Financial Forecasting using Deep Learning with an Optimized Trading Strategy

Conference Paper · June 2019

DOI: 10.1109/CEC.2019.8789932

5 authors, including:

Anuar Maratkhan
Nazarbayev University
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Ibrakhim Ilyassov
Nazarbayev University
1 PUBLICATION **1** CITATION
SEE PROFILE

M. Fatih Demirci
TOBB University of Economics and Technology
71 PUBLICATIONS **773** CITATIONS
SEE PROFILE

Murat Ozbayoglu
TOBB University of Economics and Technology
99 PUBLICATIONS **868** CITATIONS
SEE PROFILE

Financial Forecasting using Deep Learning with an Optimized Trading Strategy

Anuar Maratkhan[1], Ibrakhim Ilyassov[1], Madiyar Aitzhanov[1], M. Fatih Demirci[1], Murat Ozbayoglu[2]

[1]Department of Computer Science,
Nazarbayev University, Kazakhstan
{anuar.maratkhan, ibrakhim.ilyassov, madiyar.aitzhanov, muhammed.demirci}@nu.edu.kz
[2]Department of Computer Engineering,
TOBB University of Economics and Technology, Turkey
{mozbayoglu}@etu.edu.tr

***Abstract*—Financial forecasting using computational intelligence nowadays remains a hot topic. Recent improvements in deep neural networks allow us to predict financial market behavior. In our work we first implement a novel approach of [1], which converts financial time-series data to 2-D images and then feeds the generated images to a convolutional neural network as an input. We then hypothesize that the performance of the model can be improved using different techniques. Specifically, in our work, we improve the computational and financial performance of the previous approach by 1) fine-tuning the neural network hyperparameters, 2) creating images with 5 channels corresponding to indicator clusters, 3) improving financial evaluation using take profit and stop loss techniques, 4) evolutionary optimized parameters for trading strategy. The results of this study show that the above-mentioned strategies improve the model considerably. We conclude with future work that can be done in order to further improve the computational and financial performance of the model.**

***Index Terms*—financial forecasting, time-series classification, deep learning, convolutional neural networks, cuckoo search**

I. Introduction

Scientists have been trying to use different computational learning models for decades starting from the previous century. However, the recent literature shows that predicting the financial market is still a hot topic interesting a large number of researchers. Particularly, a neural network-based approach for financial forecasting has been utilized since 1990 [2], and at that time it showed that there was a need for considerable improvements in the theory of neural networks. Since then, many models, which use artificial neural networks (ANN), support vector machines (SVM), hybrid methods, ensemble methods have been proposed as a baseline [3]. Therefore, recent improvements in deep learning have opened an opportunity to build emerging decision-support systems for algorithmic trading in the financial market.

In general, convolutional neural networks (CNN) have shown the best performance results in a number of tasks. The rise of CNNs has started in 2012 when Krizhevsky [4] proposed a deep learning model for classifying images in ImageNet competition. From then on, deeper models that could handle more complex features have been presented to the world including VGGNet [5], Inception Net [6], and ResNet [7]. Nowadays, all state-of-the-art performances in computer vision are achieved with CNN-based approaches. Even though CNNs have proven themselves as the best models for computer vision tasks, they have also been used successfully in various other fields, e.g., natural language processing like sentence classification [8], text understanding [9], and speech recognition [10].

In this study, we first implement [1], a novel approach to financial forecasting using CNNs, which converts 1-D financial time-series data to image-like representations fed as input to the proposed CNN-TA model, and then outputs a specific label for performing action in the financial market, namely "Buy", "Sell", and "Hold". We then show that both computational and financial performance of this model can be improved significantly in a number of different ways. Specifically, parameter optimization and model fine-tuning, clustering similar indicators creating images with 5 channels, and taking the advantage of both profit and stop loss techniques outperform the previous approaches, yielding much better results.

The rest of the paper is organized as follows: we review recent studies related to financial forecasting using deep learning in Section 2, then describe our methodology in Section 3. Results of the proposed framework is provided in Section 4. We then conclude our work and present future suggestions in Section 5.

II. Related Work

Deep learning methods have shown better performance in comparison to classical models of traditional machine learning. The approach presented in [11] show that Long Short-Term Memory (LSTM) network performs better than Random Forest (RAF) and Logistic Regression (LOG) on S&P 500 stock data because linear models can not extract the same information LSTM extracts from the feature space. The study in [12] computing volatility as an important measure of risk presents the superior performance of deep neural network over Support Vector Machines (SVM). Our motivation for using

deep learning for predicting financial markets comes from such studies.

Deep learning models used for financial forecasting are various and range from Multi-Layer Perceptron (MLP) to AutoEncoders (AE). For instance, Sezer et al. [13] used MLP for classifying the current market situation as "buy", "sell", or "hold". However, the results have shown that Buy and Hold (BaH) strategy slightly outperformed the MLP model. However, another deep MLP model [14] optimized by Genetic Algorithms (GA) achieves better performance (average annualized return 11.93%) than mentioned BaH strategy. Besides that, the study argues that GA itself can perform better (15.83% average annualized return) than combined with MLP. Khare et al. [15] compared the performance of MLP trained to predict price in the next 2 minutes with LSTM, which aimed to predict price in 10 minutes. The relative performance of MLP on short-term price prediction was significantly better than the latter one. Moreover, Chong et al. [16] used Principal Component Analysis (PCA), AE and Restricted Boltzmann Machine (RBM) to predict trend movements of Korean market (KOSPI) to trade on high-frequency data.

The other studies used several kinds of models and compared their performance on the task. Honchar and Di Persio [17] presented MLP, LSTM, CNN and one experimental method Wavelet-CNN for neural network-based stock trading. The main goal of the paper was to identify which type of neural network is more accurate in predicting stock price movements (up or down). The experimental model Wavelet-NN uses wavelet transform for feature extraction and then utilizes them as input for the neural network (CNN). This presented model showed a slightly higher return on S&P 500 and FOREX EUR/USD than other considered approaches. Thus, the study concludes that feature extraction or feature engineering may improve the accuracy of ANNs. Moreover, Selvin et al. [18] compared recurrent neural network (RNN), LSTM, and CNN to one of the linear models autoregressive integrated moving average (ARIMA) but did not provide the parameters used for the deep learning models explicitly. The result of the study showed the following performance (in increasing order) - RNN, LSTM, CNN, and ARIMA. The last one performed 5-6% higher than other models. Unlike other papers, Velay and Daniel [19] attempted to classify chart patterns instead of quantitative data with CNN and LSTM, since many traders use these patterns for daily trading. In this paper, the authors worked on the recognition of chart patterns, like a bearish flag, double top, double bottom, etc. The authors tried to reduce the rate of false positives which has a very negative effect on trading, in exchange for false negatives which, in turn, implies missed opportunities. As a result, the LSTM model achieved a better detection rate than CNN.

Moreover, CNN overall showed relatively better performance on financial time-series classification in comparison to other deep neural networks. For example, the results of [20] show that 4-layer CNN trained on 200 epochs significantly outperformed the opponent methods MLP and BaH for predicting future price changes. CNN architecture, particularly, is better because it can extract valuable features from limit order book data which has a lot of noise. For the same reason, authors of [21] introduced the CNN model for predicting volatility and trend of the stock using raw data from the limit order book of London Stock Exchange, high-frequency market. The experimental results revealed that the model forecasts price-volatility (67.3% accuracy) more accurately than price-trend (48.7% accuracy). In comparison, usual ANN [22] does not show such performance because of the lack of feature extraction property which plays a key role in classifying noisy time-series. However, the data fed to CNN is also very important. For instance, Chen et al. [23] transformed financial time-series data of Taiwan stock index in the two-dimensional image of size 20 as input to a convolutional neural network (CNN) for classifying what trend in future will represent. The image creation methods used in the paper include: 1) Gramian Angular Field (GAF); 2) Moving Average Mapping (MAM); 3) Double Moving Average Mapping (DMAM), and were compared to Candlestick chart image control group. The performance shows following results: GAF $>$ DMAM $>$ MAM $>$ Candlestick chart. Thus, CNN is suggested deep learning architecture for predicting financial time-series.

CNN is considered a good architecture not only for financial time-series but also for general-purpose time-series data. The following study [24] argues that the proposed feature-based method outperforms traditional distance-based methods. Specifically, 1-Nearest Neighbor (1-NN) with Dynamic Time Warping (DTW) and Euclidean Distance (ED) distance-metric, and Multi-Layer Perceptron (MLP) have been compared as distance-based methods to the proposed feature-based architecture of CNN. Therefore, the main contribution of the paper is the usage of a feature-based method based on CNN for time series classification, and the study also suggests that the deeper architecture can learn more robust features. Furthermore, authors of [25] used CNN as a feature extractor for time-series classification from scratch without any feature engineering or preprocessing. The study results show that the batch normalization and global average pooling improves the performance by preventing overfitting on small datasets. In addition, Cui et al. [26] show that enhanced MCNN model will perform better with the following 3 stages: transformations, local convolutions, full convolutions. MCNN showed better performance on 41 out of 44 datasets of UCR. Then, it was compared with many state-of-the-art classification models and outperformed many of them. It requires enough training data in order to show good performance. For that, Guennec et al. [27] present several data augmentation techniques used by CNN for time-series classification, which include window slicing (WS), window warping (WW), and dataset mixing (DM). In addition, ANN using a genetic algorithm for finding optimal weights between neurons of the network can help to cope with backpropagation in deeper architectures, and therefore, can better predict future price direction (up or down) [28].

The proposed framework is inspired by the model presented in [1] that converts time-series data of 15 different technical

indicators on 15 intervals ranging from 6 to 20 days to 15x15 images, and then feeds them as inputs to CNN. Further, the model predicts future actions for algorithmic traders in the stock market by classifying converted time-series to 3 class labels: "Hold", "Sell", "Buy". The results show 12% average annualized return on Dow-30 stocks data.

III. Methodology

This section describes improvements on the different parts of the original methodology: data processing, architecture, hyperparameters, image creation, financial evaluation, and evolutionary optimization.

A. Data processing

The DOW-30 stocks data is retrieved using the Yahoo Finance API.[1] Then, the raw data for every stock is converted into 15 different technical indicators with 15 different parameters as it was done in the original work [1]. Technical indicators provide mathematical analysis using stock prices and aim to show the financial market direction. Thus, we received 15x15 images for every day. Every image is labeled using the hand-crafted method described in the original paper [1]. All images with labels are fed to CNN model.

B. Fine-tuning the neural network hyperparameters

After creating the images from time-series data using different technical indicators, the previous framework [1] uses the CNN structure based on LeNet for the final classification, i.e., buy, sell, or hold (see Figure 1). For fine-tuning this model, we initially trained it with 200 epochs and 1028 batch size using Adadelta optimizer. We then managed to increase the speed of learning time with 40 epochs and 128 batch size using Adam optimizer while obtaining the same results. Therefore, decreasing batch size from 1028 to 128 and using Adam optimizer significantly reduced number of training epochs but allowed us to generate the same results as the ones obtained in [1]. In addition, while the original model used simple categorical cross-entropy loss with weights of all classes equal to 1, we used the class-weighted categorical cross-entropy loss to enhance the learning process. Due to the imbalanced data, we also applied different loss weights to different misclassifications. In particular, due to the low occurrences of "Buy" and "Sell" labels, we increase their weights in the loss function, while keeping the "Hold" weight the same. Overall, this process increases the cost for "Buy" and "Sell" misclassifications, yielding better predictions.

C. Technical Indicators Clustering

One of the proposed modifications to the previous approach [1] is clustering the financial indicators into 5 separate groups based on their types. Specifically, we create the following 5 different clusters for technical indicators: 1 - RSI, Williams %R; 2 - SMA, EMA, WMA, HMA, Triple EMA; 3 - CCI, CMO, MACD; 4 - PPO, ROC; 5 - CMFI, DMI, PSI. After that, images of size 15×15 with 5 channels are created such that each channel represents a different technical indicator cluster (Figure 2). For each channel of this model, its rows are repeated to make the image 15×15. For instance, cluster 2 generates a 5×15 image, since it consists of 5 technical indicators. We then repeat the row three times to make it 15×15. The original architecture of the model remains the same, except the input shape for the first convolutional layer, since the input size is changed to $15 \times 15 \times 5$.

[1] finance.yahoo.com

Algorithm 1 CNN-TIC model training and performance evaluation

for ($stock_x$ in $stocks$) **do**
  for ($i = 0$; $i < 15$; i++) **do**
    $training\_data$ = extract data from $(1997 + i)$ to $(2002 + i)$ for 5 years for $stock_x$
    $test\_data$ = extract data from $(2002+i)$ to $(2003+i)$ for 1 year for $stock_x$
    $model = CNN\_TIC(epochs = 40, batch\_size = 128)$
    $model.train(training\_data)$
    $model.test(test\_data)$

D. Financial strategy

The financial strategy of the previous work does not have any take profit or stop loss techniques. Take profit is a practice of selling a share after its price has risen up to a particular value, in order to lock the gains made in the trade. Stop loss is opposite of take profit: if the asset price has fallen to a certain point, the asset is sold in order to prevent the further loss. Since traders usually use these practices to improve their trading strategies, we decide to apply the same techniques to our financial evaluation. In our method we use 10% for take profit and 3% for stop loss. Therefore, when the price of bought asset increases by 10% of bought price or falls by more than 3% from the original price the system automatically sells all shares it has. The original idea of buying an asset on all the money on "Buy" and sell all shares on "Sell" signals remain unchanged. These techniques along with other enhancements improved the financial evaluation results significantly. Algorithm 2 illustrates financial evaluation process with the improved trading strategy.

E. Evolutionary optimization

Since the values for take profit and stop loss in financial strategy were manually found, we decided to find more optimal values using one of the recently created evolutionary optimization algorithms, cuckoo search. Cuckoo search is an optimization algorithm that is inspired by cuckoo reproduction mechanism and can be used to solve various optimization problems [29]. The algorithm works as follows. First, initial population of N host nests with an egg, which represents a solution for the problem, is generated. Then, a cuckoo replaces one egg in a randomly chosen nest, if solution with the cuckoo's egg gives better result than an egg in a nest. Some nests are replaced with new ones with a fixed probability

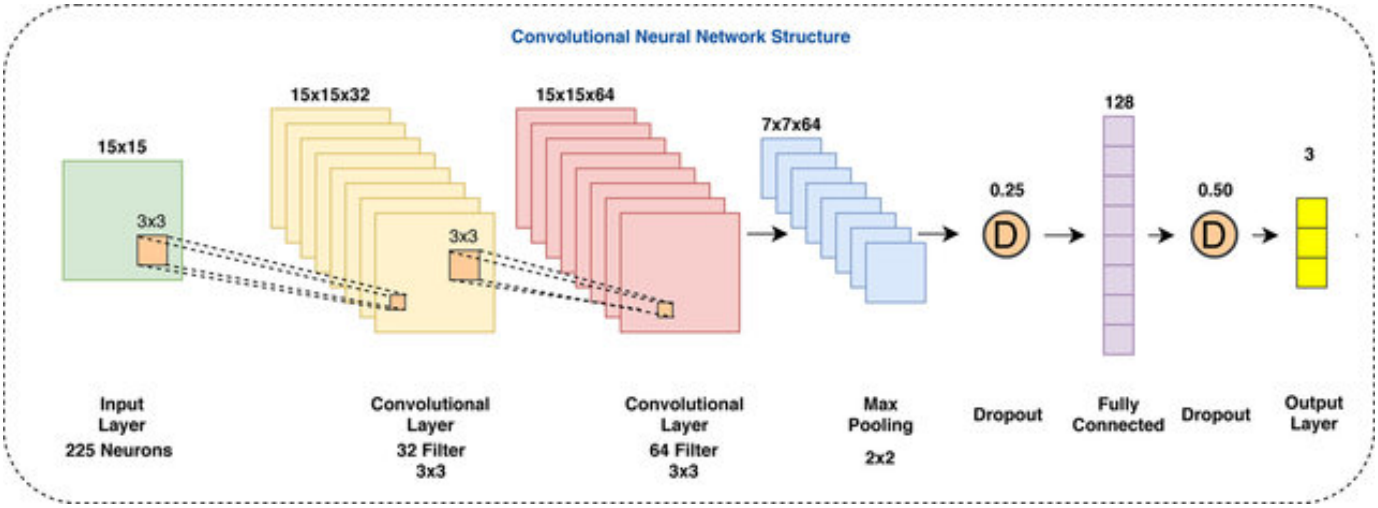

Fig. 1: LeNet architecture implemented in paper [1]

Fig. 2: Image with depth 5, where each channel is a cluster of technical indicators

Algorithm 2 Financial evaluation

```
procedure FinancialEvaluation
balance = $10000
bought_amount = 0
commission = $1
take_profit = 20%
stop_loss = 1%
foreach(price, predicted_action in testDataset)
  if(bought_amount != 0)
    if(price > take_profit_price)
      balance = bought_amount*take_profit_price − commission
      bought_amount = 0
    if(price < stop_loss_price)
      balance = bought_amount * stop_loss_price − commission
      bought_amount = 0
  if(predicted_action == BUY and balance != 0)
    bought_amount = (balance − commission)/price
    balance = 0
    take_profit_price = price * (1.0 + take_profit/100)
    stop_loss_price = price * (1.0 − stop_loss/100)
  if(predicted_action == SELL and bought_amount != 0)
    balance = bought_amount * price − commission
    bought_amount = 0
return balance
```

of 25%. The search stops after a given number of iterations or when a stopping criteria is met. In this work we used SwarmPackagePy library[2], which includes multiple swarm optimization algorithms, as well as cuckoo search.

IV. Results and Discussion

This section discusses results our modifications presented above.

A. CNN with Technical Indicators Clustering

The best model we have found, evolutionary optimized CNN with Technical Indicator Clustering (CNN-TIC), consists of several improvements described in Methodology section. Specifically, we group financial indicators into 5 clusters, use Adam optimizer and class-weighted loss, train the model 15 epochs, and take the advantage of the profit and stop loss techniques. All these improvements yield 20.45% of average annual return on Dow 30 stocks. In addition, we used cuckoo search for optimizing parameters of trading strategy and thus, increased the average annual return up to 29.54%, and named that model as optimized CNN-TIC. Moreover, the recall of both "Buy" and "Sell" labels increases considerably compared to the original values presented in [1] (see Table I).

[2]https://github.com/SISDevelop/SwarmPackagePy

TABLE I: Confusion matrix and evaluation of different models

CNN-TA

Total accuracy: 0.58

	Hold	Buy	Sell	Precision	Recall	f1-score
Hold	52364	18684	23592	0.95	0.55	0.70
Buy	1268	5175	3	0.22	0.80	0.34
Sell	1217	8	5059	0.18	0.81	0.29

Reproduced CNN-TA

Total accuracy: 0.60

	Hold	Buy	Sell	Precision	Recall	f1-score
Hold	53788	18322	20668	0.94	0.58	0.72
Buy	1503	5019	2	0.22	0.77	0.34
Sell	1829	0	4625	0.18	0.72	0.29

Tuned CNN-TA

Total accuracy: 0.45

	Hold	Buy	Sell	Precision	Recall	f1-score
Hold	35793	25356	31629	0.97	0.39	0.55
Buy	598	5923	3	0.19	0.91	0.31
Sell	520	2	5932	0.16	0.92	0.27

CNN-TIC

Total accuracy: 0.32

	Hold	Buy	Sell	Precision	Recall	f1-score
Hold	21807	31211	39760	0.97	0.24	0.38
Buy	352	6156	16	0.16	0.94	0.28
Sell	221	7	6226	0.14	0.96	0.24

B. Financial evaluation

Table II represents a financial evaluation of the existing model from [1], our own implementation of this model, and the two proposed models (Tuned CNN-TA and CNN-TIC). As can bee seen from the evaluation results, the proposed models mostly outperform the existing model in terms of annual return over 2007-2017 years period on 28 companies from Dow Jones stock index. However, the results of standard deviation across all stocks show that the proposed models more volatile but does not differ significantly. Overall, the results present improved performance of the proposed framework offered by hyperparameter fine-tuning, indicator clustering, take profit or stop loss techniques, and evolutionary optimization.

C. Optimized parameters with Cuckoo search

Trading strategy with take profit and stop loss techniqeus greatly improves average annual return. We chose 10% for the take profit and 3% for stop loss. CNN-TIC model with these parameters achieved 20.45% of the average annual return. We decided to tune these hyperparameters and find the best ones using Cuckoo search. The algorithm was run with 150 nests, 50 agents on 8 iterations. Every launch does not always find the best hyperparameters due to the random initial values in this algorithm. As a result, the best parameters are 20% for take profit and 1% for stop loss. The tuned CNN-TA model with these hyperparameters achieved 29.54% of the average annual return.

V. Conclusion and future work

With recent enhancements in deep learning theory, there is an opportunity to implement this knowledge in the direction of financial forecasting. Current literature shows that it is possible to predict financial market behavior using artificial neural networks. In this study we have reproduced the results from [1], and presented a novel framework improving its computational and financial performance by several techniques: 1) fine-tuning neural network hyperparameters, 2) creating images with 5 channels corresponding to technical indicator clusters, 3) improving financial evaluation using take profit and stop loss techniques, 4) finding optimal values for take profit and stop loss utilizing cuckoo search. Eventually, using these improvements we reached 29.54% of average annual return.

Optimized CNN-TIC model utilizes hyperparameters tuning, technical indicators clustering, upgraded financial strategy, and evolutionary optimization. Hyperparameters tuning has accelerated the learning process of the model and thus, allowed us to test various hypotheses faster by about 5 times than the original network. In addition, the use of class-weighted cross-entropy loss has increased the correctness of "Buy" and "Sell" labels detection. Images created with 5 clusters along with take profit and stop loss technique have improved overall financial performance of the model. Although the precision of "Buy" and "Sell" labels have decreased slightly, their recall have increased and thus, the model has missed only a small fraction of true signals. Cuckoo search optimization improved the overall results from 20.45% to 29.54% by finding optimal values for take profit and stop loss parameters.

We conclude by stating that there is still a gap for an improvement, and in future, we are going to try more model improvements like using dataset augmentation, stacking the same CNN as a feature-extractor and LSTM as a classifier for passing the temporal dependency as used in [30], going deeper by adding residual blocks, and using an adaptive parameters for cuckoo search. In addition to that, we will evaluate the labeling method from the original paper by implementing 1-D convolutional model from literature and test other different labeling techniques.

References

[1] O. Sezer and M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, vol. 70, 04 2018.

[2] E. Schneburg, "Stock price prediction using neural networks: A project report," *Neurocomputing*, vol. 2, no. 1, pp. 17 – 27, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/092523129090013H

[3] R. C. Cavalcante, R. C. Brasileiro, V. L. Souza, J. P. Nobrega, and A. L. Oliveira, "Computational intelligence and financial markets: A survey and future directions," *Expert Systems with Applications*, vol. 55, pp. 194 – 211, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741630029X

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

TABLE II: Overall financial evaluation (Dow Jones 30).

Stock	CNN-TA	Reproduced CNN-TA	Tuned CNN-TA	CNN-TIC	Optimized CNN-TIC
AAPL	11.37%	4.04%	-0.21%	23.62%	**41.21%**
AXP	25.05%	11.51%	13.19%	33.88%	**45.11%**
BA	7.03%	9.22%	9.02%	23.57%	**36.07%**
CAT	4.33%	-3.86%	-4.67%	15.06%	**36.25%**
CSCO	10.02%	10.22%	9.82%	26.96%	**31.87%**
CVX	14.91%	2.84%	7.21%	15.48%	**26.32%**
DIS	13.97%	14.44%	11.00%	23.81%	**41.87%**
GE	10.35%	-0.37%	4.26%	18.14%	**41.96%**
GS	6.18%	3.84%	2.35%	25.76%	**26.89%**
HD	15.20%	10.54%	7.36%	20.08%	**33.91%**
IBM	8.15%	9.83%	6.14%	15.79%	**22.45%**
INTC	11.87%	12.30%	11.56%	23.07%	**31.38%**
JNJ	4.62%	2.95%	11.56%	9.45%	**16.06%**
JPM	12.79%	17.37%	19.16%	**35.48%**	32.47%
KO	11.13%	8.97%	9.74%	14.83%	**19.56%**
MCD	17.94%	9.96%	9.06%	15.57%	**27.20%**
MMM	10.88%	11.07%	13.55%	23.27%	**24.20%**
MRK	15.93%	11.14%	8.70%	19.96%	**24.90%**
MSFT	13.43%	12.88%	13.70%	21.27%	**29.56%**
NKE	18.00%	15.14%	21.20%	32.26%	**34.89%**
PFE	8.07%	8.98%	6.79%	14.76%	**26.43%**
PG	9.79%	5.04%	4.73%	8.56%	**11.06%**
TRV	17.34%	19.44%	20.71%	24.18%	**25.84%**
UNH	9.74%	1.78%	3.26%	21.70%	**39.60%**
UTX	9.36%	4.27%	8.42%	19.75%	**26.52%**
VZ	10.23%	6.94%	6.95%	14.48%	**19.35%**
WMT	15.20%	9.29%	8.94%	12.00%	**24.05%**
XOM	14.51%	11.32%	14.57%	20.05%	**30.27%**
Average	12.59%	8.62%	8.94%	20.45%	**29.54%**
St.Dev.	**4.45%**	5.14%	5.76%	6.59%	8.13%

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1746–1751. [Online]. Available: http://aclweb.org/anthology/D14-1181

[9] X. Zhang and Y. LeCun, "Text understanding from scratch," *CoRR*, vol. abs/1502.01710, 2015. [Online]. Available: http://arxiv.org/abs/1502.01710

[10] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, Oct 2014.

[11] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654 – 669, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221717310652

[12] A. Navon and Y. Keller, "Financial time series prediction using deep learning," 11 2017.

[13] O. B. Sezer, A. M. Ozbayoglu, and E. Dogdu, "An artificial neural network-based stock trading system using technical analysis and big data framework," in *Proceedings of the SouthEast Conference*, ser. ACM SE '17. New York, NY, USA: ACM, 2017, pp. 223–226. [Online]. Available: http://doi.acm.org/10.1145/3077286.3077294

[14] O. B. Sezer, M. Ozbayoglu, and E. Dogdu, "A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters," *Procedia Comput. Sci.*, vol. 114, no. C, pp. 473–480, Nov. 2017. [Online]. Available: https://doi.org/10.1016/j.procs.2017.09.031

[15] K. Khare, O. Darekar, P. Gupta, and V. Z. Attar, "Short term stock price prediction using deep learning," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, May 2017, pp. 482–486.

[16] E. Chong, C. Han, and F. C. Park, "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies," *Expert Systems with Applications*, vol. 83, pp. 187 – 205, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417302750

[17] O. Honchar and L. Di Persio, "Artificial neural networks approach to the forecast of stock market price movements," *International Journal of Economics and Management Systems*, vol. Vol.1, pp. 158–162, 01 2016.

[18] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2017, pp. 1643–1647.

[19] M. Velay and F. Daniel, "Stock Chart Pattern recognition with Deep Learning," *ArXiv e-prints*, Aug. 2018.

[20] J. Korczak and M. Hemes, "Deep learning for financial time series forecasting in a-trader system," in *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sept 2017, pp. 905–912.

[21] J. Doering, M. Fairbank, and S. Markose, "Convolutional neural networks applied to high-frequency market microstructure forecasting," in *2017 9th Computer Science and Electronic Engineering (CEEC)*, Sept 2017, pp. 31–36.

[22] M. Dixon, D. Klabjan, and J. H. Bang, "Classification-based financial markets prediction using deep neural networks," *CoRR*, vol. abs/1603.08604, 2016. [Online]. Available: http://arxiv.org/abs/1603.08604

[23] J. Chen, W. Chen, C. Huang, S. Huang, and A. Chen, "Financial time-series data analysis using deep convolutional neural networks," in *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, Nov 2016, pp. 87–92.

[24] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Web-Age Information Management*, F. Li, G. Li, S.-w. Hwang, B. Yao,

and Z. Zhang, Eds. Cham: Springer International Publishing, 2014, pp. 298–310.

[25] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *CoRR*, vol. abs/1611.06455, 2016. [Online]. Available: http://arxiv.org/abs/1611.06455

[26] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *CoRR*, vol. abs/1603.06995, 2016. [Online]. Available: http://arxiv.org/abs/1603.06995

[27] A. L. Guennec, S. Malinowski, and R. Tavenard, "Augmentation for time series classification using convolutional neural networks," 2016.

[28] M. Qiu and Y. Song, "Predicting the direction of stock market index movement using an optimized artificial neural network model," *PLOS ONE*, vol. 11, no. 5, pp. 1–11, 05 2016. [Online]. Available: https://doi.org/10.1371/journal.pone.0155133

[29] M. Mareli and B. Twala, "An adaptive cuckoo search algorithm for optimisation," *Applied Computing and Informatics*, vol. 14, no. 2, pp. 107 – 115, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210832717301679

[30] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," *CoRR*, vol. abs/1503.08909, 2015. [Online]. Available: http://arxiv.org/abs/1503.08909