Convolutional Neural Networks Applied to High-Frequency Market Microstructure Forecasting

Jonathan Doering
Centre for Computational Finance
and Economic Agents
University of Essex
Wivenhoe Park
Colchester CO4 3SQ
Email: mail@jonathan-doering.com

Michael Fairbank
School of Computer Science
University of Essex
Wivenhoe Park
Colchester CO4 3SQ
Email: m.fairbank@essex.ac.uk

Sheri Markose
Economics Department
University of Essex
Wivenhoe Park
Colchester CO4 3SQ
Email: scher@essex.ac.uk

***Abstract*—Highly sophisticated artificial neural networks have achieved unprecedented performance across a variety of complex real-world problems over the past years, driven by the ability to detect significant patterns autonomously. Modern electronic stock markets produce large volumes of data, which are very suitable for use with these algorithms. This research explores new scientific ground by designing and evaluating a convolutional neural network in predicting future financial outcomes. A visually inspired transformation process translates high-frequency market microstructure data from the London Stock Exchange into four market-event based input channels, which are used to train six deep networks. Primary results indicate that convolutional networks behave reasonably well on this task and extract interesting microstructure patterns, which are in line with previous theoretical findings. Furthermore, it demonstrates a new approach using modern deep-learning techniques for exploiting and analysing market microstructure behaviour.**

***Index Terms*—Deep Learning, Finance, Limit Order Book**

I. Introduction

Modern financial markets are highly technology driven industries, generating a plethora of data within very short time frames. The interaction of multiple market participants in electronic limit order book exchanges, creates rich granular datasets. These are challenging to analyse and study, due to their size and scatteredness. However, limit order book data are known as the crudest and utterly raw form of market information containing all buy and sell information of the market. It is known that markets are inefficient at these levels of granularity. As a result, it is assumed that limit order book data are a promising source for predicting short-term price moves, even though they are highly noisy and hence extremely difficult to analyse.

Due to a dramatic increase in computational power, larger structured data sources, and improvements in the learning algorithms, a renaissance of neural networks in form of so-called deep-learning algorithms occurred within the last few years [1]. These powerful machine-learning techniques have been proven to be successful mainly in areas of computer vision and language processing, however, are meanwhile tested and applied on diversified fields in computer science, especially where large labelled datasets exist. Moreover, it turned out that deep-learning algorithms are capable of dealing with noise in data reasonably well.

Diverse studies already showed evidence of valuable information included in microstructure data. Malik and Markose [2] developed a method to determine mid-term, intraday price trends based on extracted evolutions of demand and supply price curves from full-depth limit order book snapshots. Lipinski and Brabazon [3] concluded, after investigating thousands of order-book snapshots, that order book shape patterns may contain valuable information for automatic trading. Furthermore, Tóth et al. [4] found significant differences in execution queue length as well as order and deletion occurrence, around large price jumps. The fact that the vast majority of all orders are cancelled nowadays leads to the assumption that order cancellations play a crucial role in price finding mechanism as well.

Although machine learning has been extensively studied in the domain of financial forecasting, it was rarely applied to market microstructure data so far. Cao et al. [5] used a k-nearest neighbour clustering to track down price manipulations in high-frequency datasets. Already mentioned Lipinski and Brabazon [3] used self-organising maps to get their results. Also, reinforcement learning showed acceptable success in trade execution optimisation [6] and trading [7], based on order-book states. Both publications by Kearns and Nevmyvaka used handcrafted features like bid-ask-spread, volume imbalance and transaction volumes to accomplish these tasks. Besides that, there are just some publications covering financial forecasting with deep learning methods. Shen, Chao and Zhao [8] used Deep Belief Networks for forecasting foreign exchanges rates and were able to outperform a classical feed-forward network. Yoshihara et al. [9] were able to outperform a support vector machine in stock-price prediction while using news sentiment based features in combination with stacked Restricted Boltzmann Machines. Also, Kuremoto et al. [10] used Restricted Boltzmann Machines and achieved decent results in modelling chaotic time series. Regardless, the combination of deep learning and market microstructure seems not to have been studied yet.

An artificial neural network (ANN) can be trained to classify arbitrary patterns in data, so it is a useful candidate for financial forecasting. ANNs are generally layered, with each layer of the neural network performing a non-linear transformation of the data. By using a "deep" neural network, the subsequent layers in the structure iteratively abstract and pool information from one layer to the next, allowing the neural network to be trained to discover any arbitrarily complicated function.

Our research uses a well known deep-learning technique called Convolutional Neural Networks (CNNs). A CNN is a specific type of neural network which has been proven to be especially successful at classifying noisy datasets such as images and speech [11], [12]. CNNs promote the use of shared weights within the neural network, which can improve recognition rates without suffering from overfitting. One key advantage of using a CNN is that it automatically identifies basic "features" which are useful for recognition, and which otherwise would have to be hand-crafted. This gives an advantage especially in areas where meaningful features are hard to extract, as in automated trading. However, using a CNN presumes that data includes some form repetitive structure that can be transformed into meaningful features. Due to the fact that the majority of trading in modern markets is undertaken by algorithms, this assumption seems plausible.

In this paper, we train a deep CNN on a full limit-order book dataset, obtained from the London Stock Exchange (LSE) from 2007-2008. We investigate whether deep learning can gain advantage out of the huge amount of available market microstructure data for forecasting purposes. This is a fist step in the direction of adapting this promising new area of machine learning to algorithmic trading. The experimental results show that the proposed method is applicable to deal with market microstructure data, although further work is necessary to improve results.

II. Method

A. Raw market data

A dataset describing all recorded market messages at the London Stock Exchange for the stock GB0031348658 (Barclays PLC), between June 2007 and June 2008, was available. The dataset covered 217 days, with approximately 72000 events per day in average. These market events describe every bid and ask order made in the limit order book, as well as every order deletion and matched trade. A formal description of the data format is given by [13] [14]. From this list of market messages, it is possible to reconstruct the limit order book for the market at any time, and also to see the current best bid and ask price at any time, and also to see every trade as it happened. To exclude the opening and closing auction which are following slightly different mechanisms, we restricted our data to the time of continuous trading (8:01 to 16:30).

B. Parsing market messages

Extensive data preparation was necessary to transform the steady data flow of market messages into a suitable form for training a CNN. First, we rebuilt the limit-order book for each moment of time by simulating the market mechanism used at LSE at the time of recording [14], [15]. As a result, the best bid and ask prices, as well as a snapshot of the limit order book, was made available at each point in time. This time series of limit-order book snapshots, events, and prices allowed us to calculate training targets and inputs for the CNN. The training targets were forecasts of price movement and price volatility, and the CNN inputs were previous snapshots of limit-order book's state and event flow.

C. Input matrices

The fact that CNNs require one or more input *matrices* makes it necessary to define a mapping from the stream of market data that we have into matrix form. We decided to prepare four different input matrices, representing the limit order book current state at time t (A_t), recently happened trades (B_t), incoming buy and sell orders (C_t), and transmitted order deletions (D_t). This four-state representation covers all of the market microstructure inherent information available to human and algorithmic limit-order book traders. Each of these matrices has a time axis of recent events along the x-axis, and price information in the y-axis.

At any given time t, a snapshot of the order book is given by O_t. When presenting the order book in matrix form to the CNN, we first rescale the prices in O_t to be relative to the latest mid-price p_t. To keep the order book size manageable, we only include β price levels above and below price p_t into the order book, for some given price-depth truncation constant, β. Hence a snapshot of the order book O_t can be represented as a column vector of volumes corresponding to the total bid and ask size at each price. By concatenating several column vectors of this kind, side-by-side, from a time window of width N, moving forward in chunks of time α, we can build the matrix A_t of dimension $2\beta \times N/\alpha$, sampled from times $t - N, t - N + \alpha, ..., t$. This matrix is shown at the bottom left of Fig. 1.

Similarly, we can represent any *matched trade* at time t as a column vector of zeros everywhere, except for at the price level at which the order took place, with the magnitude of this single non-zero number set to indicate log trade size. Hence the matrix B_t is formed by concatenating these order vectors from times $t - N$ to t. We downsampled time into chunks of size α by aggregating information, so the final matrix size of B_t was again $2\beta \times N/\alpha$. Downsampling was necessary to reduce input size and thus, allows for training in a reasonable amount of time. However, it remains to be examined in later work whether or not the downsampling process removes valuable information while blurring the exact timing information of a specific event.

The matrices C_t and D_t were formed in an identical way. All input matrices are represented along the bottom of Fig. 1.

The hyperparameters α, β and N needed to be decided in advance. Each of them defines how much information the network is presented and thus has a significant influence on the probable outcome. Their selection addresses the sensible question of how much information is necessary to build a

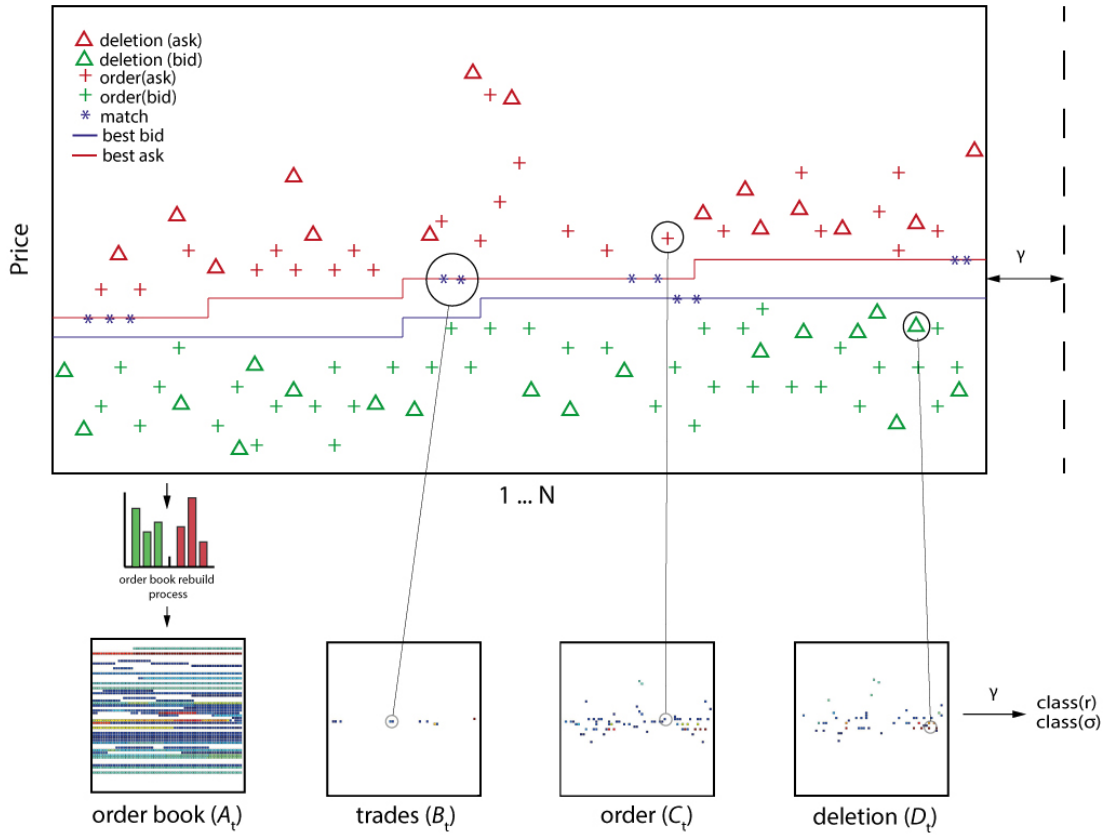

Fig. 1. Model of the designed data transformation process. A market data series (visualized, top) is transformed into four CNN-readable input matrices (bottom), described in Section II-C. The forecasting task is to predict volatility or price trend after γ events into the future.

meaningful and comprehensive representation, in order to learn the dedicated target. Here, we decided to use $N = 180$, $\alpha = 3$, and $\beta = 30$. This parameter selection was made with consideration of the (limited) computational resources available ($N = 180$, $\alpha = 3$) and the assumption that the vast majority of market activity happens in a small corridor around the best bid and ask price (hence, $\beta = 30$).

CNNs are set up to be able to receive several different input "channels". In image analysis, these might represent the different red-green-blue content of an image. In our case, the matrices A_t, B_t, C_t and D_t were each treated as a different input channel. To investigate their relative relevance to our forecasting problem, we trained three different combinations of these input matrices.

D. Labelling data with training targets

The objective for learning was a forecasting task, with look-ahead γ events. There were two learning tasks used: a volatility prediction task, and a price-trend prediction task.

The price-trend prediction problem was constructed by trying to predict the direction of *simple realised return*, defined by

$$r_t = \frac{p_{t+\gamma} - p_t}{p_t}, \tag{1}$$

and the volatility-prediction task consisted of trying to predict the velocity of *realised price volatility*, defined by

$$\sigma_t = StdDev\left(p_t, p_{t+1}, ..., p_{t+\gamma}\right) \tag{2}$$

For these forecasts, we used a look-ahead of $\gamma = 20$ events into the future. For the sake of simplicity, all values were calculated based on mid prices. Regarding the usually very-tiny returns on a high-frequency data basis, σ_t in (2) is calculated using future prices instead of future returns, i.e. differing from the definition of volatility usually used in finance. Hence σ_t serves rather as a measurement of range of the future prices.

Finally, calculated r_t and σ_t were divided into a classes using fixed thresholds th_r and th_σ, as follows:

$$class(r_t) = \begin{cases} 1 & \text{if } r_t < -th_r \\ 3 & \text{if } r_t > +th_r \\ 2 & \text{otherwise} \end{cases} \tag{3}$$

and

$$class(\sigma_t) = \begin{cases} 1 & \text{if } \sigma_t \leq th_\sigma \\ 2 & \text{if } \sigma_t > th_\sigma \end{cases} \tag{4}$$

For the sake of training optimisation, the class thresholds $th_r = 1 \times 10^{-7}$ and $th_\sigma = 0.05$, were determined on the best possible, equal distribution of all classes. That leads to a

three-class problem for price-trend forecasting, and a two-class forecasting problem for price-volatility forecasting[1].

E. Architecture of CNN

Our neural-network architecture was based upon CaffeNet [16] and GoogleNet [17], both of which have been proven successful on difficult image recognition tasks already. We adapted our architecture based on preliminary experiments done on a small sub-sample of the data to make sure that the network started learning. Our architecture is shown in Fig. 2. The model consists of three convolutional layers (kernel=5×5, stride=1), three ReLU layers, and two pooling layers (kernel=2×2, stride=2). The bottom of the network is built by two fully connected layers and a final softmax-layer for classification. To prevent overfitting, dropout layers [18] with a ratio of 75% (except last drop-out layer which has a ratio of 50%, following [18]) were interconnected.

III. Results

We created three different training datasets: I containing just order book inputs (A_t), II containing order flow inputs (B_t + C_t + D_t) and III containing all combined (A_t + B_t + C_t + D_t). A network was trained on each of the three training datasets, and for each of the price-trend and price-volatility prediction tasks, i.e. leading to six networks being trained in total.

All networks were implemented as well as trained using the high-level framework *Caffe* [16]. The cross-entropy function was chosen as the loss function. Training used RMSProp [19] with a base learning rate of 0.0005 and an RMS decay value of 0.98. Training took over 200,000 iterations, corresponding to about 100 hours of training time for each network, on a Tesla K80 GPU setup.

We split the datasets into training, validation and test subsets, in ratio 80:15:5. Figs. 3 and 4 show training progress on the price-prediction and price-volatility prediction tasks. The training progress was almost homogeneous with decreasing loss for the training and validation sets during the training period.

Training progress was validated using the kappa statistic. The kappa statistic κ evaluates current prediction accuracy about the actual class frequency, therefore showing the improvement a classifier gives as a proportion of that of a perfect classifier [20]. More generally, the kappa statistic shows the improvement of a classifier over simple random labelling, whereas a $\kappa \leq 0$ indicates no improvement (respective to random labelling) and a $\kappa = 1$ indicates a perfect classifier. κ was calculated over a full epoch on the test dataset every 20,000 iterations.

Finally, for each dataset, the best performing network snapshot, as judged by kappa on the validation set, was taken to measure accuracy on the retained test set. These results are listed in Table I. The performance on the test set was similar to the performance we observed during training on the validation set, shown in Table II.

[1]In the following the classes are also referred to as (*down*, *equal*, *up*) for price trend and (*high*, *low*) for price-volatility. The threshold was chosen with respect of a best possible equally distribution of training targets over possible classes.

A. Discussion

Accuracy and kappa statistics indicate that all networks started to learn, although with different success. While some networks improved prediction accuracy over time, others had already reached peak performance in the beginning or show fairly unstable behaviour.

In general, three main findings are to highlight.

1) The results indicate that CNNs are able to deal reasonably well ($\kappa > 0$) with the forecasting tasks provided, and start to extract meaningful features.
2) Different learning behaviour is observable between both target groups. In general, predicting price-volatility led to better performance than forecasting price-trend direction.
3) Presenting all available information to the CNNs led to the best performance for both price-trend and price-volatility prediction. Moreover, it seems that both features encourage different learning behaviour.

Although the prediction period is short and the prediction accuracy just slightly higher than random labelling, the results are relevant due to the extreme inherent difficulty of the problem. The highest achieved kappa statistic $\kappa = 0.223$ for price-prediction on combined information source shows that the network makes correct decisions far above random guessing.

Analysing the feature maps learned by the network allows for getting an idea about what the network might be detected [21]. In this case, it seems like the networks extracted features like possible algorithmic patterns as well as detectors for volatility gaps. The latter is already a well-documented phenomenon for having an impact on price moves [22]. Once more, this is an indication of reasonable learning.

The different behaviour in learning and prediction indicates that event-flow and order-book snapshots contain different exploitable types of information. Whereas the network trained on order flow for price-move prediction shows a reasonable distinction between class *up* and *down*, the network trained on order-book states only learned to detect if the price stays fairly equal. It is fair to assume that higher prediction accuracy, in the case of using all four input matrices, comes from combining patterns of both sides.

As assessed in preliminary tests, the use of dropout as a regularisation method reduced overfitting significantly. However, overfitting is still observable in almost all networks and remains a challenge for this class of financial-data forecasting.

IV. Conclusions

In this work, a deep convolutional neural network was trained on market microstructure data for financial forecasting purposes. While analysing different structures and scopes, this study has shown that the selected deep-learning technique, in combination with the developed data transformation, works

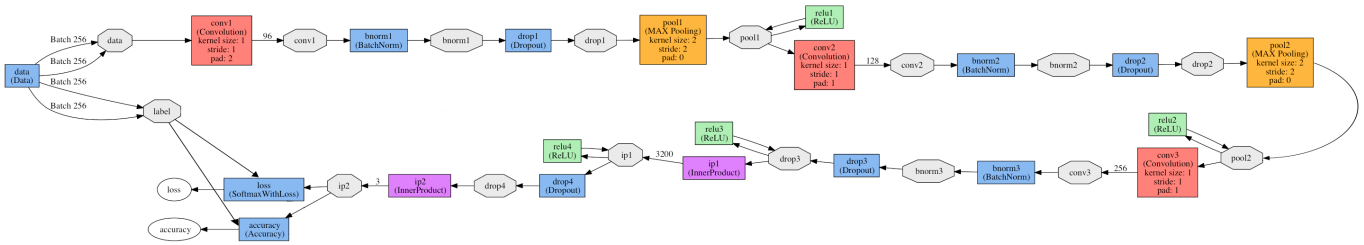

Fig. 2. Schematic overview of the architecture of the CNN as generated by the used Caffee framework

TABLE I
RESULTS OF THE NETWORK PERFORMANCE ON THE TEST SET (OUT-OF-SAMPLE DATA). THE "ITERATION" COLUMN INDICATES WHEN THE NETWORK SNAPSHOT WAS TAKEN. THE * INDICATES MAX-VALUE OF EACH COLUMN.

Dataset	Price-trend			Price-volatility		
	Iteration	Accuracy	Kappa	Iteration	Accuracy	Kappa
I (order book)	120000	0.414	0.122	20000	0.630	0.261
II (order flow)	160000	0.400	0.102	200000	0.563	0.124
III (combined)	120000	0.483*	0.223*	140000	0.682*	0.364*

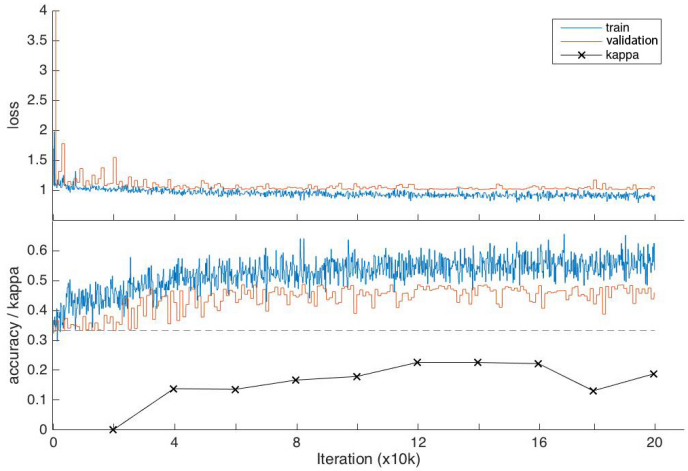

Fig. 3. Training progress of the price-trend prediction network, using the combined inputs of order book and event flow information.

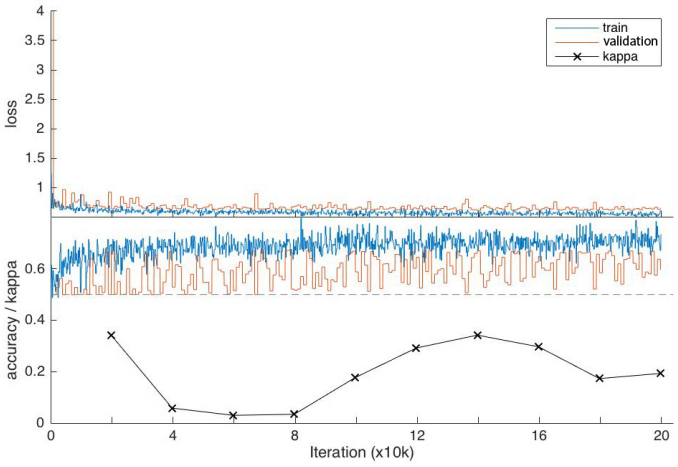

Fig. 4. Training progress of price-volatility prediction network, using the combined inputs of order book and event flow information.

TABLE II
MAXIMAL PERFORMANCE DURING THE TRAINING PERIOD FOR THE VALIDATION DATASETS. THE * INDICATES THE MAX-VALUE OF EACH COLUMN. THE NUMBER IN BRACKETS SHOWS THE CORRESPONDING (BEST) TRAINING ITERATION.

Dataset	Price-trend		Price-volatility	
	Accuracy	Kappa	Accuracy	Kappa
I (order book)	0.430 (156400)	0.128 (120000)	0.649 (41000)	0.243 (20000)
II (event flow)	0.400 (143000)	0.093 (160000)	0.639 (142000)	0.137 (200000)
III (combined)	0.487* (79000)	0.226* (120000)	0.673* (78000)	0.342* (140000)

reasonably well for this problem. The results indicate that including both limit order book and order-flow information leads to decent prediction accuracy. This is even more surprising because we did not need to handcraft meaningful input features in advance. Although the basis of decision making is hard to follow, and thus extracting usable theoretical knowledge is difficult, this work constructs a potential proof of concept for a deep-learning system based on market microstructure data.

However, it remains to be seen if the results hold for longer forecasting periods as well as for other stocks and for up-to-date data. Promisingly, informal preliminary experiments using much longer forecasting periods ($\gamma = 500$) showed comparable results. In further work, different setups and network structures could be tested to get a broadened picture about possible fields of applications.

Through the definition of more specific training sets, it is probably possible to make use of the feature extraction ability for gaining theoretical insights about market microstructure patterns around specific market events. First strong indications

of clear extractable patterns were given in this work. Finally, in consideration of the high computational cost, it also remains to be seen if the technique used performs clearly better than other machine learning methods or even methods of econometrics, even though those methods require the manual choice of meaningful input features first.

Altogether, it can be said that deep learning seems to be legitimate as a potential analysis method and is applicable for high-frequency market data.

ACKNOWLEDGEMENT

We thank Prof Dr Tim Landgraf of Freie Universitaet Berlin for supporting this work by granting access to the high-performance cluster.

REFERENCES

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[2] A. Malik and S. Markose, "Price Trends and Liquidity Dynamics in Limit Order Book," 2016, working Paper, University of Essex.

[3] P. Lipinski and A. Brabazon, "Pattern Mining in Ultra-High Frequency Order Books with Self-Organizing Maps," *Applications of Evolutionary Computation*, vol. 8602, pp. 288–298, 2014.

[4] B. Toth, J. Kertesz, and J. D. Farmer, "Studies of the limit order book around large price changes," *European Physical Journal B*, vol. 71, no. 4, pp. 499–510, 2009.

[5] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T. M. Mcginnity, "Detecting Price Manipulation in the Financial Market," in *IEEE Conference on Computational Intelligence for Financial Engineering & Economics*, 2014.

[6] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement learning for optimized trade execution," in *Proceedings of the 23rd international conference on machine learning ICML 06*, vol. 17, no. 1, 2006, pp. 673–680.

[7] M. Kearns and Y. Nevmyvaka, "Machine learning for market microstructure and high frequency trading," in *High frequency trading - New realities for traders, markets and regulators*, 1st ed., D. Easley, M. L. Prado, and M. OHara, Eds. Risk Books, 2013.

[8] F. Shen, J. Chao, and J. Zhao, "Forecasting exchange rate using deep belief networks and conjugate gradient method," *Neurocomputing*, vol. 167, pp. 243–253, 2015.

[9] A. Yoshihara, K. Fujikawa, K. Seki, and K. Uehara, "Predicting Stock Market Trends by Recurrent Deep Neural Networks," *PRICAI 2014: Trends in artificial intelligence*, pp. 759–769, 2014.

[10] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi, "Time series forecasting using a deep belief network with restricted Boltzmann machines," *Neurocomputing*, vol. 137, pp. 47–56, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2013.03.047

[11] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, 2010.

[12] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[13] M. K. Nguyen, N. Rayner, and S. Phelps, "Inferring the state of a double-auction market from empirical high-frequency transaction data," 2008, working Paper 045, CFFEA, University of Essex, Colchester, UK.

[14] J. Puddik, "Historic Order Book Rebuild, Data Description and Guidance notes," 2007.

[15] London Stock Exchange, "Guide to the New Trading System," no. 7, 2006.

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[17] C. Szegedy, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: prevent NN from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[19] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[20] R. Artstein and M. Poesio, "Inter-Coder Agreement for Computational Linguistics," *Computational Linguistics*, vol. 34, no. 4, pp. 555–596, 12 2008.

[21] J. Doering, "Convolutional Neural Networks in Market Microstructure Data," Master's thesis, University of Essex, Essex, United Kingdom, 2016.

[22] J. D. Farmer, L. Gillemot, F. Lillo, S. Mike, and A. Sen, "What really causes large price changes?" *Quantitative Finance*, vol. 4, no. 4, pp. 383–397, 2004.