High-Frequency Trading Strategy Based on Deep Neural Networks

Conference Paper · August 2016

DOI: 10.1007/978-3-319-42297-8_40

4 authors, including:

Jaime Nino
National University of Colombia
10 PUBLICATIONS **57** CITATIONS
SEE PROFILE

German Hernandez
National University of Colombia
53 PUBLICATIONS **238** CITATIONS
SEE PROFILE

Javier Sandoval
Externado University of Colombia
17 PUBLICATIONS **83** CITATIONS
SEE PROFILE

High-Frequency Trading Strategy Based on Deep Neural Networks

Andrés Arévalo[1(✉)], Jaime Niño[1], German Hernández[1], and Javier Sandoval[2]

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

Abstract. This paper presents a high-frequency strategy based on Deep Neural Networks (DNNs). The DNN was trained on current time (hour and minute), and *n*-lagged one-minute pseudo-returns, price standard deviations and trend indicators in order to forecast the next one-minute average price. The DNN predictions are used to build a high-frequency trading strategy that buys (sells) when the next predicted average price is above (below) the last closing price. The data used for training and testing are the AAPL tick-by-tick transactions from September to November of 2008. The best-found DNN has a 66 % of directional accuracy. This strategy yields an 81 % successful trades during testing period.

Keywords: Computational finance · High-frequency trading · Deep neural networks

1 Introduction

Financial Markets modelling has caught a lot of attention during the recent years due to the growth of financial markets and the large number of investors around the world in pursuit of profits. However, modelling and predicting prices of Financial Assets is not an easy work, due to the complexity and chaotic dynamics of the markets, and the many non-decidable, non-stationary stochastic variables involved [3, 9]. Many researchers from different areas have studied historical patterns of financial time series and they have proposed various models to predict the next value of time series with a limited precision and accuracy [8].

Since late 1980s, neural networks are a popular theme in data analysis. Artificial Neural Networks (ANNs) are inspired by brain structure; they are composed of many neurons that have connections with each other. Each neuron is a processor unit that performs a weighted aggregation of multiple input signals, and propagates a new output signal depending on its internal configuration. ANNs have the ability to extract essential features and learn complex information patterns in high dimensional spaces. Those features have proven useful for forecasting financial time series. Although neural network models have existed for long time and they have been used in many disciplines, only since early 1990s they are used in the field of finance [7]; The first known application for forecasting financial time series was described in [16].

D.-S. Huang et al. (Eds.): ICIC 2016, Part III, LNAI 9773, pp. 424–436, 2016.
DOI: 10.1007/978-3-319-42297-8_40

A Deep Neural Network (DNN) is an ANN with multiple hidden layers between the input and the output layer, such that data inputs are transformed from low-level to high-level features. The input layer is characterized by having many inputs. At each hidden layer, the data are encoded in features of less dimensions by non-linear transformations; then, the next layers refine the learned patterns in high-level features of less dimensions, and so on until it is capable of learning complex patterns, which are of interest in this work. This type of neural networks can learn high-level abstractions from large quantities raw data through intermediate processing and refinement that occurs in each hidden layer [2, 14].

Traditionally, neural networks are trained with the back-propagation algorithm, which consists in initializing the weights matrices of the model with random values. Then the error between network output and desired output is evaluated. In order to identify the neurons that contributed to the error, the error is propagated backwards from the output layer to all neurons in the hidden layers that contributed to it. This process is repeated layer by layer, until all neurons in the network have received an error signal describing their relative contribution to the total error. Later, the weights are updated in order to try to reduce the error. Then the error is calculated again and this process is repeated until a tolerable error or maximal number of iterations is reached [13].

A serious problem of back-propagation algorithm is that the error is diluted exponentially as it passes through hidden layers on their way to the network beginning. In a DNN that has many hidden layers, only the last layers are trained, while the first ones have barely changed. Although DNNs exist long time ago, they were useless because the challenge of training networks with many layers had remained unsolved. This challenge was solved in 2006 by [5], who successfully included paradigms of Deep Learning in Computer Science.

In recent years Deep Learning (DL) has emerged as a very robust machine learning technique, improving limitations of ANN. Models based on DL have begun to arouse the interest of the general public, because they are able to learn useful representations from raw data and they have shown high performance in complex data, such as text, images and even video. However, applications of DNNs in computational finance are limited [15, 18, 19].

The paper is organized as follows: Sect. 2 presents some important definitions of key concepts in this work. Section 3 describes the dataset used for the experiment. Section 4 presents the DNN modelling for forecasting the next one-minute average price of Apple, Inc. within financial crisis of 2008, when a high volatility behaviour was evidenced. Section 5 describes the proposed trading strategy algorithm. Section 6 presents the strategy performance. Moreover, Sect. 7 presents some conclusions and recommendations for future research.

2 Definitions

Bellow some important definitions are presented:

Definition 1. *Log-return.* It is a term commonly used in finance. Let p_t be the current trade or close price and p_{t-1} the previous trade or close price.

$$R = \ln\left(\frac{p_t}{p_{t-1}}\right) = \ln(p_t) - \ln(p_{t-1}) \tag{1}$$

From a log-return R, the original price p_t can be reconstructed easily:

$$p_t = p_{t-1}e^R \tag{2}$$

Definition 2. *Pseudo-log-return.* It is defined as a logarithmic difference (log of quotient) of between average prices on consecutive minutes. On the other hand, the typical log-return is a logarithmic difference of between closing prices on consecutive minutes.

Definition 3. *Trend Indicator.* It is a new statistical indicator created for this work. All trades within each minute are taken, then a linear model ($y = ax + b$) is fitted. The Trend indicator is equal to the parameter a. A small value, close to zero, means that in the next minute, the price is going to remain stable. A positive value means that the price is going to rise. A negative value means that the price is going to fall. Change is proportional to distance value compared to zero; if distance is too high, the price will rise or fall sharply.

3 Dataset Description

From the TAQ database of the NYSE [6], all trade prices for Apple ordinary stock (ticker: AAPL) were downloaded from the September 2nd to November 7th of the year 2008. Figure 1 shows price behaviour in the selected period.

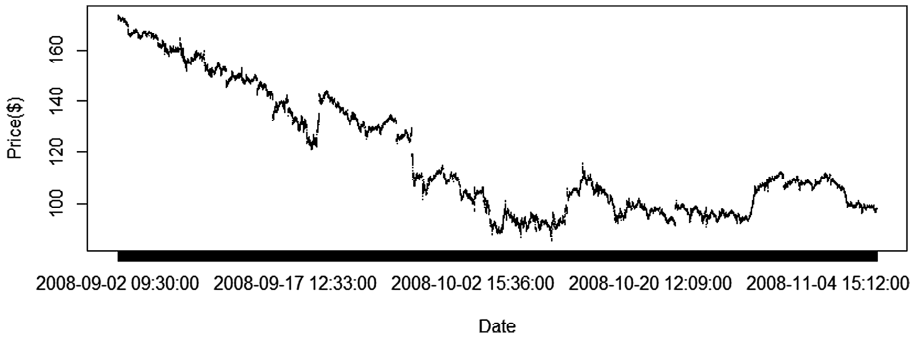

Fig. 1. Apple stock price.

The selected period covers stock crash due to the financial crisis of 2008. During this crash, the AAPL price suffered a dramatic fall from 172 to 98 dollars. This period was chosen intentionally to demonstrate the performance of proposed strategy under high volatility conditions. During a financial crisis, market behaviour is strongly impacted by external factors to the system, such as news, rumours, anxiety of traders,

among others. If a DNN can identify and learn patterns under these difficult conditions, it can yield equal or even better with other time series without financial crisis.

As it is shown on Fig. 2, the distribution of Tick-by-Tick log-returns is some symmetric with mean $-3.802453e^{-8}$, zero in practical terms. The dataset is composed by 14,839,394 observations, has a maximum value on 0.09429531 and a minimum one on -0.09433554.

Reviewed literature suggests that any stock log-returns follows approximately a normal distribution with mean zero [4, 10, 17]. For this reason, the best variables that describe the behaviour of the market within a minute are mean price and standard deviation of prices.

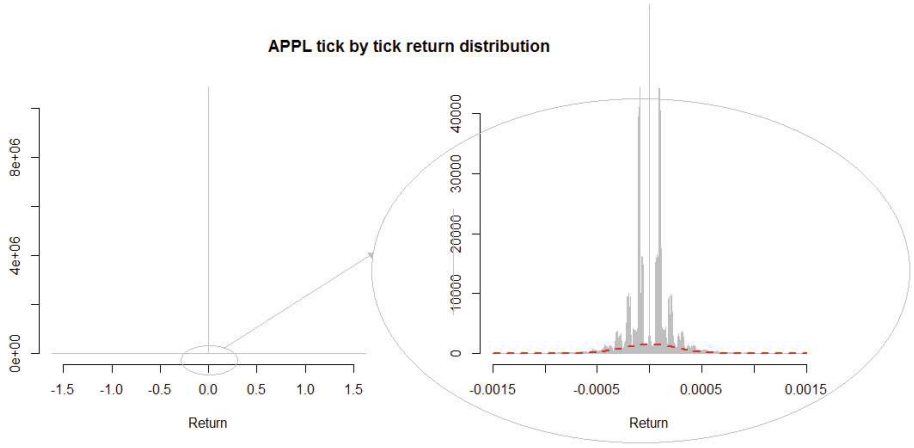

Fig. 2. Distribution of Tick-by-Tick log-returns.

First, the data consistency was verified. All dates were in working days (not holidays and not weekends). All times were during normal hours of trading operation (between 9:30:00 am and 3:59:59 pm EST). All prices and volumes were positive. Therefore, it was not necessary to delete records.

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

4 Deep Neural Network Modelling

4.1 Features Selection

In total four inputs-groups were chosen: Current Time, last n pseudo-log-returns, last n standard deviations of prices and last n trend indicators, where n is the window size. The current time group is composed of two inputs: Current hour and current minute. The others groups are composed of n inputs for each one. In total the number of DNN inputs I is $3n + 2$. The following paragraphs describe each input group:

1. **Current Time:**

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

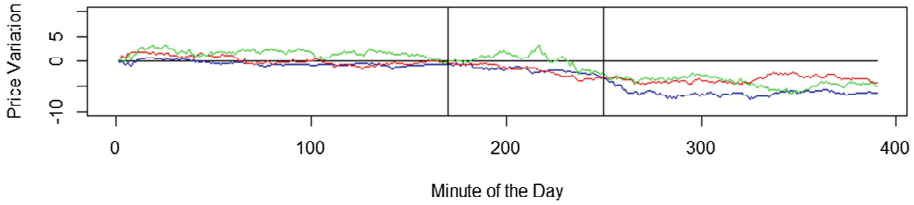

Fig. 3. The price variations that occurred at the first (blue line), third (red line) and sixth (green line) day. (Color figure online)

To illustrate this affirmation, Fig. 3 shows the price variations 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.

2. **Last *n* pseudo-log-returns:**

It is common to see works of neural networks used to forecast time series whose inputs are composed principally by the last untransformed observations. This is fine for several types of time series, but it is not advisable in financial time series forecasting. In any dataset and particularly in the one used in this work, if the nominal prices are used, it will be useless because a neural network will train with special conditions (prices fluctuates between 120 and 170 dollars) and then it will be tested against different market conditions (prices fluctuates between 90 and 120 dollars).

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 the 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 the data variance is stabilized, but also the 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.

3. **Last *n* standard deviations of prices:**

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

4. **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 it is shown on Fig. 4, the average price **(black line)** is the variable that best describes market behaviour. 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 sometimes coincides 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, for example, some traders listen a false rumour about bad news that will cause a sudden fall in the price, in order to reduce losses. As a result, they decide to sell at a lower price than the one traded before. This operation could be done at a certain second and it could affect numerically the highest, lowest or closing prices on the minute.

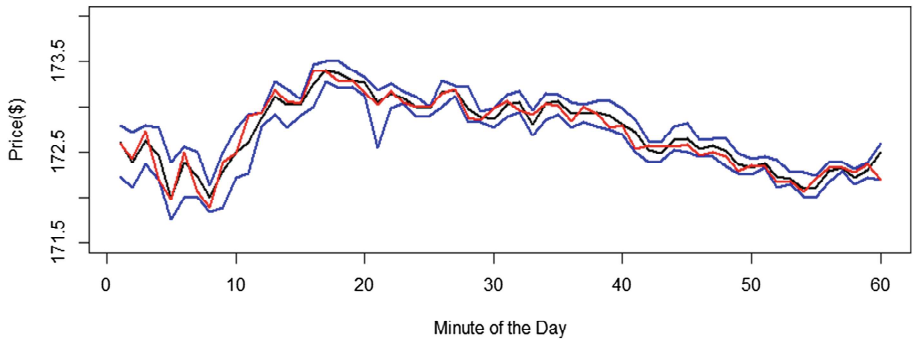

Fig. 4. First 60 one-minute Apple Stock Prices at September 2nd, 2008. Blue: High and Low. Red: Close. Black: Average. (Color figure online)

Since the objective of this work is to learn the dynamics of the market to take advantage of it eventually, the average price forecasts could be more profitable than the closing price forecast. With a good average price forecast, it is known that the stock is going to trade to that predicted value at any moment within the next minute. A real automated trading strategy should wait until the right time (for example, stock price reaches to price forecast) to open or to close their positions.

4.3 Deep Neural Network Architecture

The architecture was selected arbitrarily. It has one input layer, five hidden layers and one output layer. The number of neurons in each layer depends on the number of inputs I. Each layer has I, I, $\lfloor 4I/5 \rfloor$, $\lfloor 3I/5 \rfloor$, $\lfloor 2I/5 \rfloor$, $\lfloor I/5 \rfloor$ and 1 neurons respectively. All neurons use a *tanh* activation function except the output neuron that 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). 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.

As shown on Fig. 1, the dataset has two regimes: one bearish regime (first 50 % samples) and a no-trending one (last 50 % samples). The final dataset was divided into two parts: In-sample data (first 85 % samples in bearish regime and first 85 % samples in no-trending regime) and out-sample data (last 15 % samples in bearish regime and last 15 % samples in no-trending regime). The Fig. 5 shows the splitting.

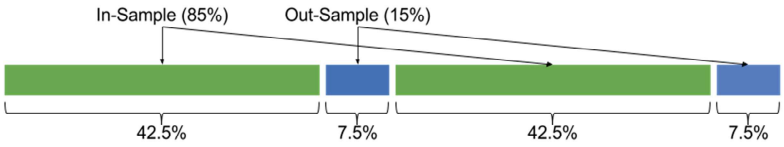

Fig. 5. Data splitting.

For this work, H_2O, an open-source software for big-data analysis [12], 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 [20]. It was chosen in order to improve the learning process, due:

- It is a per-dimension adaptive learning rate method for gradient descent.
- It is not necessary to search parameters for gradient descent manually.
- It is robust to large gradients and noise.

4.5 Deep Neural Network Assessment

In order to assess the DNN performance, two statistics were chosen. Let E as the expected series and F as the series forecast:

- **Mean Squared Error:** $MSE = \frac{1}{n}\sum_{t=1}^{n}(E_t - F_t)^2$
- **Directional Accuracy:** Percent of predicted directions that matches with the ideal differences time series. 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

The DNN predictions are used by the following 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 ensure the profit. If the price never yields the expected price, it sells (buys) the stock with the closing price of the minute, in order to close the position and potentially stop losing positions. Figure 6 shows the strategy flowchart. Below the algorithm is formally presented in pseudo-code:

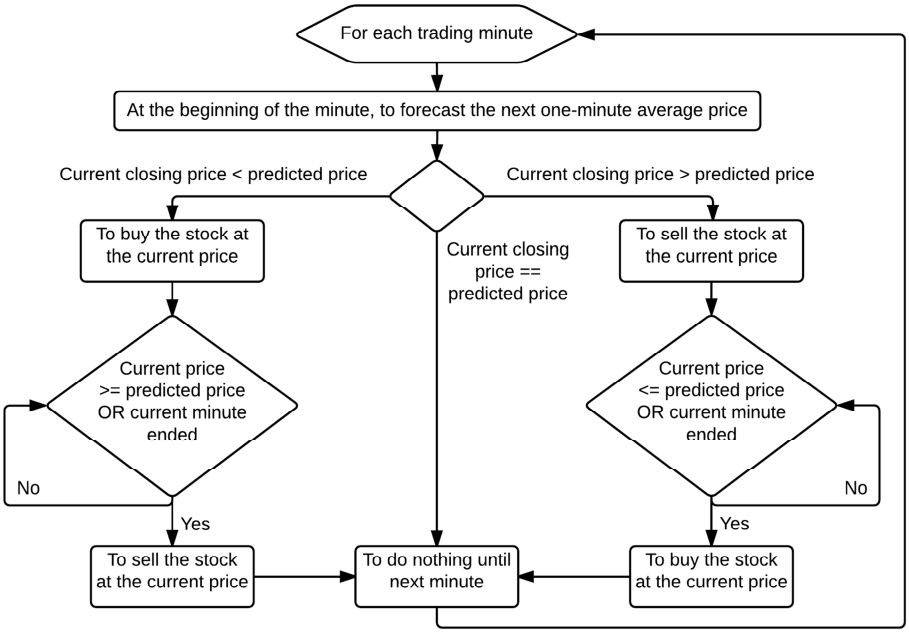

Fig. 6. Strategy flowchart.

```
for each trading minute
  {At the beginning of the minute, to forecast the next
  one-minute average price}
  I ← Current features vector.
  predicted.pseudo.return ← DNN.forecast (I)
  predicted.average.price ← last.average.price
   e^predicted.pseudo.return

  {To open a position}
  if predicted.average.price>previous.closing.price
  then
    {To buy a stock at the current price}
    current.operation ← BUY
  else if predicted.average.price<previous.closing.price
  then
    {To sell a stock at the current price}
    current.operation ← SELL
  else
    continue {To do nothing until next minute}
  end if

  {To close the position}
  while the current minute has not ended
    current.price ← Current stock price
    if current.operation==BUY and
       current.price≥predicted.average.price
    then
      {To sell the stock at the current price and to earn
      the difference}
      current.operation ← null
      continue {To do nothing until next minute}
    end if
    if current.operation==SELL$ and
       current.price≤predicted.average.price
    then
      {To buy the stock at the current price and to earn
      the difference}
      current.operation ← null
      continue {To do nothing until next minute}
    end if
  end while
```

```
{To stop the losses}
if The operation was not closed
then
  if current.operation==BUY
  then
    {To sell the stock at the current price}
  else if current.operation==SELL
  then
    {To buy the stock at the current price}
  end if
  current.operation ← null
 end if
end for
```

6 Experiment

The DNN was trained only with the in-sample data during 50 epochs. The chosen ADADELTA parameters were $\rho = 0.9999$ and $\in = 10^{-10}$. On the other hand, the DNN was tested only with the out-sample date. Table 1 illustrates DNN performance using different windows sizes and the same network architecture. Ten different networks were trained for each parameter. The best results are obtained with small window sizes such as three, four and five.

Table 1. DNN performance.

Window Size	DNN Architecture	Maximum		Minimum		Mean		σ	
		MSE	*DA* (%)	*MSE*	*DA* (%)	*MSE*	*DA* (%)	*MSE*	*DA* (%)
2	8 8:6:4:3:1 1	0.07832	65.71328	0.06768	61.63236	0.07042	64.47506	0.00294	1.30354
3	11 11:8:6:4:2 1	0.07678	66.15492	0.06823	63.67759	0.07125	65.17794	0.00233	0.71620
4	14 14:11:8:5:2 1	0.09158	65.71328	0.07197	63.30659	0.07576	65.07847	0.00579	0.70564
5	17 17:13:10:6:3 1	0.10132	66.05024	0.07569	64.30565	0.08561	64.83949	0.00729	0.52326
6	20 20:16:12:7:3 1	0.10512	65.74816	0.07574	62.99267	0.08514	64.91105	0.00874	0.83872
7	23 23:18:13:9:4 1	0.10383	65.63154	0.08251	63.22400	0.08929	64.43475	0.00634	0.73761
8	26 26:20:15:10:5 1	0.09873	65.60865	0.07813	63.41123	0.08754	64.40879	0.00648	0.68195
9	29 29:23:17:11:5 1	0.09020	65.49197	0.07628	63.78227	0.08437	64.59874	0.00475	0.45499
10	32 32:25:19:12:6 1	0.10250	65.50401	0.07400	63.09731	0.08476	64.55877	0.00829	0.76088
11	35 35:28:21:13:6 1	0.10565	65.24773	0.07702	62.90997	0.08537	64.25680	0.00892	0.83953
12	38 38:30:22:15:7 1	0.09440	65.32961	0.07746	63.69026	0.08698	64.46110	0.00562	0.62078
13	41 41:32:24:16:8 1	0.09442	64.61967	0.07437	61.89811	0.08491	63.79972	0.00618	0.96908
14	44 44:35:26:17:8 1	0.09833	65.32961	0.07849	62.39972	0.08781	64.25531	0.00635	0.94794
15	47 47:37:28:18:9 1	0.09871	64.86392	0.08201	61.86322	0.08916	63.64270	0.00531	1.13385

Overall, the networks achieved between 63 % and 66 % directional accuracy. Depending on training results, DNN performance may be better, but all networks converge with very similar and homogeneous results. The DNN is able to predict these sudden rises or falls in price. This information may be useful for any trading strategy.

Figures 7, 8 and 9 show the strategy performance during a trading simulation over the testing data. The simulation did not consider transaction costs and it was performed with the best-found DNN (66.15492 % of DA). Buying and selling only one stock, the strategy accumulated 72.3036 dollars at the end of the period. It made 2333 successful trades and 520 unsuccessful ones, approximately 81.77 % successful trades.

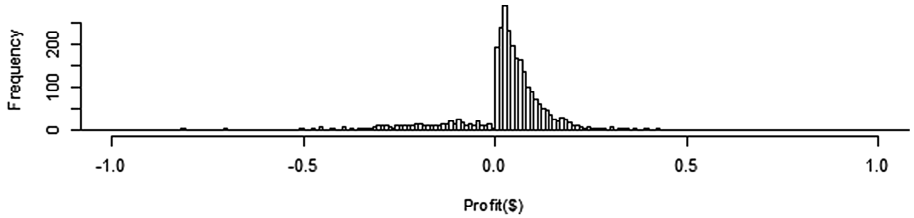

Fig. 7. Profit histogram of the trading strategy

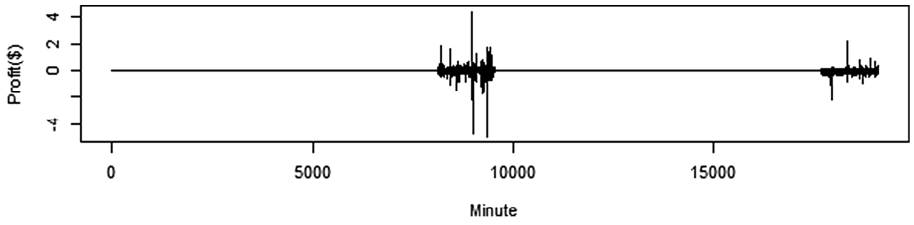

Fig. 8. Profit of the trading strategy

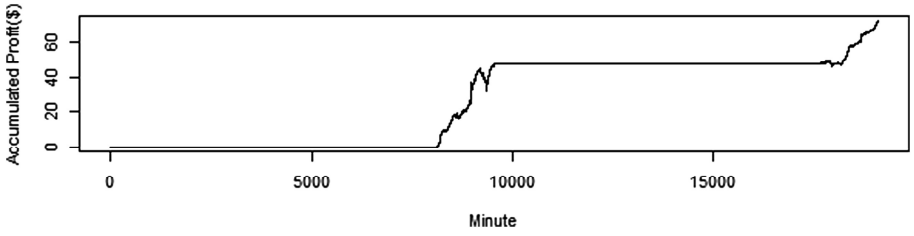

Fig. 9. Cumulated profit of the trading strategy

During the training data period (from 0 % to 42.5 % and from 50 % to 92.5 % in the time series), the strategy did not perform trades, and then it did not yields profits and losses on those minutes. For this reason, Figs. 8 and 9 have a horizontal line during these periods.

7 Conclusions

Although the strategy turns out to be interesting and yields a good performance, it must be refined in order to implement in a real environment, for example, it could analyse whether it closes its position in the next minute or it keeps it open in order to decrease transaction costs.

Traders collectively repeat the behaviour of the traders that preceded them [11]. 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., et al.: Deep Learning with H2O (2015)
2. Bengio, Y.: Learning deep architectures for AI. Found. Trends® Mach. Learn. **2**(1), 1–127 (2009)
3. De Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. Int. J. Forecast. **22**(3), 443–473 (2006)
4. Härdle, W., et al.: Applied Quantitative Finance: Theory and Computational Tools. Springer, Heidelberg (2013)
5. Hinton, G.E., et al.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
6. Intercontinental Exchange Inc.: TAQ NYSE Trades (2016). http://www.nyxdata.com/data-products/nyse-trades-eod
7. Kaastra, I., Boyd, M.: Designing a neural network for forecasting financial and economic time series. Neurocomputing **10**(3), 215–236 (1996)
8. Li, X., et al.: Enhancing quantitative intra-day stock return prediction by integrating both market news and stock prices information. Neurocomputing **142**, 228–238 (2014)
9. Marszałek, A., Burczyński, T.: Modeling and forecasting financial time series with ordered fuzzy candlesticks. Inf. Sci. (Ny) **273**, 144–155 (2014)
10. Mills, T., Markellos, R.: The econometric modelling of financial time series (2008)
11. Murphy, J.J.: Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. Penguin, New York (1999)
12. Nusca, A., et al.: Arno Candel, physicist and hacker, 0xdata. Meet Fortune's 2014 Big Data All-Stars (2014)
13. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: IEEE International Conference on Neural Networks, pp. 586–591. IEEE (1993)
14. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Netw. **61**, 85–117 (2014)

15. Takeuchi, L., Lee, Y.: Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks. cs229.stanford.edu
16. Trippi, R.R., Turban, E.: Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance. Probus Publishing Company, Chicago (1992)
17. Tsay, R.S.: Analysis of Financial Time Series. Wiley, New York (2005)
18. Yeh, S., et al.: Corporate Default Prediction via Deep Learning (2014)
19. Pham, D.-N., Park, S.-B. (eds.): PRICAI 2014. LNCS, vol. 8862. Springer, Heidelberg (2014)
20. Zeiler, M.D.: ADADELTA: An Adaptive Learning Rate Method, 6 (2012)