Algorithmic Trading Using Deep Neural Networks on High Frequency Data

Conference Paper *in* Communications in Computer and Information Science · August 2017

DOI: 10.1007/978-3-319-66963-2_14

6 authors, including:

Jaime Nino
National University of Colombia
10 PUBLICATIONS **208** CITATIONS
SEE PROFILE

German Hernandez
National University of Colombia
67 PUBLICATIONS **519** CITATIONS
SEE PROFILE

Javier Sandoval
Universidad Externado de Colombia
22 PUBLICATIONS **254** CITATIONS
SEE PROFILE

Diego León
Universidad Externado de Colombia
19 PUBLICATIONS **127** CITATIONS
SEE PROFILE

Algorithmic Trading Using Deep Neural Networks on High Frequency Data

Andrés Arévalo[1(✉)], Jaime Niño[1], German Hernandez[1], Javier Sandoval[2], Diego León[2], and Arbey Aragón[1]

[1] Universidad Nacional de Colombia, Bogotá, Colombia
{ararevalom,jhninop,gjhernandezp,aaragonb}@unal.edu.co
[2] Universidad Externado, Bogotá, Colombia
{javier.sandoval,diego.leon}@uexternado.edu.co

Abstract. In this work, a high-frequency trading strategy using Deep Neural Networks (DNNs) is presented. The input information consists of: (i). Current time (hour and minute); (ii). The last n one-minute pseudo-returns, where n is the sliding window size parameter; (iii). The last n one-minute standard deviations of the price; (iv). The last n trend indicator, computed as the slope of the linear model fitted using the transaction prices inside a particular minute. The DNN predicts the next one-minute pseudo-return, this output is later transformed to obtain a the next predicted one-minute average price. This price is used to build a high-frequency trading strategy that buys (sells) when the next predicted average price is above (below) the last closing price.

Keywords: Short-term forecasting · High-frequency trading · Computational finance · Algorithmic trading · Deep Neural Networks

1 Introduction

Even tough financial markets have been focus of attention by investors worldwide, modeling and predicting prices of Financial Assets have been proved to be a very difficult task [4,10,12,13,16]. As time advances, two important families of techniques have been used for financial assets price modeling: The Analytical and the Machine Learning techniques. Analytical techniques are composed by statistical and stochastic models. Within statistical models like Linear Regression (LR), Multiple Linear Regression (MLR), Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroscedasticity (GARCH/N-GARCH). On the stochastic side models, Brownian Motion (BM), Diffusion, Poisson, Jumps, and Levy Processes are found. On the other hand, machine learning techniques include Decisions Trees, Artificial Neural Networks (ANN), Fuzzy Systems, Kernel Methods, Support Methods Machines (SVM) and recently, Deep Neural Networks (DNN), an extension of Artificial Neural Networks [4,10,14].

Since late 1980s, ANNs have been a popular theme in data analysis. Although ANNs have existed for long time and they have been used in many disciplines, only

J.C. Figueroa-García et al. (Eds.): WEA 2017, CCIS 742, pp. 144–155, 2017.
DOI: 10.1007/978-3-319-66963-2_14

since early 1990s they started to be used for financial assets prices modeling [9]. The first known application of ANN in finance "Economic prediction using neural networks: the case of IBM daily stock returns" was published in 1988 [19].

ANNs are inspired by brain structure; they are composed of many neurons that have connections with each other. Each neuron is a processing unit that performs a weighted aggregation of multiple input signals. Depending on its inputs, a particular set of neurons are activated and propagate a transformed signal, through application of a non-linear function such as Unit Step, Hyperbolic Tangent, Gaussian, Rectified Linear, among others [6]. A Multilayer perceptron (MLP) is an ANN that consists of layers of neurons (an input, multiple hidden and an output layers). All neurons in a layer are fully connected to all neurons in the next layer. Therefore, the information always moves forward from inputs to outputs nodes [6].

The Universal Approximation Theorem says a MLP with a single hidden layer and enough neurons can approximate any function either linear or non-linear [6,7]. Nevertheless of their ability to approximate any function, ANNs have several limitations, because it gets stacked on local minimum avoiding the ANN to converge to optimal solution, causing that found weights and biases not necessary approximate the target data; furthermore, over-training causes over-fitting, which means the ANN tend to memorize training data, causing it to generalize poorly [17,22].

A Deep Neural Network (DNN) is a deep MLP (with many layers), which uses DL training techniques. In a DNN, data inputs are transformed from low-level to high-level features. As signals are propagated forward within the network, data is encoded in more complex features until it is capable of learning high level patterns, which are the ones of interest in this work. In recent years, Deep Learning (DL) has emerged as a very robust machine learning technique, improving the limitations of ANNs [2,17]. Furthermore, advances in hardware such as Graphics Processing Units (GPUs), make possible to accelerate training times for deep networks [2,3,11].

This paper is organized as follows: Sect. 2 presents some important definitions for this work. Section 3 describes data preprocessing done. Section 4 presents DNN modeling for forecasting the next one-minute average price. Section 5 describes trading strategy algorithm proposed. Section 6 presents strategy's performance. Section 7 presents strategy validation. Section 8 presents some conclusions and recommendations for future research.

2 Definitions

Below some important definitions are presented:

Log-return. Let p_t be the current closing price and p_{t-1} the previous closing price.

$$R = \ln \frac{p_t}{p_{t-1}} \cdot 100\% = (\ln p_t - \ln p_{t-1}) \cdot 100\% \tag{1}$$

Pseudo-log-return. It is defined as a logarithmic difference (log of quotient) between average prices on consecutive minutes. On the other hand, the typical log-return is a logarithmic difference between closing prices on consecutive minutes. Let $\overline{p_t}$ be the current one-minute average price and $\overline{p_{t-1}}$ the previous one-minute average price.

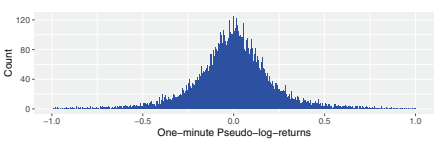
$$\hat{R} = \ln \frac{\overline{p_t}}{\overline{p_{t-1}}} \cdot 100\% = (\ln \overline{p_t} - \ln \overline{p_{t-1}}) \cdot 100\% \tag{2}$$

One-minute Trend Indicator. It is a statistical indicator computed as the linear model's slope ($price = at + b$) fitted on transaction prices for a particular minute, where t is time in milliseconds inside the particular minute; A small slope, close to 0, means that in the next minute, price is going to remain stable. A positive value means that price is going to rise. A negative value means that price is going to fall. Change is proportional to distance value compared to 0; if distance is too high, the price will rise or fall sharply.

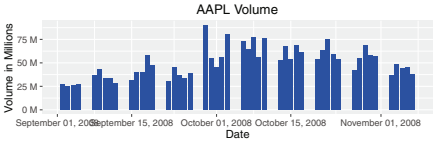
3 Data Preprocessing

Data was taken from NYSE's TAQ database [8], all traded prices for Apple ordinary stock (ticker: AAPL) were downloaded from September 2nd to November 7th of 2008. Figure 1 show price behavior on this period.

(a) Apple Stock Price.



(b) Histogram of One-minute Pseudo-log-returns.



(c) Daily Trading Volume.

Fig. 1. Apple stock

The tick-by-tick dataset is composed by 14,839,395 observations. It has a maximum price on 173.5 dollars and a minimum one on 85 dollars. Figure 1b shows the one-minute pseudo-log-returns histogram. It is some symmetric with mean of -0.002961% and standard deviation of 0.282765. The maximum pseudo-log-return

is 7.952000%, the minimum one is −7.659000%, the first quartile is −0.112200% and the third one is 0.108500. Given these properties, the pseudo-log-returns distribution can be approximated to a normal distribution.

Reviewed literature suggests that any stock log-returns follows approximately a normal distribution with mean zero [5,14,18]. As expected, the pseudo-log-returns distribution behaves like the normal distribution. For this reason, the best variables that describe the market behavior within a minute are the mean price and standard deviation of prices.

Figure 1c shows the daily trading volume. The 98.5% of transactions are carried out with less than 1,000 shares with a mean equal to 165.58 ones. The minimum value is 100 and the maximum one is 5,155,222 shares.

In order to check data consistency, all dates were working days (not holidays and not weekends). All times were during normal trading hours (between 9:30:00 am and 3:59:59 pm EST) and all prices and volumes were positive. Therefore, it was not necessary to delete or adjust records.

All data were summarized with a one-minute detailed level. Three time series were constructed from traded prices: **Average Price**, **Standard Deviation of Prices** and **Trend Indicator**. Each series has 19110 records (49 trading days × 390 minute per day).

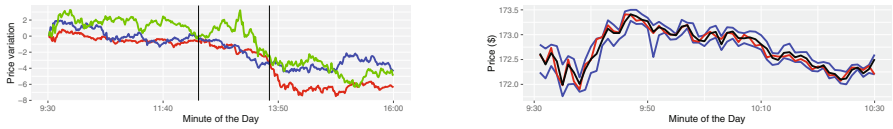
4 Deep Neural Network Modeling

4.1 Features Selection

DNN's inputs were selected in four groups, explained as follows: Current Time, last n pseudo-log-returns, last n standard deviations of prices and last n trend indicators, where n is the window size. Current time group is composed of two inputs: **Current hour** and **Current minute**. The other three groups are composed of n inputs for each one. The number of DNN's inputs I are $3n + 2$. Following paragraphs describe each input group:

Current time: Literature reviewed did not include time as an input. However, hour and minute as integer values were chosen as two additional inputs, because financial markets are affected by regimes that occurs in certain minutes or hours during the trading day. This behavior may be explained by the fact that both human and automatized (machines) traders have strategies that are running in synchronized periods.

To illustrate this affirmation, Fig. 2a shows price changes that occurred at the first, third and sixth day. Approximately, in the minute 170, the stock was traded at the same opening price of corresponding day. Approximately, in the minute 250, the stock price fell 3 dollars relative to the opening price in these days. As these patterns, many more are repeated at certain time of day. In order to identify and to differentiate better these patterns, the current hour and current minute of day were added as additional inputs of DNN. These variables have 7 and 60 possible values ranging from 0 to 6 and from 0 to 59 respectively.



(a) Price changes that occurred at the first (blue line), third (red line) and sixth (green line) day

(b) First 60 Apple Stock Prices. Blue: High and Low. Red: Close. Black: Average.

Fig. 2. Temporal patterns of the time series (Color figure online)

Last n ***pseudo-log-returns:*** It is common to read works about forecasting financial time series with neural networks, which inputs are composed principally by untransformed observations. This is fine for several types of time series, but it is not advisable in financial time series forecasting, because when nominal prices are used, the neural network will train with special conditions (i.e. prices fluctuate between 120 and 170 dollars) and when it is tested against different market conditions (i.e. prices fluctuate between 90 and 120 dollars), overall performance should decline due to poor generalization.

In other words, the neural network learns to identify many static patterns that will be not appearing at all. For example, a pattern like when a price is over 150 dollars, raises to 150.25 dollars and falls to 149.75 dollars, then it will change to 150.50 dollars, could be found, but this pattern never will occur because in the closest future the prices fluctuates between 90 and 120 dollars. However, if prices are transformed into differences or logarithmic returns, not only data variance is stabilized, but also time series acquire temporal independence. For example, at the beginning of the selected period, a pattern, like when the price rises 25 cents and it falls 50 cents, then it will raise 75 cents, could be found and this pattern is more likely to occur in the future. Therefore, the last n one-minute pseudo-log-returns are inputs of DNN.

Last n ***standard deviations of prices:*** The last n one-minute standard deviations of prices are DNN inputs.

Last n ***trend indicators:*** The last n one-minute trend indicators are DNN inputs.

4.2 Output Selection of the Deep Neural Network

The DNN forecasts the next one-minute pseudo-log-return. As shown on Fig. 2b, the average price (black line) is the best descriptor of market behavior. The highest or lowest prices (blue lines) usually are found within a confidence range of average price, therefore the next highest and lowest prices can be estimated from a predicted average price. The closing price (red line) can be any value close to the average price; it coincides sometimes with the highest or lowest price. Unlike the average price, the highest, lowest and closing ones are exposed largely to noise or external market dynamics.

Since this work's objective is to learn market dynamic and to take advantage of it, forecasting average prices could be more profitable than forecasting closing prices. With a good average price forecast, it is known that traded prices will be equal to the average predicted at some point during the next minute.

4.3 Deep Neural Network Architecture

The architecture was selected arbitrarily. It has one input layer, L hidden layers and one output layer. Neuron number in each layer depends on input number I as well as hidden layers number L. In each hidden layer, the number of neurons decreases with a constant factor $\frac{1}{L}$. For example, with five hidden layers: Each layer will have I, $\lceil \frac{4}{5} I \rceil$, $\lceil \frac{3}{5} I \rceil$, $\lceil \frac{2}{5} I \rceil$, $\lceil \frac{1}{5} I \rceil$ neurons respectively. For example, with three hidden layers: Each layer will have I, $\lceil \frac{2}{3} I \rceil$, $\lceil \frac{1}{3} I \rceil$ respectively. The output layer always has one neuron. All neurons in hidden layers use a $Tanh$ activation function, whereas the output neuron uses a $Linear$ activation function.

4.4 Deep Neural Network Training

The final dataset is made up of $19109 - n$ records. Each record contains $3n + 3$ numerical values ($3n+2$ inputs and 1 output). The final dataset was divided into two parts: training data (the first 85% samples) and testing data (the remaining 15% samples). It should be noted that to construct each record, only information from the past is required. Therefore, there is not look-ahead bias and this DNN could be used for a real trading strategy.

For this work, H_2O, an open-source software for big-data analysis [15] was used. It implements algorithms at scale, such as deep learning [1], as well as featuring automatic versions of advanced optimization for DNN training. Additionally, it implements an adaptive learning rate algorithm, called ADADELTA [1], which is described in [21].

4.5 Deep Neural Network Assessment

In order to assess DNN's performance, four statistics were chosen. Let be E real time series and F predicted series:

1. **Mean Squared Error:** $MSE = \frac{1}{n} \sum_{t=1}^{n} (E_t - F_t)^2$
2. **Mean Absolute Percentage Error:** $MAPE = \frac{100}{n} \sum_{t=1}^{n} \frac{|E_t - F_t|}{|E_t|}$
3. **Directional Accuracy:** Percent of predicted directions that matches the real-time series' direction. This measure is unaffected by outliers or variables scales. $DA = \frac{100}{n} \sum_{t=1}^{n} E_t \cdot F_t > 0$

5 Proposed Strategy

DNN predictions are used for building a high-frequency trading strategy: For each trading minute, it always buys (sells) a stock when the next predicted

average price is above (below) the last closing price. When the price yields the predicted average price, it sells (buys) the stock in order to lock the profit. If the price never yields the expected price, it sells (buys) the stock with the last trading price of that minute (closing price), in order to close the position. Additionally, one cent is added/subtracted in order to consider or include the spread, which is the difference between the best quotes (best ask, best bid). Figure 3 shows the strategy flowchart.

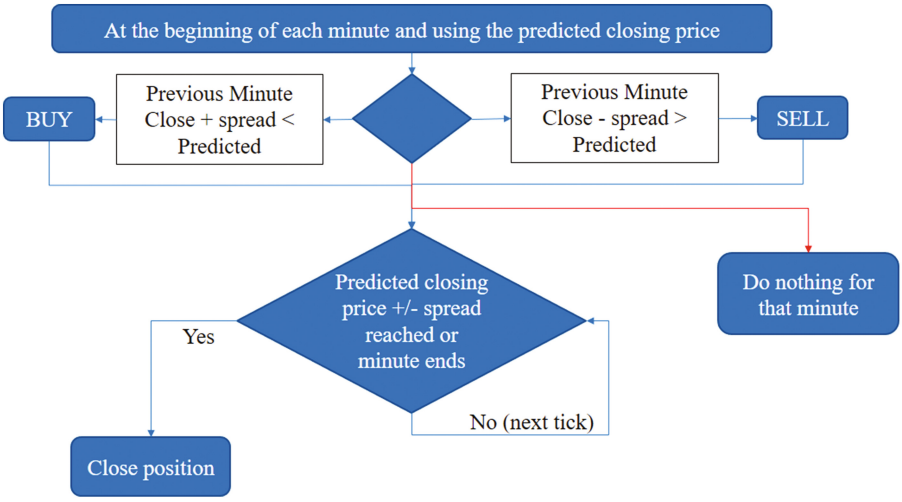


Fig. 3. Strategy flowchart

6 Experiment

6.1 Short-Term Forecasting

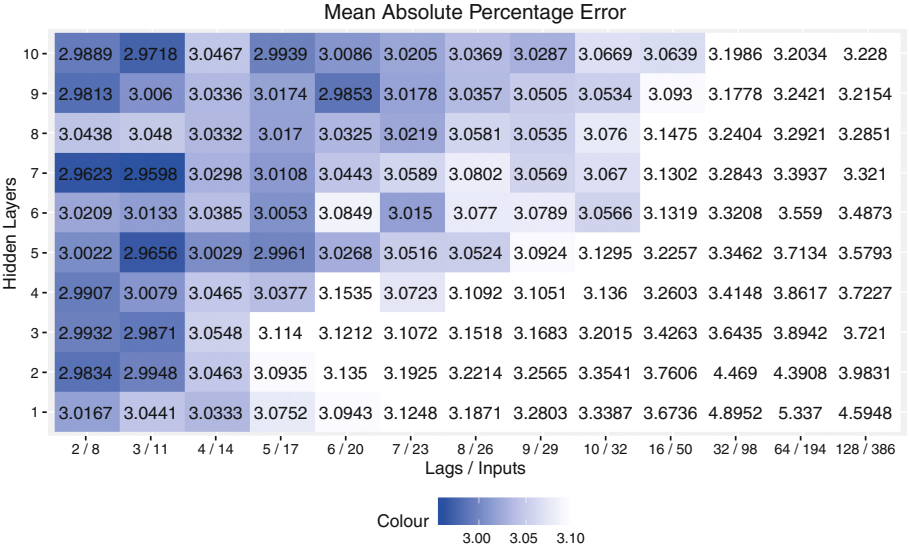
DNNs were trained only with the training data during 50 epochs each one. The chosen ADADELTA parameters were $\rho = 0.9999$ and $\epsilon = 10^{-10}$. On the other hand, DNNs were tested only with testing data.

Tables 1, 2 and 3 illustrate the average DNN performance using different sliding windows sizes (n lags) and number of hidden layers L. The number of neurons in each layer depends on the number of inputs $I = 3n+2$. In each hidden layer, the number of neurons decreases with a constant factor $\frac{1}{L}$. For example, five with hidden layers: Each layer will have I, $\lceil \frac{4}{5} I \rceil$, $\lceil \frac{3}{5} I \rceil$, $\lceil \frac{2}{5} I \rceil$, $\lceil \frac{1}{5} I \rceil$ neurons respectively. For example, with three hidden layers: Each layer will have I, $\lceil \frac{2}{3} I \rceil$, $\lceil \frac{1}{3} I \rceil$ respectively. All DNNs have a single output layer with only one neuron.

For each combination, ten different networks were trained. In each table, the best configuration was highlighted with a darker color. As shown on the Tables 1, 2 and 3, the best combinations are located leftward of x-axis and upward of y-axis, that is, the best yields are obtained by architectures that consider small sliding windows sizes and that are multi-layered.

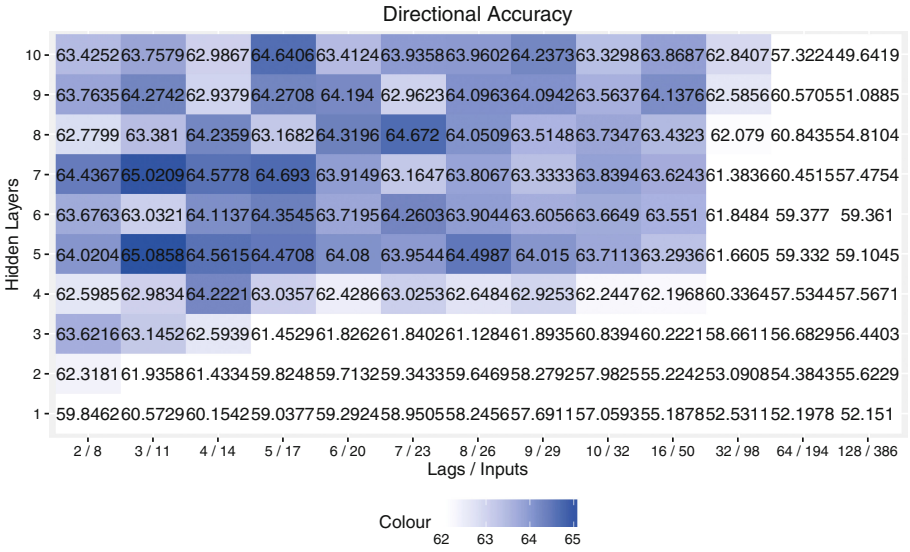
Table 1. DNN performance: MSE

Table 2. DNN performance: MAPE



The best (lowest) MSE were achieved by almost all architectures using small sliding windows of four minutes or less, regardless of the number of hidden layers. The best (lowest) MAPE were achieved by architectures having four or more hidden layers. The best (highest) DA were achieved by architectures having four

Table 3. DNN performance: DA



or more hidden layers. Also, when the number of layers increased, the DNNs achieved good MAPE and DA with larger sliding window sizes.

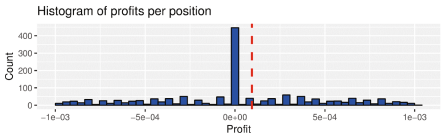
Comparing the figures, the best results were obtained by architectures having four to seven hidden layers, and sliding windows sizes of five or less minutes. Depending on training results, DNN performance may be better, but all networks converge with very similar and homogeneous results.

On average, the DNNs achieved approximately between 0.03 and 0.08 of MSE, between 2.95% and 5.33% of MAPE and between 49.64% and 65.08% of DA. The DNNs are able to predict these sudden rises or falls in price. This information may be useful for any trading strategy.

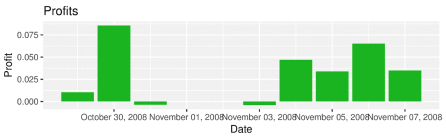
6.2 Strategy Simulation

For the effectiveness of the proposed strategy, it is required a DNN that has a good performance identifying the price direction. For this reason, the architecture with the best directional accuracy (DA) was selected: A DNN that uses a sliding window size of 3 min, and that have five hidden layers, 14 inputs and 1 output; Each hidden layer has 14, 12, 9, 6 and 3 neurons respectively.

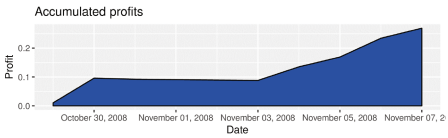
Figure 4 shows the strategy performance during a trading simulation over testing data. The simulation did not consider transaction costs. Buying and selling the equivalent of a dollar in shares, the strategy accumulated approximately $0.28 in 8 trading days.



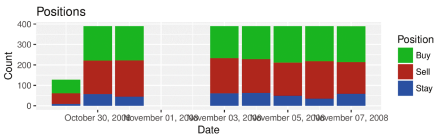
(a) Histogram of profits per position



(b) Daily profits



(c) Accumulated profits



(d) Number of positions

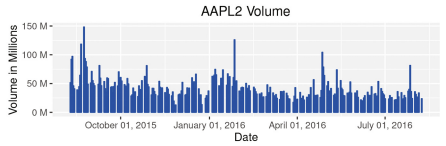
Fig. 4. Trading strategy performance

7 Strategy Validation

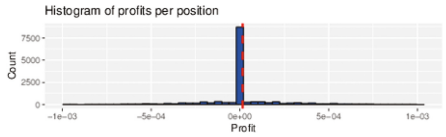
The dataset used in the experiment can be a little dramatic, since it belongs to a period in financial crisis. For this reason, it was decided to select a recent dataset and apply the same strategy using the same network architecture and same data split.

It is important to remember that this high-frequency trading strategy is only applicable to high liquidity stocks, because it requires to open and close

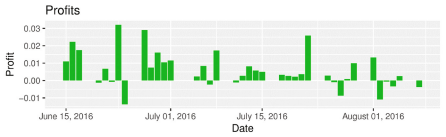
(a) Dataset A (AAPL): Price



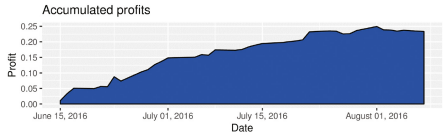
(b) Dataset A (AAPL): Volume



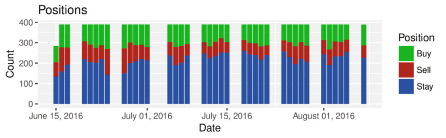
(c) Histogram of profits per position



(d) Daily profits



(e) Accumulated profits



(f) Number of positions

Fig. 5. Dataset a (AAPL): trading strategy performance

positions in a time interval equal or less than one minute. Therefore, from the TAQ database of the NYSE [8], all trade prices for Apple ordinary stock (ticker: AAPL - 2016) were downloaded from the August 10^{th}, 2015 to August 8^{th}, 2016. Figure 5a shows the price and volume behavior of the second dataset.

A DNN with the selected architecture (5 hidden layers and a sliding window size of 3 min) was trained with the dataset A. The data set was split in the same way: The first 85% as training data and the remaining 15% as testing set. Figure 5 shows the strategy performance with the dataset A.

The simulation was performed with the same conditions of the previous experiment (only with the testing data, and the strategy buys or sells the equivalent of a dollar per trade). The strategy accumulated approximately $0.25 in 38 trading days.

8 Conclusions

The strategy has a consistent performance, it turns out to be interesting and yields a good performance. The inclusion of time as an input, proved to be definitive for improving performance, further analysis of this effect is desirable.

Nevertheless, spreads were taken into account, it is desirable to extend the analysis including additional transaction cost. Furthermore, it must be refined in order to implement in a real environment, for example, it could analyze whether it closes its position in the next minute or it keeps it open in order to decrease transaction costs. The strategy can be extended by including other data sources, such as volume and news.

Traders collectively repeat the behavior of the traders that preceded them [20]. Those patterns can be learned by a DNN. The proposed strategy replicates the concept of predicting prices for short periods. Furthermore, adding time as a DNN input allows it to differentiate atypical events and repetitive patterns in market dynamics. Moreover, small data windows sizes are able to explain future prices in a simpler way.

Overall, the DNNs can learn the market dynamic with a reasonable precision and accuracy. Within the deep learning arena, the DNN is the simplest model, as a result, a possible research opportunity could be to evaluate the performance of the strategy using other DL model such as Deep Recurrent Neural Networks, Deep Belief Networks, Convolutional Deep Belief Networks, Deep Coding Networks, among others.

References

1. Arora, A., Candel, A., Lanford, J., LeDell, E., Parmar, V.: Deep Learning with H2O (2015)
2. Bengio, Y.: Learning deep architectures for AI. Found. Trends Mach. Learn. **2**(1), 1–127 (2009). http://dx.doi.org/10.1561/2200000006
3. Dalto, M.: Deep neural networks for time series prediction with applications in ultra-short-term wind forecasting. Rn (Θ1). http://centar.open.hr/_download/repository/KDI-Djalto.pdf

4. De Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. Int. J. Forecast. **22**(3), 443–473 (2006). https://doi.org/10.1016/j.ijforecast.2006.01.001
5. Härdle, W., Kleinow, T., Stahl, G.: Applied Quantitative Finance. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-662-05021-7
6. Haykin, S.: Neural Networks and Learning Machines. Pearson Education, Inc., Upper Saddle River (2009)
7. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Netw. **4**(2), 251–257 (1991). https://doi.org/10.1016/0893-6080(91)90009-T
8. Intercontinental Exchange Inc: TAQ NYSE Trades (2016). http://www.nyxdata.com/data-products/nyse-trades-eod
9. Kaastra, I., Boyd, M.: Designing a neural network for forecasting financial and economic time series. Neurocomputing **10**(3), 215–236 (1996). http://dx.doi.org/10.1016/0925-2312(95)00039-9
10. Krollner, B., Vanstone, B., Finnie, G.: Financial time series forecasting with machine learning techniques: a survey (2010). http://works.bepress.com/bruce_vanstone/17/
11. Larochelle, H., Bengio, Y.: Exploring strategies for training deep neural networks. J. Mach. Learn. Res., 1–40 (2009). http://dl.acm.org/citation.cfm?id=1577070
12. Li, X., Huang, X., Deng, X., Zhu, S.: Enhancing quantitative intra-day stock return prediction by integrating both market news and stock prices information. Neurocomputing **142**, 228–238 (2014). http://dx.doi.org/10.1016/j.neucom.2014.04.043
13. Marszałek, A., Burczyński, T.: Modeling and forecasting financial time series with ordered fuzzy candlesticks. Inf. Sci. **273**, 144–155 (2014). https://doi.org/10.1016/j.ins.2014.03.026
14. Mills, T.C., Markellos, R.N.: The Econometric Modelling of Financial Time Series. Cambridge University Press, Cambridge (2008). https://doi.org/10.1017/CBO9780511817380
15. Nusca, A., Hackett, R., Gupta, S.: Arno Candel, physicist and hacker, 0xdata. Meet Fortune's 2014 Big Data All-Stars (2014). http://fortune.com/2014/08/03/meet-fortunes-2014-big-data-all-stars/
16. Preethi, G., Santhi, B.: Stock market forecasting techniques: a survey. J. Theoret. Appl. Inf. Technol. **46**(1) (2012)
17. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2014). https://doi.org/10.1016/j.neunet.2014.09.003
18. Tsay, R.S.: Analysis of Financial Time Series, vol. 543. Wiley, New York (2005)
19. White, H.: economic prediction using neural networks: the case of IBM daily stock returns. In: IEEE International Conference on Neural Networks, pp. 451–458. IEEE (1988). https://doi.org/10.1109/ICNN.1988.23959
20. Wilder, J.W.: New Concepts in Technical Trading Systems. Trend Research, McLeansville (1978)
21. Zeiler, M.D.: ADADELTA: An Adaptive Learning Rate Method, p. 6 (2012). http://arxiv.org/abs/1212.5701
22. Zekic, M.: Neural network applications in stock market predictions-a methodology analysis. In: Proceedings of the 9th International Conference on Information and Intelligent Systems (1998). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.3374&rep=rep1&type=pdf