N	COP.N	CPB.N	CTL.N	CVC.N
CVS.N	CVX.N	DD.N	DE.N	DGX.N	DHI.N
DHR.N	DIS.N	D.N	DNB.N	DO.N	DOV.N
DOW.N	DRI.N	DTE.N	DUK.N	DVA.N	ECL.N
ED.N	EFX.N	EIX.N	EL.N	EMC.N	EMN.N
EMR.N	EOG.N	EQR.N	EQT.N	ESS.N	ETN.N
ETR.N	EW.N	EXC.N	FCX.N	FDX.N	FE.N
FLR.N	FLS.N	FMC.N	F.N	FRT.N	FTI.N
GAS.N	GD.N	GE.N	GIS.N	GLW.N	GME.N
GM.N	GPC.N	GPN.N	GPS.N	GS.N	GWW.N
HAL.N	HAR.N	HCN.N	HCP.N	HD.N	HIG.N
HON.N	HOT.N	HP.N	HPQ.N	HRB.N	HRL.N
HRS.N	HSY.N	HUM.N	IBM.N	IFF.N	IPG.N
IP.N	IRM.N	IR.N	ITW.N	JCI.N	JEC.N
JNJ.N	JPM.N	JWN.N	KEY.N	KIM.N	KMB.N
KMX.N	K.N	KO.N	KR.N	KSS.N	KSU.N

LEG.N	LEN.N	LH.N	LLL.N	LLY.N	LM.N
LMT.N	LNC.N	LOW.N	LUK.N	LUV.N	MAC.N
MAS.N	MCD.N	MCK.N	MCO.N	MDT.N	MET.N
MHK.N	MKC.N	MLM.N	MMC.N	MMM.N	MO.N
MON.N	MRK.N	MRO.N	MTB.N	MUR.N	NBL.N
NEM.N	NFX.N	NI.N	NKE.N	NOC.N	NSC.N
NUE.N	NWL.N	OI.N	OKE.N	OMC.N	O.N
OXY.N	PBI.N	PCG.N	PEG.N	PEP.N	PFE.N
PFG.N	PG.N	PGR.N	PHM.N	PH.N	PKI.N
PLD.N	PNC.N	PNR.N	PNW.N	PPG.N	PPL.N
PRU.N	PSA.N	PVH.N	PWR.N	PXD.N	PX.N
RCL.N	RF.N	RHI.N	RIG.N	RL.N	R.N
ROK.N	ROP.N	RRC.N	RSG.N	RTN.N	SCG.N
SEE.N	SHW.N	SJM.N	SLB.N	SLG.N	SNA.N
SO.N	SPG.N	SRE.N	STI.N	STJ.N	STT.N