Available online at www.sciencedirect.com

ScienceDirect

The Journal of Finance and Data Science 5 (2019) 1−11

http://www.keaipublishing.com/en/journals/jfds/

Improving trading technical analysis with TensorFlow Long Short-Term Memory (LSTM) Neural Network

Chenjie Sang, Massimo Di Pierro*

School of Computing, DePaul University, 243 S Wabash Ave, Chicago, IL, 60604, USA

Received 19 May 2018; revised 17 August 2018; accepted 29 October 2018
Available online 14 November 2018

Abstract

In this paper we utilize a Long Short-Term Memory Neural Network to learn from and improve upon traditional trading algorithms used in technical analysis. The rationale behind our study is that the network can learn market behavior and be able to predict when a given strategy is more likely to succeed. We implemented our algorithm in Python pursuing Google's TensorFlow. We show that our strategy, based on a combination of neural network prediction, and traditional technical analysis, performs better than the latter alone.

Keywords: LSTM; Neural network; Trading; Tensorflow; Stock market

1. Introduction

Artificial Neural Networks (ANN) are a class of computational models used in Machine Learning (ML). They have been around for more than 70 years.[7] An ANN can be described as a non-linear fitting algorithm whether the fit is performed by adjusting the weights of information propagation between stacked layers of elementary functions, called neurons, in analogy with the simplest computational units of the human brains. ANNs have proven to be extremely good at feature detection compared with other models. The reason is that it is extremely difficult to describe features in traditional algorithms and, even more so, to create an abstraction paradigm for a generic feature. ANNs have found many applications in finance[12] ranging from modeling stock performance[10] to modeling bankruptcies.[8] Their effectiveness in predicting the stock market has been studied and compared with traditional models[3] and, as one naively would expect, results depends on the stock and on market conditions.

There is a revived interest in ANNs because of the confluence of three factors: 1) extremely promising results from Google in applying Recurrent Neural Networks (RNN) to image and speech recognition; 2) the availability of many open source libraries that implement different types of networks (for example TensorFlow from Google); and 3) the increased availability of cheap computing power, thanks to multi-core GPU, such as those produced by NVIDIA.

* Corresponding author.
E-mail address: massimo.dipierro@gmail.com (M. Di Pierro).
Peer review under responsibility of China Science Publishing & Media Ltd.

https://doi.org/10.1016/j.jfds.2018.10.003

In this paper we propose utilizing an ANN for financial time series forecasting but we take a slightly different spin in respect to previous work. In particular we ask whether we can use the ANN to learn when traditional and established Technical Analysis methods work best.

The term Technical Analysis usually refers to the analysis methodology for forecasting the direction of prices through the study of patterns in past market data, primarily price and volume. Our interest is not so much whether we can detect those patterns using Neural Networks but whether we can use Neural Networks to augment traditional algorithms that are already popular and widely used. We assume that there is no fundamental theoretical reason why most technical analysis algorithms should work other than people use them, thus affecting the stock prices, in a self-fulfilling prophecy. Yet, it is reasonable to assume that those traders do not always follow the algorithms blindly and mechanically. Other considerations factor into their trading decisions. The rationale behind our study is that investors who use technical analysis also apply their own subjective judgment and a neural network can learn that behavior thus predicting when a given strategy is more likely to succeed.

Specifically we consider the following technical analysis trading strategies:

- Simple Moving Average (SMA)
- Relative Strength Index (RSI)
- Moving average convergence divergence (MACD)

For each of those trading strategies we train a neural network to determine whether the strategy produces a profit or a loss at a specific moment in time.

We then compare the performance based on annual cumulative profit per 1 share stock investment from using the naive trading strategy or using the strategy only when the trained neural network recommends doing so.

In our analysis we only consider daily adjusted closing prices and therefore assume that all our trades are executed at the end of the day without any lag or cost.

For each asset we use two distinct data sets: a training data set and a testing data set. Training data set was selected from daily stock market data during 2014 and testing data set is being selected from year as of 2015.

We consider different assets: individual stocks in sectors with top five weights that do not have any major M&A activities and for which we have complete records in the period 2009−2018; and nine indexes representing entire sectors under Standard & Poor 500. Specifically, the performance of the indexes will be obtained from their ETFs identified as follows: XLB, XLE, XLF, XLI, XLK, XLP, XLU, XLV, XLV. The stocks, grouped by sector are the following (see Table 1):

2. Technical analysis trading

All of the technical analysis trading algorithms that we consider here are based on looking for short time trends and using them to make a forecast. As mentioned above, we decided to adopt trading strategies based on three naive trading signals: Simple Moving Average (SMA), Relative Strength Index (RSI), Moving average convergence divergence (MACD). We selected them because they are easy to implement and are commonly used as trading signals.

The SMA signal is defined as follows:

Table 1
Individual stock listing.

S&P 500 Index ETF by Sector								
XLY	XLP	XLF	XLK	XLI	XLB	XLE	XLU	XLV
AMZN	PG	BRK/B	AAPL	BA	PX	XOM	NEE	JNJ
HD	KO	JPM	MSFT	MMM	ECL	CVX	DUK	UNH
NFLX	PEP	BAC	FB	GE	LYB	SLB	SO	PFE
DIS	PM	WFC	GOOGL	UNP	APD	EOG	D	MRK
CMCSA	WMT	C	INTC	HON	SHW	OXY	EXC	AMGN

$$SMA_t = \frac{1}{N}\sum_{i=1}^{N} P_{t_i} \tag{1}$$

where SMA_t is the moving average at time t, P_{t-i} is the price i days before, and N is an arbitrary parameter which we set equal to 20, based on Praekhaow's research[9] which suggests that using a time length equal to 20 generates the most profitable results in the case of short-term trading.

The SMA trading strategy consists of buying when $P_t > SMA_t$ and selling when $P_t < SMA_t$. As previously stated, we assume all positions will be flat at the end of the day with closing price.

The RSI signal is defined as:

$$RSI_t = 100 - \frac{100}{1 + RS_t} \tag{2}$$

where

$$RS_t = \frac{1}{N}\left(\sum_{i=1}^{N} Gain_{t-i}\right) \Big/ \left(\sum_{i=1}^{N} Loss_{t-i}\right) \tag{3}$$

and

$$\text{Gain}_t = max(0, P_t - P_{t-1}) \tag{4}$$

$$\text{Loss}_t = max(0, P_{t-1} - P_t) \tag{5}$$

We choose the value $N = 14$ as our trading strategy following the recommendation of Ref. 11 in which the authors suggest that both $N = 14$ and $N = 21$ generate favorable stock trading results and recommend $N = 14$ for short term results.

Our RSI strategy consists of buying when $RSI_t > 50$ and selling when $RSI_t < 50$.

The *Moving Average Convergence Divergence* (MACD) is a momentum indicator that shows the relationship between two moving averages of prices. It is calculated by subtracting the M-days *Exponential Moving Average* (EMA) from the N-days EMA at an earlier time L.

$$MACD_t = EMA_t^M - EMA_{t-L}^N \tag{6}$$

Here EMA_t is defined recursively:

$$EMA_t^N = \frac{2}{N}\left(O_t - EMA_{t-1}^N\right) + EMA_{t-1}^N \tag{7}$$

In our case we choose $M = 12$, $N = 26$, and $L = 9$ as recommended in Ref. 11. We buy when $MACD_t > 0$ and sell when $MACD_t < 0$. Again we assume the position will be cleared at the end of the day with daily closing price.

3. Neural network and TensorFlow LSTM model

When using a Neural Network we need to distinguish between five distinct concepts: the computer program that simulates the neural network, the network being simulated (the model), the weights of the network, the training data, and the test data.

To some extent it does not matter which computer program is used. This only affects the speed, the file formats, and the hardware we can run the program on. We choose to use TensorFlow[1] from Google because its ease of installation, its speed, and its easy customization using the Python language.

The model is the single most important choice we must make. It is somewhat arbitrary although based on some empirical facts. The choice of the model corresponds to a choice of neurons, their numbers, their layers, and their connection topology. This is equivalent to choosing the overall shape of the fitting function. Different choices yield different results. In this empirical study we decided to utilize the Long Short-Term Memory (LSTM) neural network instead of other neural networks.

Long Short-Term Memory (LSTM) neural network belongs to Recurrent Neural Network (RNN), which is specialized in training time-series/sequential data. LSTM neural networks is structured in a similar way as RNN in a chain structure as follows (see Fig. 1):

A LSTM unit is consisted of four gates: Input Gate; Output Gate; Forget Gate; Update Gate. They are connected in such way that:

$$x_t : \text{Input Tensor} \tag{8}$$

$$h_t : \text{Output Tensor} \tag{9}$$

$$W, b : \text{Weights and Biases functions} \tag{10}$$

where

f_t is the Forget Gate defined by:

$$f_t = \sigma_f\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right) \tag{11}$$

i_t is Input Gate defined by:

$$i_t = \sigma_i(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \tag{12}$$

and o_t is the Output Gate defined by:

$$o_t = \sigma_o(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \tag{13}$$

and finally, c_t is the State Update Tensor defined as:

$$c_t = f_t c_{t-1} + i_t \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \tag{14}$$

and

$$h_t = o_t \tanh(c_t) \tag{15}$$

The TensorFlow LSTM is able to update the current state with information in the past and compute the gradient using truncated Backpropogation Through Time (BPTT).[4] There is an advantage of Truncated BPTT over full BPTT when the length of the input sequential data is large as the former is able to effectively prevent gradient vanishing problem which is common in many neural networks especially Recurrent Neural Networks.

Our sample data consists of 1764 training samples and 504 testing samples. These are relatively small samples, therefore we used a shallow LSTM models consisting of one single hidden layer in additional to the input and output layers. In TensorFlow, the hidden unit size is simply equal to the cell size times 4 since one unit consists of four parts (input gate, output gate, forget gate, hidden state).

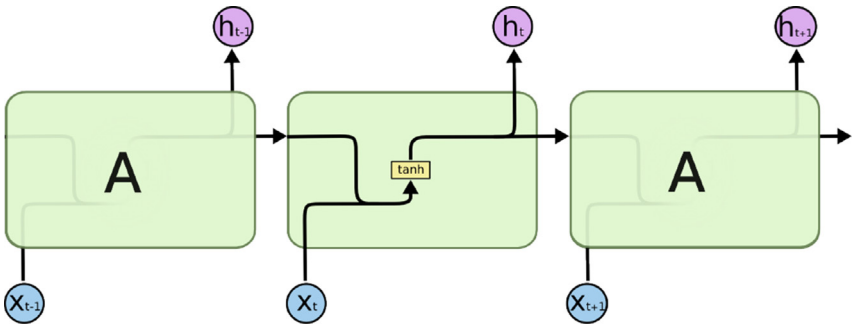

Fig. 1. LSTM chain structure.[6]

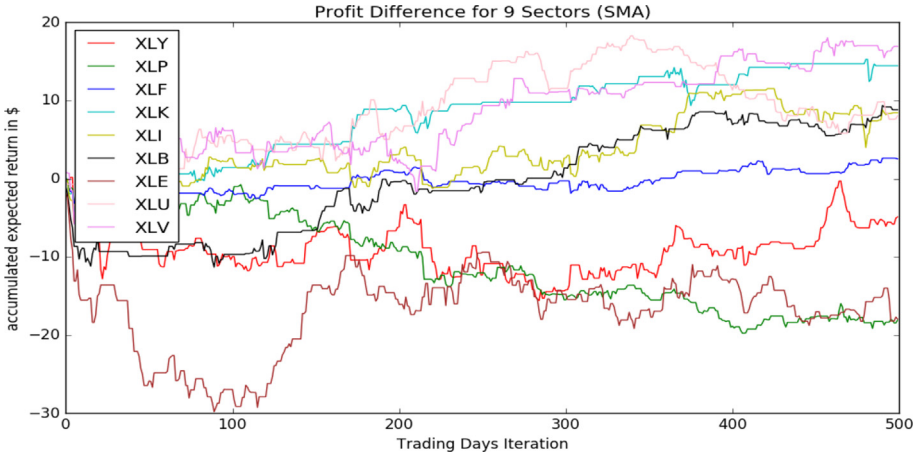

Fig. 2. Cumulative profit differentials (SMA).

We experimented with different learning rates. A large learning rate causes the gradient to bounce and explode. A small learning rate prevents the cost function from converging to its minimum. Also a small learning rate may cause the gradient to remain stuck in local minimum. In our experiments, we found that a learning rate as of 0.0001 works best. The choice of a learning rate will be discussed below.

We also experimented the dropout function in TensorFlow to minimize over-fitting problems but given small data sample size we found it not to be useful and it resulted in under-fitting problems. We found that setting the batch size to 15 works well for our training process.

Another technique which we utilized is the Batch Normalization (BN)[5] method. The general idea of BN is to normalize the input data around sample mean and variance before the activation function. This allows faster learning rate. The batch normalization requires scaling and shifting parameters and they were both trained during the back-propagation process.

The training data set consists of the data we want to fit. The data contains both an output, what we want to predict, and an input information to be used for the prediction. For each trading strategy, every day, we want to predict the success of the strategy ($P_t - P_{t-1}$) given the knowledge of the previous N-day prices P_{t-i} for $i = 1...7$.

From the training data we adjusted the weights (the parameters) of the model to minimize the difference between predicted data and actual data. The testing data is not used for training but it is used exclusively for benchmarks of the

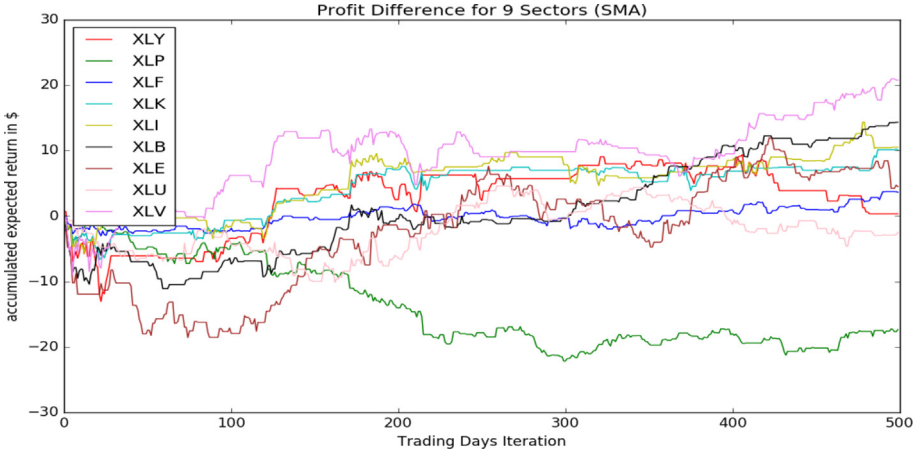

Fig. 3. Cumulative profit differentials (RSI).

trained network. Throughout the training and testing processes, we assume there are no trading related cost (no lending or borrowing cost, no margin interest cost etc). The testing result is based on annual cumulative profit in dollar amount per 1 share of the stock starting at the beginning of the year til the year end.

4. Results

(See Figs. 2−4).

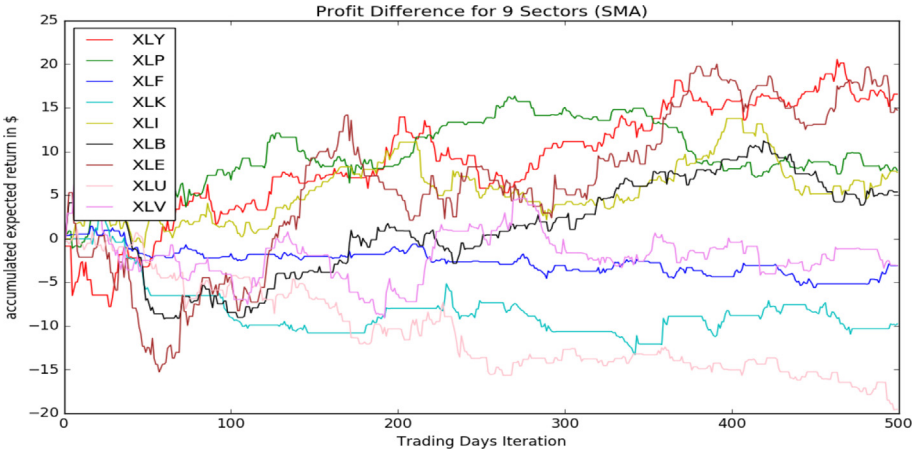

Fig. 4. Cumulative Profit with Differentials (MACD).

5. Sector trading performance before and after training summary in dollars

	SMA	SMA −TF	SMA-Alpha	RSI	RSI-TF	RSI-Alpha	MACD	MACD −TF	MACD-Alpha
XLY	9.11	4.23	−4.88	6.69	6.37	−0.32	−12.85	3.71	16.56
XLP	14.83	−3.27	−18.1	7.88	−9.47	−17.35	−1.72	6.1	7.82
XLF	1.29	3.83	2.54	−0.87	2.76	3.63	−0.9	−3.97	−3.07
XLK	−1.47	12.96	14.43	1.09	11.17	10.08	−3.2	−12.97	−9.77
XLI	2.35	10.78	8.43	2.65	13.17	10.52	−3.81	3.8	7.61
XLB	−2.15	6.65	8.8	−6.11	8.21	14.32	−0.41	4.93	5.34
XLE	7.12	−10.81	−17.93	0.91	5.4	4.49	−19.89	−5.17	14.72
XLU	5.18	13.21	8.03	8.21	5.71	−2.5	11.43	−8.11	−19.54
XLV	−4.24	12.69	16.93	−6.17	14.59	20.76	−4.97	−8.06	−3.09
TOTAL	32.02	50.27	18.25	14.28	57.91	43.63	−36.32	−19.74	16.58

As shown in the plots and table above, TensorFlow generated consistent positive results in the S&P 500 sectors using three trading strategies. Under SMA trading method, TensorFlow outperformed 6 out of 9 sectors; Under RSI trading method, TensorFlow outperformed 6 out of 9 sectors and under MACD trading method, TensorFlow outperformed 5 out of 9 sectors as well. Noticing that under each trading method, the total profit after TensorFlow all outperforms that without TensorFlow training. In numerical analysis, we also found out the annual cumulative profit per 1 share underlying ETF difference under three trading strategies are +$18.25, +$43.63, +$16.58, respectively. In individual stock analysis (the tables are appended in appendix), Out of 45 stocks, for each of the three trading strategies we find 24, 22, and 24 stock, respectively. The annual cumulative profit per share underlying ETF differences are +$198.66, +$10.81, +$368.96.

In this study, we discovered that when changing the parameters, including the learning rate and cell size, affects the performance of the neural network drastically. Consequently we had to carefully choose the parameters setup to ensure TensorFlow's convergence.

6. Time consumption and average cost for different optimizers

	avg time for each sector (seconds)	avg cross entropy cost for each batch training
Gradient Descent	0.305	1.03
Adam	0.321	0.91
Adadelta	0.336	1.187
Adagrad	0.315	0.831
RMSProp	0.331	0.945
Momentum	0.316	1.01

Currently there are few optimizers that are commonly used in deep learning training. In this paper, because of the relatively small sample size, we selected optimizers with algorithm that converge faster. One of such optimizers was discussed by Adam and Adagrad. They have shown it to be relatively fast using data of comparable size. In our empirical study, Adam optimizer not only achieves decent performance in terms of speed and cost but also gives the best performance in computing the 1-yr return of each trading algorithm after training. The table also indicates that no matter what optimizer we choose, it takes about around 0.3 s for TensorFlow training using only one CPU (excluding initialization time and data loading time). Using a GPU would further improve the speed.

7. Learning rate and its related cost comparison

Learning Rate	Training Cost Under SMA TF	Training Cost Under RSI TF	Training Cost Under MACD TF
0.00001	0.99	1.2	1.05
0.00005	0.91	0.95	0.9
0.0001	0.89	0.81	0.85
0.0005	0.89	0.95	0.88
0.001	1.05	1.04	0.94
0.005	1.63	1.77	1.65
0.01	2.05	2.23	2.03
0.05	2.21	2.42	2.39

The table above shows the cross entropy cost with different learning rate for all TensorFlow trading methods. And as we can see learning rate between 0.00005 and 0.001 usually generates the best result with lowest training cost. Thus we are picking learning rate to be 0.0001 since it generates the lowest training cost among all others.

8. Conclusions

Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) have been widely applied in image recognition with remarkable success. In this empirical study we explored whether they the potential to enhance the performance of the trading algorithms. We found positive results and, most importantly, we showed that TensorFlow, and deep learning in general, can be useful to the Financial Industry. Most financial data is time-serially correlated, consequently, Long Short-Term Memory (LSTM) and Recurrent Neural Network perform best and, in our studies, better than traditional trading algorithms.

Acknowledgements

We thank Dr. Mark Shore for useful comments about this manuscript and Prof. Luca De Alfaro for interesting discussions and advice about LSTM.

Appendix

Annual cumulative Profit per 1 Share Stock Before & After Training Using SMA.

stock name	1-yr return before tf in dollars	1-yr return after tf in dollars	1-yr return alpha in dollars
AMZN	−100.9	−332.11	−231.21
HD	−2.375	−25.115	−22.74
NFLX	−49.7205	−31.9855	17.735
DIS	5.388	9.308	3.92
CMCSA	14.235	22.785	8.55
PG	11.5596	−15.1364	−26.696
KO	2.881	10.781	7.9
PEP	11.212	−18.48	−29.692
PM	33.955	−17.325	−51.28
WMT	6.665	1.335	−5.33
BRK/B	5.0395	−23.6505	−28.69
JPM	0.2382	0.3018	0.0636
BAC	0.67	−5.87	−6.54
WFC	9.085	11.985	2.9
C	13.54	29.86	16.32
AAPL	48.7525	20.6825	−28.07
MSFT	−41.575	14.325	55.9
GOOGL	−202.525	211.005	413.53
INTC	9.73	3.81	−5.92
V	−14.403	−13.487	0.916
BA	59.6079	62.7445	3.1366
MMM	47.2381	81.2621	34.024
GE	7.815	−0.565	−8.38
UNP	−6.811	48.179	54.99
HON	8.49	−3.05	−11.54
PX	23.785	51.845	28.06
ECL	−43.775	−18.115	25.66
LYB	−0.82	−7.9	−7.08
APD	8.21	42.97	34.76
SHW	−26.42	−52.28	−25.86
XOM	2.68	−3.85	−6.53
CVX	−17.126	−14.954	2.172
SLB	12.63	18.93	6.3
EOG	−20.74	2.35	23.09
OXY	−3.14	3.15	6.29
NEE	7.365	18.955	11.59
DUK	16.2334	18.7134	2.48
SO	−0.249	−14.611	−14.362
D	13.685	−3.505	−17.19
EXC	8.25	18.49	10.24
JNJ	28.855	25.795	−3.06
UNH	−29.01	1.09	30.1
PFE	2.325	1.243	−1.082
MRK	10.019	−5.239	−15.258
AMGN	76.3951	20.9349	−55.4602
TOTAL	−53.0552	145.6018	198.657

Annual cumulative Profit per 1 Share Stock Before & After Training Using RSI.

stock name	1-yr return before tf in dollars	1-yr return after tf in dollars	1-yr return alpha in dollars
AMZN	145.88	24.55	−121.33
HD	12.005	−25.965	−37.97
NFLX	−53.4205	−59.5195	−6.099
DIS	5.352	18.828	13.476
CMCSA	8.675	−18.275	−26.95
PG	14.8976	−11.9764	−26.874
KO	−5.199	−6.231	−1.032
PEP	9.482	20.22	10.738
PM	29.925	−29.805	−59.73
WMT	10.265	10.875	0.61
BRK/B	21.1095	37.6125	16.503
JPM	1.4382	25.5658	24.1276
BAC	0.63	−1.21	−1.84
WFC	1.785	10.365	8.58
C	8.62	3.17	−5.45
AAPL	15.2625	−1.4375	−16.7
MSFT	−4.875	7.965	12.84
GOOGL	−3.335	123.775	127.11
INTC	1.5	10.42	8.92
V	6.257	−15.247	−21.504
BA	67.4879	55.0499	−12.438
MMM	46.1181	81.9221	35.804
GE	10.095	−1.455	−11.55
UNP	−20.547	23.157	43.704
HON	24.35	1.43	−22.92
PX	12.565	40.945	28.38
ECL	−25.185	−7.555	17.63
LYB	−20.26	−5.48	14.78
APD	−11.67	17.23	28.9
SHW	44.94	136.2	91.26
XOM	6.64	−20.27	−26.91
CVX	−18.386	9.686	28.072
SLB	24.13	−8.03	−32.16
EOG	−7.47	−32.47	−25
OXY	2.92	4.6	1.68
NEE	5.305	−4.845	−10.15
DUK	10.1534	15.7434	5.59
SO	7.551	−11.771	−19.322
D	6.405	−21.055	−27.46
EXC	3.3	7.37	4.07
JNJ	8.375	41.355	32.98
UNH	−1.87	32.89	34.76
PFE	8.185	0.453	−7.732
MRK	7.679	−8.919	−16.598
AMGN	31.9549	−10.0351	−57.99
TOTAL	449.0206	459.8262	10.8056

Annual cumulative Profit per 1 Share Stock Before & After Training Using MACD.

stock name	1-yr return before tf in dollars	1-yr return after tf in dollars	1-yr return alpha in dollars
AMZN	−134.83	9.45	144.28
HD	−35.795	−16.845	18.95
NFLX	23.7595	−49.0455	−72.805
DIS	−6.278	19.298	25.576

(*continued on next page*)

(*continued*)

stock name	1-yr return before tf in dollars	1-yr return after tf in dollars	1-yr return alpha in dollars
CMCSA	6.665	22.865	16.2
PG	−9.3204	−14.1416	−4.8212
KO	−2.779	−11.321	−8.542
PEP	−28.902	9.14	38.042
PM	1.265	13.905	12.64
WMT	−5.505	7.775	13.28
BRK/B	25.4495	−19.8105	−45.26
JPM	−20.6718	−20.8982	−0.2264
BAC	−6.74	2.69	9.43
WFC	1.955	−20.005	−21.96
C	−11.85	10.43	22.28
AAPL	59.6175	31.4375	−28.18
MSFT	−9.035	5.715	14.75
GOOGL	−13.215	98.435	111.65
INTC	4.58	−4.55	−9.13
V	−2.627	−36.497	−33.87
BA	32.4201	−95.2919	−127.712
MMM	−18.9419	−8.0521	10.8898
GE	−2.585	−11.685	−9.1
UNP	−41.669	71.739	113.408
HON	−8.19	−1.49	6.7
PX	14.345	58.585	44.24
ECL	−1.375	−16.835	−15.46
LYB	−12.58	16.08	28.66
APD	−12.53	16.57	29.1
SHW	−62.2	16.06	78.26
XOM	−12.65	19.33	31.98
CVX	−19.384	23.376	42.76
SLB	4.81	1.05	−3.76
EOG	−46.76	−18.56	28.2
OXY	14.25	−1.3	−15.55
NEE	8.695	−19.975	−28.67
DUK	6.1766	−24.6234	−30.8
SO	−14.331	−10.229	4.102
D	−1.955	6.855	8.81
EXC	0.73	−7.15	−7.88
JNJ	14.255	43.935	29.68
UNH	−6.49	−10.37	−3.88
PFE	3.925	−2.285	−6.21
MRK	2.681	−6.321	−9.002
AMGN	33.2551	1.1649	−32.0902
TOTAL	−290.3548	78.6042	368.959

References

1. Abadi Martn, Barham Paul, Chen Jianmin, et al. TensorFlow: a system for large-scale machine learning. In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16). USENIX Association, Berkeley, CA, USA*. 2016:265−283.
3. Fernandez-Rodrguez Fernando, Gonzalez-Martel Christian, Sosvilla-Rivero Simon. On the profitability of technical trading rules based on artificial neural networks: Evidence from the Madrid stock market. *Econ Lett*. 2000;69(1):89−94 [in absence of trading costs, the technical trading rule is always superior to a buy-and-hold strategy for both bear market and stable market episodes. On the other hand, we find that the buy-and-hold strategy generates higher returns than the trading rule based on ANN only for a bull market subperiod.].
4. Hochreiter Sepp, Schmidhuber Jurgen. Long short-term memory. *Neural Comput*. 1997;9(8):17351780.
5. Ioffe Sergey, Szegedy Christian. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." [1502.03167] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 02 Mar. 2015.
6. Olah Christopher. *Understanding LSTM networks. Understanding LSTM networks − Colah's blog*. Colah's Blog; 27 Aug. 2015, colah.github.io/posts/2015-08-Understanding-LSTMs/.

7. McCulloch Warren, Pitts Walter. A logical Calculus of ideas immanent in nervous activity. *Bull Math Biophys*. 1943;5(4):115133. https://doi.org/10.1007/BF02478259.
8. Odom Marcus D, Ramesh Sharda. *A Neural Network Model for Bankruptcy prediction*. 1990. Neural Networks, 1990., 1990 IJCNN International Joint Conference on. IEEE.
9. Praekhaow Puchong. *Determination of Trading Points Using the Moving Average Methods*. 01 June 2010.
10. Refenes Apostolos Nicholas, Zapranis Achileas, Francis Gavin. Stock performance modeling using neural networks: a comparative study with regression models. *Neural Network*. 1994;7(2):375−388.
11. Terence Tai-Leung, Wing-Kam Ng, Venus Khim-Sen Liew. *Revisiting the Performance of MACD and RSI Oscillators*. Germany: MPRA Paper. University Library of Munich; 02 Feb. 2014.
12. Trippi Robert R, Efraim Turban. *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance*. McGraw-Hill, Inc.; 1992.