Financial Forecasting using Deep Learning

Anuar Maratkhan

School of Science and Technology
Nazarbayev University
anuar.maratkhan@nu.edu.kz

***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, reproduce 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 model can be improved using different techniques. Specifically, in our work, we try to improve the performance by converting financial time-series to a feature vector of size 19 instead of converting to image as it was done in the recent literature. We then compare the performance of proposed deep neural network to the following linear models: logistic regression, support vector machine, gradient boosted trees. The results of this study show that the mentioned strategies improve the model slightly. We conclude with future work that needs to be done in order to improve the computational and financial performance of the model.**

***Index Terms*—Financial forecasting, time-series classification, deep learning, convolutional neural networks**

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 the research is still going on and predicting the financial market is a hot topic. 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, there were many models proposed which use artificial neural networks (ANN), support vector machines (SVM), hybrid methods, ensemble methods 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 were presented to the world, which includes VGGNet [5], Inception Net [6], ResNet [7] and many others. 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 model for computer vision tasks, they are also used in various natural language processing tasks like sentence classification [8], text understanding [9], speech recognition [10] and many others.

In this study, we, first, reproduce [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". The proposed model from the study outperformed algorithmic trading strategy called Buy and Hold (BaH), and other neural networks. We then hypothesize that changing the model to 1-D convolutions might improve the performance of the proposed network. Thus, we change image creation part from the study and change it to creation of 1-dimensional feature vector of size 19 with specific parameters for technical indicators. We also compare the model with linear models like logistic regression with regularization (LogReg), support vector machine with 'rbf' kernel (SVM), gradient boosting trees implemented in LightGBM framework [11]. Further, we show that these methods perform similarly and might improve the results of the network slightly.

The rest of the paper is organized as follows: we review recent studies related to financial forecasting using deep learning in section 2, then in section 3 our methodology is explicitly described. Results of our approaches are provided in section 4. In section 5, we conclude our work and present future suggestions.

II. Related Work

Deep learning methods have shown better performance in comparison to classical models of traditional machine learning. [12] argues that Long Short-Term Memory (LSTM) network does 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. A study of [13] which computed volatility as an important measure of risk presented deep neural network that outperformed Support Vector Machine (SVM) model. Thus, we decided to use deep learning for predicting financial markets.

Deep learning models used for financial forecasting are various and range from Multi-Layer Perceptron (MLP) to AutoEncoders (AE). For instance, [14] 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 from [15] 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 alone can do better (15.83% average annualized return) than combined with MLP. Another study [16] 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, [17] 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. [18] 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, [19] 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, [20] 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 [21] 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 [22] 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 [23] 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, [24] tried to transform 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 [25] 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 [26] 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, [27] shows 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, [28] presents 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) [29].

Overall, in our work, as our baseline we decided to use the CNN 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 feds them as input for 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 of the work show 12% average annualized return on Dow-30 stocks data.

III. Methodology

A. Reproducing the paper

We start our work by reproducing the baseline proposed in [1], CNN-TA. As in the paper, we, first, extract historical data of the same stocks as in the paper, Dow Jones 30, from finance.yahoo.com for the period starting from 1/1/2002 to 1/1/2017, and then normalize it according to the adjusted close

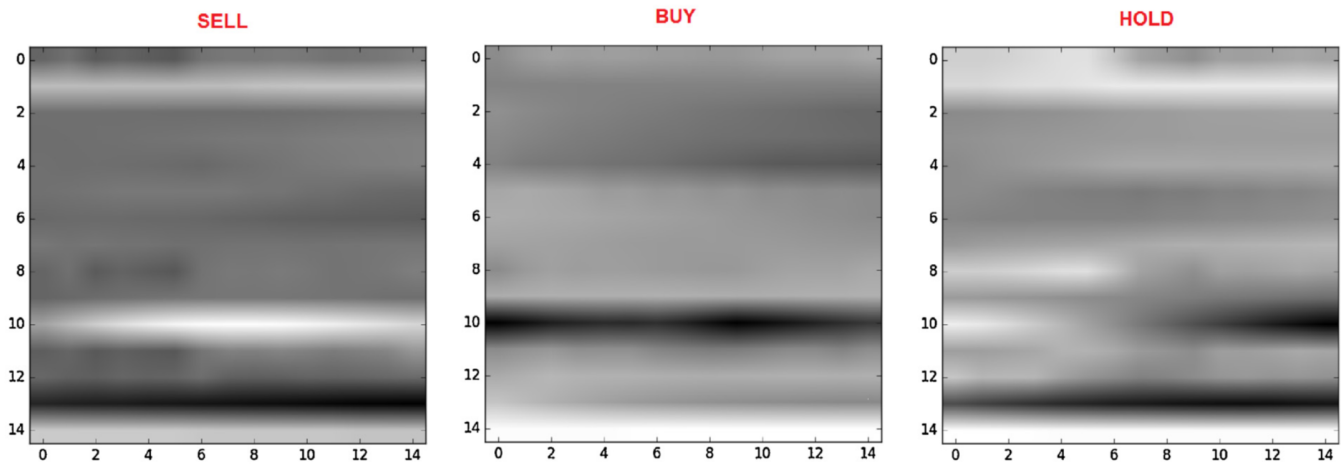

Fig. 1: 15x15 labeled images feed to CNN creating by the same method as in [1].

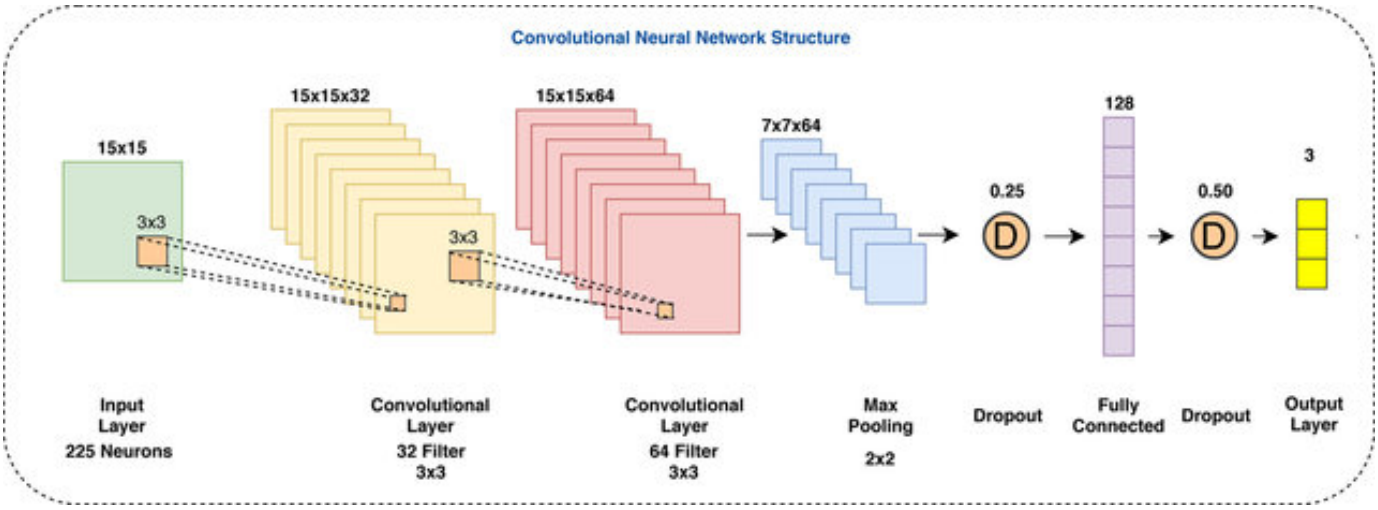

Fig. 2: CNN architecture used in originally in [1].

prices. We then manually label the data as "Hold", "Buy" or "Sell" depending on the top and bottom points in a window of size equal to 11 days. Because in order to gain profit in the stock market, one has to buy low and sell high, we label bottom points as "Buy" and top points as "Sell". When we obtain labeled data normalized according to the adjusted close prices, we proceed by calculating technical indicator values with intervals between 6 and 20 days using TA4J (Technical Analysis for Java) [1]. In general, usage of technical analysis based data is highly appreciated technique in the literature [1], [14]–[16], [24], [29]. The indicators we used are the same as in the original [1] paper: RSI, Williams %R, WMA, EMA, SMA, HMA, Triple EMA, CCI, CMO, MACD, PPO, ROC, CMFI, DMI, and PSI. From those 15 different indicator values at 15 different intervals, we create a 15x15 image that we feed to the CNN further. As in the original paper, we clustered indicator groups by their types and similarities in behaviors like oscillator or trend indicators (in the same order as are listed above). In total, there are five different clusters that are explicitly described and ordered in [30]. As a result, we achieve 1250 images for each 5 year period of training used by the sliding window technique used in the recent literature [1], [12], [16], [19], [24] and then test it on the following year. When we retrieve 15x15 images presented in Figure 1, we then feed those images to the following simple CNN architecture used for MNIST classification: input layer where we pass our generated images (15x15), two convolutional layers with kernel size 3 for extracting the features from images (15x15x32, 15x15x64), a max pooling operation (7x7x64), two dropout layers to prevent overfitting in the dataset of small size (0.25, 0.50), fully connected dense layers for classification (128), and an output layer with probabilities of each label (3). The mentioned architecture is presented in Figure 2 and was trained with the same conditions (same error function, learning rate, number of epochs, batche size). At the end, when the network produces the predicted labels, we further simulate the financial market trading using financial evaluation part from [1].

[1]https://github.com/mdeverdelhan/ta4j

B. One-dimensional convolutional neural network

In the baseline that we have chosen authors did 2-D convolutions of the financial time-series converted to image-like representation based on 15 technical indicators with 15 different parameters (6 to 20-day intervals). In real-world algorithmic trading, analysts usually use specific values for those technical indicators. For instance, the most used day interval for RSI indicator is 14, which means that it will look 14 days back to evaluate overbought or oversold conditions in the price of a stock. Similarly, we have used default or most used parameters for the rest indicators. The indicators and values used for them are presented in Table I. As a result, we receive 19-dimensional feature vector that is then fed to the convolutional neural network. The proposed CNN is changed to 1-D convolutional layers. We left the architecture unchanged, except removing `MaxPooling1D` layer.

TABLE I: Technical indicators and their values used

Indicator	Type	Value
RSI	Momentum indicator	14
Williams %R	Momentum indicator	14
SMA	Trend indicator	12, 26
EMA	Trend indicator	12, 26
WMA	Trend indicator	12, 26
HMA	Trend indicator	12, 26
Triple EMA	Trend indicator	26
CCI	Momentum indicator	20
CMO	Momentum indicator	20
MACD	Trend and momentum indicator	(12, 26)
PPO	Momentum indicator	(12, 26)
ROC	Momentum indicator	21
CMFI	Volume indicator	21
DMI	Trend indicator	14
PSI	Trend indicator	14

C. Logistic regression

Because our task is to classify feature vector either as "Buy", "Sell", or "Hold", we have a multi-class logistic regression. Thus, we decided first to compare it with the simplest model, logistic regression provided in `sklearn` module. The parameters for LogReg used were default, except the `class_weight` parameter because we have data imbalance problem here. The regularization by default is set to L2. The model then was evaluated with the same financial method as proposed deep neural network.

D. Support vector machine

We also implemented SVM from `sklearn` module with `'rbf'` kernel to evaluate the performance of our proposed model. All of the parameters were set to default, except `class_weight` which was as in LogReg set to `'balanced'` in order to make class weights balanced. The model had to predict the class label as previously, "Buy", "Sell", or "Hold". The model then was evaluated with the same financial method as proposed deep neural network.

E. Gradient boosted trees

We also decided to change the model and try LightGBM framework for gradient boosting technique based on decision trees [11]. Gradient boosting is a technique that uses ensemble of weak models to build a high performance model, usually it utilizes decision trees. In this approach, feature vectors were fed into the LightGBM model as input features, not an image. The hyperparameters of the model were standard, except we specified that input data is unbalanced and did not used data augmentation. Other methodology parts remained the same.

IV. Results and discussion

A. Reproducing the paper

The performance of the reproduced model was evaluated in the same way the authors of the original study did: first, we evaluated the computational performance of the model, further, we simulated financial market and evaluated financial performance in terms of annualized average return. Particularly, for computational performance evaluation, we used recall and precision values which more clearly show how model behave on imbalanced data in comparison to test accuracy performance. For instance, as in the paper, we achieved high recall values for "Buy" and "Sell" labels which are good because we can recognize most of the buy and sell points in the market. However, the precision for the same labels is lower which means that the model does not perform well on false signals of "Buy" and "Sell" labels. From Table II it can be clearly seen that we achieved nearly the same results as authors of the original paper did (Table III). There are slight differences in the model performances of the reproduced and the original models because in our work we have used different data augmentation proportions (equal conditions of 1/3 for each label), otherwise, the model did not learn anything and labeled everything as "Hold". The reproduced financial evaluating showed lower results: 8.16% average annualized return as opposed to 12.59% of average annualized return that is presented in [1]. However, the financial performance of our model seems to be volatile for each stock within itself, and in general, for each run of the model training.

TABLE II: Original evaluation of test data from [1] (Dow Jones 30).

Total accuracy: 0.58						
	Hold	**Buy**	**Sell**	**Precision**	**Recall**	**f1-score**
Hold	52,364	18,684	23,592	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
Mean annualized return: 12.59%						

TABLE III: Reproduced evaluation of test data from [1] (Dow Jones 30).

Total accuracy: 0.60						
	Hold	**Buy**	**Sell**	**Precision**	**Recall**	**f1-score**
Hold	35,718	12,234	13,867	0.94	0.58	0.72
Buy	1023	3357	1	0.22	0.77	0.34
Sell	1248	2	3054	0.18	0.71	0.29
Mean annualized return: 8.16%						

B. One-dimensional convolutional neural network

The results of the proposed methodology shows a slight improvement in the financial performance. However, com-

putational performance worsened slightly. As we see from the confusion matrix in Table IV, recall values for all three classes lowered a bit. Even though the test set accuracy is not a good measure for imbalanced dataset problem, the test set accuracy shows insignificantly lower results compared to the baseline. However, as we see from financial evaluation, the results seem to be to some extent better compared to our reproduced baseline results.

TABLE IV: Evaluation of test data for one-dimensional convolutional neural network (Dow Jones 30).

Total accuracy: 0.44						
	Hold	**Buy**	**Sell**	**Precision**	**Recall**	**f1-score**
Hold	24,749	17,603	19,467	0.92	0.40	0.56
Buy	1056	3274	51	0.16	0.75	0.26
Sell	1131	81	3092	0.14	0.72	0.23
Mean annualized return: 8.45%						

C. Logistic regression

The LogReg model results does not seem to be good relatively to other models, and show the lowest financial performance. That is mainly due to imbalanced dataset problem. The logistic regression model even with balanced class weights, seem to be weak on the imbalanced dataset because as we see from the confusion matrix in Table V, the buy and sell signals predicted very rarely, which results in lost opportunity. Because of that we buy or sell relatively rarely.

TABLE V: Evaluation of test data for logistic regression (Dow Jones 30).

Total accuracy: 0.87						
	Hold	**Buy**	**Sell**	**Precision**	**Recall**	**f1-score**
Hold	61,230	208	381	0.88	0.99	0.93
Buy	4340	33	8	0.14	0.01	0.01
Sell	4257	1	46	0.11	0.01	0.02
Mean annualized return: 2.53%						

D. Support vector machine

Support Vector Machines with `'rbf'` kernel performs other way round compared to LogReg model because the model generates more of buy and sell signals. Sell signals are generated the most, which are three times more than buy signals. And as we see from the computational performance in Table VI, the model's both recall and precision values have lowered significantly compared to the proposed model. The financial evaluation showed the second worst results, as it was expected from linear model. In addition, the model sometimes mislabel "Sell" with "Buy" labels and vice-versa, which is essential for the task.

TABLE VI: Evaluation of test data for support vector machine (Dow Jones 30).

Total accuracy: 0.26						
	Hold	**Buy**	**Sell**	**Precision**	**Recall**	**f1-score**
Hold	13,676	10,588	37,555	0.88	0.22	0.35
Buy	1075	1260	2046	0.10	0.29	0.15
Sell	820	385	3099	0.07	0.72	0.13
Mean annualized return: 4.90%						

E. Gradient boosted trees

The results of financial evaluation of the LightGBM model in Table VII showed approximately 5% average annualized return. However, confusion matrix differs from the original one, recall of "buy" and "sell" classes dropped dramatically. In addition, total training time of the model decreased by factor of 60 compared to the original neural network model, which is significantly accelerates the process for testing different theories. However, as we see from confusion matrix, the model predicts "Buy" and "Sell" signals rarely as LogReg model.

TABLE VII: Evaluation of test data for gradient boosted trees (Dow Jones 30).

Total accuracy: 0.85						
	Hold	**Buy**	**Sell**	**Precision**	**Recall**	**f1-score**
Hold	60,409	823	587	0.88	0.98	0.93
Buy	4192	185	4	0.18	0.04	0.07
Sell	4181	6	117	0.17	0.03	0.05
Mean annualized return: 5.59%						

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 reproduce the results from [1], and hypothesize that we can improve the computational and financial performance by several techniques: changing the image creation approach to creating a feature vector of technical indicators with default and most used parameters, changing the model to one-dimensional convolutional neural network. We also compare the proposed deep neural network to linear models in terms of computational and financial performance.

In our approach, we are using 15 different technical indicators with 19 different values in order to create feature vector for training. As we see from the experiments, best both computational performance and financial performance were achieved by the proposed one-dimensional convolutional neural network. Specifically, the performance of the models is the following : CNN ¿ LightGBM ¿ SVM ¿ LogReg. However, the proposed 1-D CNN has showed slightly worse performance in terms of recall, and thus, still needs improvements.

We conclude by stating that there is still a gap for improvement, and in future, we are going to try more model improvements like using dataset augmentation from [28], stacking the same CNN as a feature-extractor and LSTM as a classifier for passing the temporal dependency like was used in [31]. In addition to that, we will change our methodology to multivariate time-series 2-D convolutions presented in [32].

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

[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] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3146–3154. [Online]. Available: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

[12] 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

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

[14] 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

[15] 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

[16] 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.

[17] 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

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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

[27] 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

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

[29] 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

[30] O. Sezer, "Analysis and optimization of the time series data with deep artificial neural networks: Financial estimation algorithms," 06 2018.

[31] 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

[32] M. U. Gudelek, S. A. Boluk, and A. M. Ozbayoglu, "A deep learning based stock trading model with 2-d cnn trend detection," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Nov 2017, pp. 1–8.