26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

# Predicting the Price of Bitcoin Using Machine Learning

Sean McNally* Jason Roche† Simon Caton*
*School of Computing, National College of Ireland, Dublin 1, Ireland
sean.mcnally@ncirl.ie; simon.caton@ncirl.ie
†Dublin Business School, Dublin 2, Ireland
jason.roche@dbs.ie

***Abstract*—The goal of this paper is to ascertain with what accuracy the direction of Bitcoin price in USD can be predicted. The price data is sourced from the Bitcoin Price Index. The task is achieved with varying degrees of success through the implementation of a Bayesian optimised recurrent neural network (RNN) and a Long Short Term Memory (LSTM) network. The LSTM achieves the highest classification accuracy of 52% and a RMSE of 8%. The popular ARIMA model for time series forecasting is implemented as a comparison to the deep learning models. As expected, the non-linear deep learning methods outperform the ARIMA forecast which performs poorly. Finally, both deep learning models are benchmarked on both a GPU and a CPU with the training time on the GPU outperforming the CPU implementation by 67.7%.**

***Index Terms*—Bitcoin, Deep Learning, GPU, Recurrent Neural Network, Long Short Term Memory, ARIMA**

## I. Introduction

Bitcoin [1] is the worlds' most valuable cryptocurrency and is traded on over 40 exchanges worldwide accepting over 30 different currencies. It has a current market capitalization of 9 billion USD according to https://www.blockchain.info/ and sees over 250,000 transactions taking place per day. As a currency, Bitcoin offers a novel opportunity for price prediction due its relatively young age and resulting volatility, which is far greater than that of fiat currencies [2]. It is also unique in relation to traditional fiat currencies in terms of its open nature; no complete data exists regarding cash transactions or money in circulation for fiat currencies.

Prediction of mature financial markets such as the stock market has been researched at length [3], [4]. Bitcoin presents an interesting parallel to this as it is a time series prediction problem in a market still in its transient stage. Traditional time series prediction methods such as Holt-Winters exponential smoothing models rely on linear assumptions and require data that can be broken down into trend, seasonal and noise to be effective [5]. This type of methodology is more suitable for a task such as forecasting sales where seasonal effects are present. Due to the lack of seasonality in the Bitcoin market and its high volatility, these methods are not very effective for this task. Given the complexity of the task, deep learning makes for an interesting technological solution based on its performance in similar areas. The recurrent neural network (RNN) and the long short term memory (LSTM) are favoured over the traditional multilayer perceptron (MLP) due to the temporal nature of Bitcoin data.

The aim of this paper is to investigate with what accuracy the price of Bitcoin can be predicted using machine learning and compare parallelisation methods executed on multi-core and GPU environments. This paper contributes in the following manner: of approximately 653 papers published on Bitcoin [6], only 7 (at the time of writing) are related to machine learning for prediction. To facilitate a comparison to more traditional approaches in financial forecasting, an ARIMA time series model is also developed for performance comparison purposes with the neural network models.

The independent variable for this study is the closing price of Bitcoin in USD taken from the Coindesk Bitcoin Price Index. Rather than focusing on one specific exchange, we take the average price from five major Bitcoin exchanges: Bitstamp, Bitfinex, Coinbase, OkCoin and itBit. If we were to implement trades based on the signals it would be beneficial to focus on just one exchange. To assess the performance of models, we use the root mean squared error (RMSE) of the closing price and further encode the predicted price into categorical variable reflecting: price up, down or no change. This latter step allows for additional performance metrics that would be useful to a trader in the formation of a trading strategy: classification accuracy, specificity, sensitivity and precision. The dependent variables for this paper come from the Coindesk website, and Blockchain.info. In addition to the closing price, the opening price, daily high and daily low are also included as well as Blockchain data, i.e. the mining difficulty and hash rate. The features which have been engineered (considered as technical analysis indicators [7]) include two simple moving averages (SMA) and a de-noised closing price.

## II. Related Work

Research on predicting the price of Bitcoin using machine learning algorithms specifically is lacking. [8] implemented a latent source model as developed by [9] to predict the price of Bitcoin noting 89% return in 50 days with a Sharpe ratio of 4.1. There has also been work using text data from social media platforms and other sources to predict Bitcoin prices. [10] investigated sentiment analysis using support vector machines coupled with the frequency of Wikipedia views, and the network hash rate. [11] investigated the relationship between

0-7695-6380-5/18/$31.00 ©2018 IEEE
DOI 10.1109/PDP2018.2018.00060

Bitcoin price, tweets and views for Bitcoin on Google Trends. [12] implemented a similar methodology except instead of predicting Bitcoin price they predicted trading volume using Google Trends views. However, one limitation of such studies is the often small sample size, and propensity for misinformation to spread through various (social) media channels such as Twitter or on message boards such as Reddit, which artificially inflate/deflate prices [13]. In the Bitcoin exchanges liquidity is considerably limited. As a result, the market suffers from a greater risk of manipulation. For this reason, sentiment from social media is not considered further.

[14] analysed the Bitcoin Blockchain to predict the price of Bitcoin using support vector machines (SVM) and artificial neural networks (ANN) reporting price direction accuracy of 55% with a regular ANN. They concluded that there was limited predictability in Blockchain data alone. [15] also used Blockchain data, implementing SVM, Random Forests and Binomial GLM (generalised linear model) noting prediction accuracy of over 97% however without cross-validating their models limiting the generalisability of their results. Wavelets have also been utilised to predict Bitcoin prices, with [16], [17] noting positive correlations between search engine views, network hash rate and mining difficulty with Bitcoin price. Building on these findings, data from the Blockchain, namely hash rate and difficulty are included in the analysis along with data from the major exchanges provided by CoinDesk.

Predicting the price of Bitcoin can be considered analogous to other financial time series prediction tasks such as forex and stock prediction. Several bodies of research have implemented the MultiLayer Perceptron (MLP) for stock price prediction [4] [18]. However, the MLP only analyses one observation at a time [19]. In contrast, the output from each layer in a recurrent neural network (RNN) is stored in a context layer to be looped back in with the output from the next layer. In this sense, the network gains a memory of sorts as opposed to the MLP. The length of the network is known as the temporal window length. [20] notes that the temporal relationship of the series is explicitly modelled by the internal states contributing significantly to model effectiveness. [21] successfully took this approach in predicting stock returns combining a RNN with a genetic algorithm for network optimization.

Another form of RNN is the Long Short Term Memory (LSTM) network. They differ from Elman RNN in that in addition to having a memory, they can choose which data to remember and which data to forget based on the weight and importance of that feature. [22] implemented a LSTM for a time series prediction task finding that the LSTM performed as well as the RNN for this task. This type of model is implemented here also. One limitation in training both the RNN and LSTM is the significant computation required. For example, a network of 50 days is comparable to training 50 individual MLP models. Since the development of the CUDA framework by NVIDIA in 2006, the development of applications that take advantage of the extremely parallel capabilities of the GPU has grown greatly including the area of machine learning. [23] reported over three times faster training and testing of its ANN model when implemented on a GPU rather than a CPU. Similarly, [24] reported an increased speed in classification time to the magnitude of eighty times when implementing a SVM on a GPU over an alternative SVM algorithm run on a CPU. In addition, training time was nine times greater for the CPU implementation. [25] also received speeds that were forty times faster for training a when training deep neural networks for image recognition on a GPU as opposed to a CPU. Due to the apparent benefits of utilising a GPU, our RNN and LSTM models are implemented on both the CPU and GPU.

## III. Methodology

This paper follows the CRISP data mining methodology.[1] The motivation for CRISP-DM over the more traditional KDD [26] revolves around the business setting of the prediction task. The dataset Bitcoin dataset used, ranges from the 19th of August 2013 until the 19th of July 2016. A time series plot of this can be seen in Figure 1. Data from previous to August 2013 has been excluded as it no longer accurately represents the network. In addition to the Open, High, Low, Close (OHLC) data from CoinDesk, the difficulty and hash rate are taken from the Blockchain. The data was also standardised to give it a mean of 0 and standard deviation of 1. Standardisation was chosen over normalisation as it better suits the activation functions used by the deep learning models.

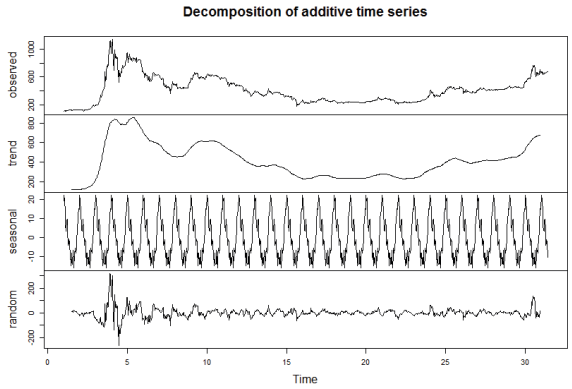


Figure 1: Decomposition of the Bitcoin Time Series Data

### A. Feature Engineering and Feature Evaluation

Feature engineering is the art of extracting useful patterns from data to make it easier for machine learning models to perform their predictions. It can be considered one of the most important parts of the data mining process in order to achieve good results in prediction tasks [27], [28]. Several papers in recent years have included indicators including the Simple Moving Average (SMA) for machine learning classification tasks [29], [30]. An example of an appropriate technical indicator is a SMA recording the average price over the previous x days, and is correspondingly included.

[1]CRISP-DM 1.0: ttps://www.te-modeling-agency.com/crisp-dm.pdf

To evaluate which features to include, Boruta (a wrapper built around the random forest classification algorithm) was used. This is an ensemble method in which classification is performed by voting of multiple classifiers. The algorithm works on a similar principle as the random forest classifier. It adds randomness to the model and collects results from the ensemble of randomised samples to evaluate attributes and provides a clear view on which attributes are important [31]. All features were deemed important to the model based on the random forest, with 5 day and 10 days (via SMA) the highest importance among the tested averages. The de-noised closing price was one of the most important variables also.

### B. Deep Learning Models

Appropriate design of deep learning models in terms of network parameters is imperative to their success. The three main options available when choosing how to select parameters for deep learning models are random search, grid search and heuristic search methods such as genetic algorithms. Manual grid search and Bayesian optimisation were utilised in this study. Grid search, implemented for the Elman RNN, is the process of selecting two hyperparamaters with a minimum and maximum for each. One then searches that feature space looking for the best performing parameters. This approach was taken for parameters which were unsuitable for Bayesian optimisation. This model was built using Keras in the Python programming language [32]. Similar to the RNN, Bayesian optimisation was chosen for selecting LTSM parameters where possible. This is a heuristic search method which works by assuming the function was sampled from a Gaussian process and maintains a posterior distribution for this function as the results of different hyperparameter selections are observed. One can then optimise the expected improvement over the best result to pick hyperparameters for the next experiment [33]. The performance of both the RNN and LSTM network are evaluated on validation data with measures to prevent overfitting. Dropout is implemented in both layers, and we automatically stop model training if its validation loss hasn't improved in 5 epochs.

## IV. Implementation

### A. RNN

The first parameter to consider was the temporal length window. As suggested by supporting literature [34] these type of networks may struggle to learn long term dependencies using gradient based optimisation. An autocorrelation function (ACF) was run for the closing price time series to assess the relationship between the current closing price and previous or future closing prices. While this is not a guarantee of predictive power for this length, it was a better choice than random choice. Closing price is correlated with a lag of up to 20 days in many cases, with isolated cases at 34, 45 and 47 days. This led the grid search for the temporal window to test from 2 to 20, 34, 45 and 47 days. To ensure a robust search, larger time periods of up to 100 days were also tested in increments of five. The most effective window temporal length was 24.

In addition to the temporal window, some hyperparameters also need tuning: Learning rate is the parameter that guides stochastic gradient descent (SGD), i.e. how the network learns. Similarly, momentum updates the learning rate to avoid the model falling into local minima (in terms of error) and attempts to move towards the global minimum of the error function [35]. We used the RMSprop optimiser to improve on SGD, as it keeps a running average of recent gradients and as a result is more robust against information loss [36]. According to Heaton [37], one hidden layer is enough to approximate the vast majority of non-linear functions. Two hidden layers were also explored and were chosen as they achieved lower validation error. Heaton also recommends for the number of hidden nodes to select between the number of input and output nodes. In this case, less than 20 nodes per layer resulted in poor performance. 50 and 100 nodes were tested with good performance. However, too many nodes can increase the chances of overfitting, and significantly increase the time needed to train the network. As 20 nodes performed sufficiently well this was chosen for the final model. An activation function, a non-linear stepwise equation that passes signals between layers, is also needed. The options explored were Tanh, ReLu, and Sigmoid. Tanh performed the best but the differences were not significant. The final parameters for selection are batch size and number of training epochs. Batch size was found to have little effect on accuracy but considerable effect on training time when using smaller batches in this case. The number of epochs tested ranged from 10 to 10000, however, too many training epochs can result in overfitting. To reduce the risk of overfitting, dropout was implemented as discussed above. Optimal dropout between 0.1 and 1 was searched for both layers with .5 dropout the optimal solution for both layers. A Keras callback method was also used to stop the training of the model if its performance on validation data did not improve after 5 epochs to prevent overfitting. Generally, the RNN converged between 20 and 40 epochs with early stopping.

### B. LSTM

In terms of temporal length, the LSTM is considerably better at learning long term dependencies. As a result, picking a long window was less detrimental for the LSTM. This process followed a similar process to the RNN in which autocorrelation lag was used as a guideline. The LSTM performed poorly on smaller window sizes. Its most effective length found was 100 days, and two hidden LSTM layers were chosen. For a time series task two layers is enough to find non-linear relationships among the data. 20 hidden nodes were also chosen for both layers as per the RNN model. The Hyperas library[2] was used to implement the Bayesian optimisation of the network parameters. The optimiser searched for the optimal model in terms of how much dropout per layer and which optimizer to use. RMSprop again performed the best for this task. In the LSTM model, activation functions weren't

[2]Hyperas: ttps://gitub.com/maxpumperla/hyperas

changed as the LSTM has a particular sequence of tanh and sigmoid activation functions for the different gates within the cell.

LSTM models converged between 50 and 100 epochs with early stopping. Similar to the RNN, batch size was found to have a greater effect on execution time than accuracy. This may be due to the relatively small size of the dataset.

### C. Model Comparison

A confusion matrix representing the ratio of true/false and positive/negative classifications is used to derive the ratings metrics. Accuracy can be defined as the total number of correctly classified predictions (price up, down, and no change). To combat inherent class imbalance (bitcoin price predominately increases) the metrics sensitivity, specificity and precision are also analysed. Sensitivity represents how good a model is at detecting positives. Specificity represent how good the model is at avoiding false alarms. Finally, precision represents how many positively classified predictions were relevant. Root Mean Square Error (RMSE) is used to evaluate and compare the regression accuracy. To instrument the evaluation of models, a 80/20 holdout validation strategy is used.

In order to facilitate a comparison of the deep learning methods to more traditional methods we built (and optimised) an ARIMA model, as they have been extensively used in price prediction problems (e.g. [38], [39]). The ARIMA forecast was created by splitting the data into 5 periods and then predicting 30 days into the future. The data was differenced before being fit with several ARIMA models. The best fit was found by auto.arima from the R forecast package.

## V. Evaluation

As can be seen in Table I, LSTM achieved the highest accuracy while the RNN achieved the lowest RMSE. The ARIMA prediction performed poorly in terms of accuracy and RMSE. Upon analysis of the ARIMA forecast, it predicted the price would gradually rise each day. There were no false positives from the model. One reason for this may be due to the class imbalance in predictive portion of the ARIMA forecast (the price tends to always increase). This contributed to the specificity and precision being so high (specificity, precision= 100%). This does not necessarily suggest good overall performance, but rather that it does a decent job at identifying price direction change(s).

Table I: Model Results

| Model | Temporal Length | Sensitivity | Specificity | Precision | Accuracy | RMSE |
|---|---|---|---|---|---|---|
| LSTM | 100 | 37% | 61.30% | 35.50% | 52.78% | 6.87% |
| RNN | 20 | 40.40% | 56.65% | 39.08% | 50.25% | 5.45% |
| ARIMA | 170 | 14.7% | 100% | 100% | 50.05% | 53.74% |

Based on the results on the validation data it is apparent that all models struggled to effectively learn from the data. On training data, the model reduced error to below 1%. On validation data the LSTM achieved error of 8.07% while the RNN achieved error of 7.15%. The 50.25% and 52.78% accuracy achieved by the neural network models is a marginal improvement over the odds one has in a binary classification task (price up vs. down), i.e. 50%. The RNN was effectively of no use when using a temporal length over 50 days. In contrast, the LSTM performed better in the 50 to 100 day range with 100 days producing the best performance.

Table II: Performance Comparison

| Model | Epochs | CPU | GPU |
|---|---|---|---|
| RNN | 50 | 56.71s | 33.15s |
| LSTM | 50 | 59.71s | 38.85s |
| RNN | 500 | 462.31s | 258.1s |
| LSTM | 500 | 1505s | 888.34s |
| RNN | 1000 | 918.03s | 613.21s |
| LSTM | 1000 | 3001.69s | 1746.87s |

To evaluate the training of the models, Table II compares different models. The CPU utilised was a Intel Core i7 2.6GHz. The GPU used was a NVIDIA GeForce 940M 2GB. Both were running on Ubuntu 14.04 LTS installed on a SSD. For comparability the same batch size and temporal length of 50 were chosen for both the RNN and LSTM. The GPU considerably outperformed the CPU. In terms of overall training time for both networks, the GPU trained 67.7% faster than the CPU. The RNN trained 58.8% faster on the GPU while the LSTM trained 70.7% faster on the GPU. From monitoring performance in Glances, the CPU spread the algorithm out over 7 threads. The GPU has 384 CUDA cores which provide it with greater parallelism. These models were relatively quite small in terms of data with two layers. For deeper models with more layers or bigger datasets the benefits of implementing on a GPU will be even greater.

That the LSTM takes longer to train than the RNN with the same network parameters, may be due to the increased number of activation functions, and thus an increased number of equations to be performed by the LSTM. Due to the increased computation, this raises the question of the value of using an LSTM over an RNN. In financial market prediction small margins can make all the difference. As a result of this the use of an LSTM is justified. In other areas the slight improvement in terms of performance isn't justifiable for the increase in computation.

## VI. Conclusion and Future Work

Deep learning models such as the RNN and LSTM are evidently effective for Bitcoin prediction with the LSTM more capable for recognising longer-term dependencies. However, a high variance task of this nature makes it difficult to transpire this into impressive validation results. As a result it remains a difficult task. There is a fine line between overfitting a model and preventing it from learning sufficiently. Dropout is a valuable feature to assist in improving this. However, despite using Bayesian optimisation to optimize the selection of dropout it still couldn't guarantee good validation results. Despite the metrics of sensitivity, specificity and precision indicating good performance, the actual performance of the ARIMA forecast based on error was significantly worse than the neural network models. The LSTM outperformed the RNN

marginally, but not significantly. However, the LSTM takes considerably longer to train.

The performance benefits gained from the parallelisation of machine learning algorithms on a GPU are evident with a 70.7% performance improvement for training the LSTM model. Looking at the task from purely a classification perspective it may be possible to achieve better results. One limitation of the research is that the model has not been implemented in a practical or real time setting for predicting into the future as opposed to learning what has already happened. In addition, the ability to predict using streaming data should improve the model. Sliding window validation is an approach not implemented here but this may be explored as future work. One problem that will arise is that the data is inherently shrouded in noise.

In terms of the dataset, based on an analysis of the weights of the model the difficulty and hash rate variables could be considered for pruning. Deep learning models require a significant amount to data to learn effectively. If the granularity of data was changed to per minute this would provide 512,640 data points in a year. Data of this nature is not available for the past but is currently being gathered from CoinDesk on a daily basis for future use. Finally, parallelisation of algorithms is not limited to GPU devices. Field Programmable Gate Arrays (FPGA) are an interesting alternative to GPU devices in terms of parallelisation and machine learning models have been shown to perform better on FPGA than on a GPU [40].

## REFERENCES


[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] M. Brière, K. Oosterlinck, and A. Szafarz, "Virtual currency, tangible return: Portfolio diversification with bitcoins," *Tangible Return: Portfolio Diversification with Bitcoins (September 12, 2013)*, 2013.

[3] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996.

[4] H. White, "Economic prediction using neural networks: The case of ibm daily stock returns," in *Neural Networks, 1988., IEEE International Conference on*. IEEE, 1988, pp. 451–458.

[5] C. Chatfield and M. Yar, "Holt-winters forecasting: some practical issues," *The Statistician*, pp. 129–140, 1988.

[6] B. Scott, "Bitcoin academic paper database," *suitpossum blog*, 2016.

[7] M. D. Rechenthin, "Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction," 2014.

[8] D. Shah and K. Zhang, "Bayesian regression and bitcoin," in *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*. IEEE, 2014, pp. 409–414.

[9] G. H. Chen, S. Nikolov, and D. Shah, "A latent source model for nonparametric time series classification," in *Advances in Neural Information Processing Systems*, 2013, pp. 1088–1096.

[10] I. Georgoula, D. Pournarakis, C. Bilanakos, D. N. Sotiropoulos, and G. M. Giaglis, "Using time-series and sentiment analysis to detect the determinants of bitcoin prices," *Available at SSRN 2607167*, 2015.

[11] M. Matta, I. Lunesu, and M. Marchesi, "Bitcoin spread prediction using social and web search media," *Proceedings of DeCAT*, 2015.

[12] ——, "The predictor impact of web search media on bitcoin trading volumes."

[13] B. Gu, P. Konana, A. Liu, B. Rajagopalan, and J. Ghosh, "Identifying information in stock message boards and its implications for stock market efficiency," in *Workshop on Information Systems and Economics, Los Angeles, CA*, 2006.

[14] A. Greaves and B. Au, "Using the bitcoin transaction graph to predict the price of bitcoin," 2015.

[15] I. Madan, S. Saluja, and A. Zhao, "Automated bitcoin trading via machine learning algorithms," 2015.

[16] R. Delfin Vidal, "The fractal nature of bitcoin: Evidence from wavelet power spectra," *The Fractal Nature of Bitcoin: Evidence from Wavelet Power Spectra (December 4, 2014)*, 2014.

[17] L. Kristoufek, "What are the main drivers of the bitcoin price? evidence from wavelet coherence analysis," *PloS one*, vol. 10, no. 4, p. e0123923, 2015.

[18] Y. Yoon and G. Swales, "Predicting stock price performance: A neural network approach," in *System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on*, vol. 4. IEEE, 1991, pp. 156–162.

[19] T. Koskela, M. Lehtokangas, J. Saarinen, and K. Kaski, "Time series prediction with multilayer perceptron, fir and elman neural networks," in *Proceedings of the World Congress on Neural Networks*. Citeseer, 1996, pp. 491–496.

[20] C. L. Giles, S. Lawrence, and A. C. Tsoi, "Noisy time series prediction using recurrent neural networks and grammatical inference," *Machine learning*, vol. 44, no. 1-2, pp. 161–183, 2001.

[21] A. M. Rather, A. Agarwal, and V. Sastry, "Recurrent neural network and a hybrid model for prediction of stock returns," *Expert Systems with Applications*, vol. 42, no. 6, pp. 3234–3241, 2015.

[22] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying lstm to time series predictable through time-window approaches," pp. 669–676, 2001.

[23] D. Steinkrau, P. Y. Simard, and I. Buck, "Using gpus for machine learning algorithms," in *Proceedings of the Eighth International Conference on Document Analysis and Recognition*. IEEE Computer Society, 2005, pp. 1115–1119.

[24] B. Catanzaro, N. Sundaram, and K. Keutzer, "Fast support vector machine training and classification on graphics processors," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 104–111.

[25] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[26] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The kdd process for extracting useful knowledge from volumes of data," *Communications of the ACM*, vol. 39, no. 11, pp. 27–34, 1996.

[27] T. Dettmers, "Deep learning in a nutshell: Core concepts," *NVIDIA Devblogs*, 2015.

[28] D. K. Wind, "Concepts in predictive machine learning," in *Maters Thesis*, 2014.

[29] Y. Wang, "Stock price direction prediction by directly using prices data: an empirical study on the kospi and hsi," *International Journal of Business Intelligence and Data Mining*, vol. 9, no. 2, pp. 145–160, 2014.

[30] S. Lauren and S. D. Harlili, "Stock trend prediction using simple moving average supported by news classification," in *Advanced Informatics: Concept, Theory and Application (ICAICTA), 2014 International Conference of*. IEEE, 2014, pp. 135–139.

[31] M. B. Kursa, W. R. Rudnicki *et al.*, "Feature selection with the boruta package," 2010.

[32] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[33] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[34] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[35] L. Yu, S. Wang, and K. K. Lai, "A novel nonlinear ensemble forecasting model incorporating glar and ann for foreign exchange rates," *Computers And Operations Research*, vol. 32, no. 10, pp. 2523–2541, 2005.

[36] G. Hinton, N. Srivastava, and K. Swersky, "Lecture 6a overview of mini–batch gradient descent," *Coursera Lecture slides https://class. coursera. org/neuralnets-2012-001/lecture,[Online*.

[37] J. Heaton, "Introduction to neural network for java, heaton research," *Inc. St. Louis*, 2005.

[38] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "Arima models to predict next-day electricity prices," *IEEE transactions on power systems*, vol. 18, no. 3, pp. 1014–1020, 2003.

[39] P.-F. Pai and C.-S. Lin, "A hybrid arima and support vector machines model in stock price forecasting," *Omega*, vol. 33, no. 6, pp. 497–505, 2005.

[40] M. Papadonikolakis, C.-S. Bouganis, and G. Constantinides, "Performance comparison of gpu and fpga architectures for the svm training problem," in *Field-Programmable Technology, 2009. FPT 2009. International Conference on*. IEEE, 2009, pp. 388–391.