National College of Ireland

# Predicting the price of Bitcoin using Machine Learning

MSc Reseach Project
Data Analytics


Sean McNally
x15021581

School of Computing
National College of Ireland


Supervisor: Dr. Jason Roche

**National College of Ireland**
**Project Submission Sheet – 2015/2016**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Sean McNally |
| **Student ID:** | x15021581 |
| **Programme:** | Data Analytics |
| **Year:** | 2016 |
| **Module:** | MSc Reseach Project |
| **Lecturer:** | Dr. Jason Roche |
| **Submission Due Date:** | 22/08/2016 |
| **Project Title:** | Predicting the price of Bitcoin using Machine Learning |
| **Word Count:** | XXX |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

| | |
|---|---|
| **Signature:** | |
| **Date:** | 9th September 2016 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS:**

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Predicting the price of Bitcoin using Machine Learning

Sean McNally
x15021581
MSc Reseach Project in Data Analytics

9th September 2016

**Abstract**

This research is concerned with predicting the price of Bitcoin using machine learning. The goal is to ascertain with what accuracy can 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 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. Wavelets are explored as part of the time series narrative but not implemented for prediction purposes. 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%.

## 1 Introduction

Time series prediction is not a new phenomenon. Prediction of mature financial markets such as the stock market has been researched at length (1)(2). Bitcoin presents an interesting parallel to this as it is a time series prediction problem in a market still in its transient stage. As a result, there is high volatility in the market (3) and this provides an opportunity in terms of prediction. In addition, Bitcoin is the leading cryptocurrency in the world with adoption growing consistently over time. Due to the open nature of Bitcoin it also poses another paradigm as opposed to traditional financial markets. It operates on a decentralised, peer-to-peer and trustless system in which all transactions are posted to an open ledger called the Blockchain. This type of transparency is unheard of in other financial markets.

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 (4). 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. Tasks such as natural language processing which are also sequential in nature and have shown promising results (5). This type of task uses data of a sequential nature and as a result is similar to a price prediction task. The recurrent neural network (RNN) and the long short term memory (LSTM) flavour of artificial neural networks are favoured over the traditional multilayer perceptron (MLP) due to the temporal nature of the more advanced algorithms (6).

The aim of this research is to ascertain with what accuracy can the price of Bitcoin be predicted using machine learning. Section one addresses the project specification which includes the research question, sub research questions, the purpose of the study and the research variables. A brief overview of Bitcoin, machine learning and time series analysis concludes section one. Section two examines related work in the area of both Bitcoin price prediction and other financial time series prediction. Literature on using machine learning to predict Bitcoin price is limited. Out of approximately 653 papers published on Bitcoin (7) only 7 have related to machine learning for prediction. As a result, literature relating to other financial time series prediction using deep learning is also assessed as these tasks can be considered analogous. Section 3 focuses on the design and implementation of the solution to the research question. Section 4 analyses the results of the implemented solution. A forecast using a traditional ARIMA time series model is also developed for performance comparison purposes with the neural network models. Section 5 concludes the paper with reference to future work in the area.

## 1.1 Project Specification

### 1.1.1 Research Question

Research question: *With what accuracy can the direction of the price of Bitcoin be predicted using machine learning?*

Sub Research question: *What magnitude of performance improvement can be achieved from parallelisation of algorithms on a GPU compared to a CPU?*

### 1.1.2 Purpose

The purpose of this study is to find out with what accuracy the direction of the price of Bitcoin can be predicted using machine learning methods. This is fundamentally a time series prediction problem. While much research exists surrounding the use of different machine learning techniques for time series prediction, research in this area relating specifically to Bitcoin is lacking. In addition, Bitcoin as a currency is in a transient stage and as a result is considerably more volatile than other currencies such as the USD. Interestingly, it is the top performing currency four out of the last five years[1]. Thus, its prediction offers great potential and this provides motivation for research in the area. As evidenced by an analysis of the existing literature, running machine learning algorithms on a GPU as opposed to a CPU can offer significant performance improvements. This is explored by benchmarking the training of the RNN and LSTM network using both the GPU and CPU. This provides an answer to the sub research question. Finally, in analysing the chosen dependent variables, each variables importance is assessed using a random forest algorithm. This body of research builds on existing literature in the area which is assessed in section 2.

In addition, the ability to predict the direction of the price of an asset such as

[1]The Economist: `https://www.economist.com/`

Bitcoin offers the opportunity for profit to be made by trading the asset. To implement a full trading strategy based on the results of the models is worthy of a dissertation in itself and as a result this paper will focus solely on the accuracy at which the price direction can be predicted. In basic terms, the model would initiate a short position if the price was predicted to go up and a long position if the price was predicted to go down. Several Bitcoin exchanges offer margin trading accounts to facilitate this [2]. The profitability of this strategy would be based not only on the accuracy of the model, but also on the size of the positions taken. This is outside the scope of this research but could be addressed in future work.

## 1.2 Research Variables

The independent variable for this study is the closing price of Bitcoin in US Dollars taken from the Coindesk Bitcoin Price Index. Rather than focusing on one specific exchange this price index takes the average prices from five major Bitcoin exchanges; Bitstamp, Bitfinex, Coinbase, OkCoin and itBit. If one were to implement trades based on the signals it would be beneficial to focus on one exchange. However, the average price is more suitable for this research as some exchanges suffer isolated drops in price from internal issues such as Bitfinex who were hacked recently[3]. As a result, there is less noise in the averaged dataset. The closing price is chosen over a three-class dummy classification variable representing price going up, down or staying the same for the following reason; the use of a regression model over a classification model offers further model comparison potential through the capture of the root mean squared error (RMSE) of the models. Classifications are then made based on the prediction of the regression model e.g. price up, price down or no change. Additional performance metrics include accuracy, specificity, sensitivity and precision. This is discussed in the implementation section.

The dependent variables are taken from the Coindesk website, Blockchain.info and from the process of feature engineering. In addition to the closing price, the opening price, daily high and daily low are also included. Data taken from the Blockchain includes mining difficulty and hash rate. The features which have been engineered are considered technical analysis indicators (8) and include two simple moving averages (SMA) and a de-noised closing price. The rationale behind the selection of variables is discussed in chapter 2.

## 1.3 Bitcoin

Bitcoin is the worlds most valuable cryptocurrency introduced following the release of a whitepaper published in 2008 under the pseudo name Satoshi Nakamoto (9). The currency is built on a decentralised, peer-to-peer network with the creation of money and transaction management carried out by the members of the network. The result of this is no central authority controls Bitcoin. All Bitcoin transactions are posted in blocks to an open ledger known as the Blockchain to be verified by miners using cryptographic proof. This verification takes place in a trustless system with no intermediary required to pass the funds from sender to receiver. Bitcoin offers a novel opportunity for prediction due its relatively young age and resulting volatility. In addition, it is unique in relation to traditional fiat currencies in terms of its open nature. In comparison, no complete data exists regarding cash transactions or money in circulation of fiat currencies. The well-known efficient market hypothesis (10) suggests the price of assets such as currencies reflect all available information, and as a result trade at their

[2]Poloniex Exchange: `https://www.poloniex.com/`
[3]Bitfinex Exchange: `https://www.bitfinex.com/`

fair value. Although there is an abundance of data available relating to Bitcoin and its network, the author argues that not all market participants will utilise all this information effectively and as a result it may not be reflected in the price. This paper aims to take advantage of this assumption through various machine learning methods.

Bitcoin is traded on over 40 exchanges worldwide accepting over 30 different currencies and has a current market capitalization of 9 billion dollars [4]. Interest in Bitcoin has grown significantly with over 250,000 transactions now taking place per day. In addition to the regular use of Bitcoin by private individuals, its lack of correlation with other assets have made it an attractive hedging option to investors. Some research has found that the price volatility of Bitcoin is far greater than that of fiat currencies(3). This offers significant potential in comparison to mature financial markets.

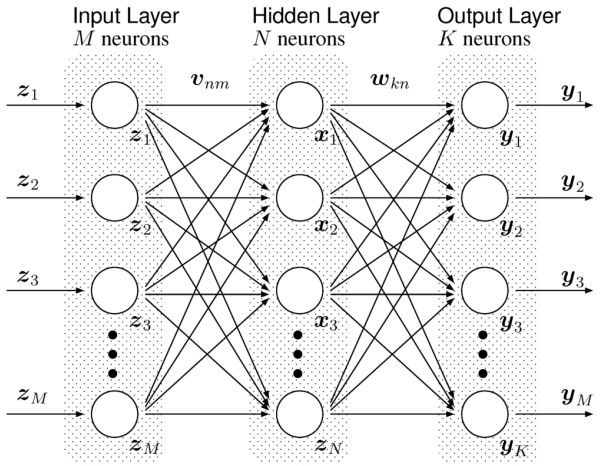
## 1.4 Machine Learning

Data mining can be defined as the extraction of implicit, previously unknown and potentially useful information from data. Machine learning provides the technical basis for data mining (11). A dataset is comprised of observations which are known as instances which contain one or more variables known as attributes. Broadly speaking, machine learning can be split into two categories. Supervised learning involves the modelling of datasets with labelled instances. Each instance can be represented as x and y, with x a set of independent predictor attributes and y the dependent target attribute. The target attribute can be continuous or discrete however this has an effect on the model. If the target variable is continuous then a regression model is used and if the target variable is discrete then a classification model is used (8). Examples of supervised methods include neural networks and support vector machines.

Unsupervised learning involves the modelling of datasets with no known outcome or attribute. The purpose of these techniques is to group similar data into clusters or groups. The purpose of this research is to predict the direction of the price of Bitcoin. As this is a task with a known target it is a supervised machine learning task although some pre-processing can take advantage of unsupervised learning methods. The supervised algorithms explored include Wavelets and the wavelet discrete transform, several type of artificial neural networks including the Multi-Layer perceptron (MLP), Elman Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM). In terms of pre-processing, random forests were used for feature selection while Bayesian optimisation was performed to optimize some the parameters of the LSTM.

### 1.4.1 Multilayer Perceptron

**Figure 1: Multilayer Perceptron**




Simple feed forward neural networks are known as multilayer perceptrons and they form the basis for other neural network models. In terms of neural network terminology, examples fed to the model are known as inputs and the predictions are known as outputs. Each modular sub function is a layer. A model consists of an in-

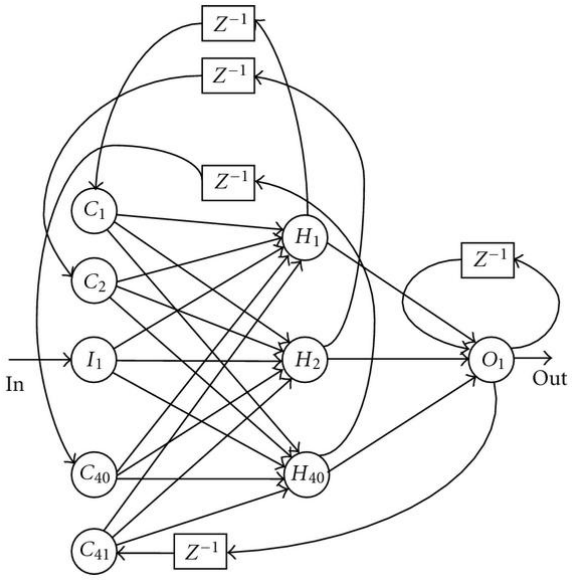
[4]Blockchain: `https://www.blockchain.info/`

put and output layer, with layers between these known as hidden layers. Each output of one of these layers is a unit which can be considered analogous to a neuron in the brain. The connections between these units is known as a weight which is analogous to a synapse in the brain. The weights define the function of the model as they are the parameter that is adjusted when training a model. Non-linear element wise operators such as the hyperbolic tangent or rectified linear unit (ReLu) are applied to the input to perform the non-linear transformations between layers. An example of an MLP can be seen in figure 1. One limitation of the MLP and similarly the RNN is that they are affected by the vanishing gradient problem. This issue is that as layers and time steps of the network relate to each other through multiplication, derivatives are susceptible to exploding or vanishing gradients. Vanishing gradients are more of a concern as they can become too small for the network to learn whereas exploding gradients can be limited using regularisation. Another limitation of the MLP is that its signals only pass forward in the network in a static nature. As a result, it does not recognise the temporal element of a time series task effectively as its memory can be considered frozen in time [5]. One can consider an MLP to treat all input as a bucket of objects with no order in time. As a result, the same weights are applied to all incoming data which is a naive approach. The recurrent neural network, sometimes known as a dynamic neural network addresses some of these limitations.

## 1.5 Recurrent Neural Network

The recurrent neural network (RNN) was first developed by Elman (6). The RNN is structured similarly to the MLP, with the exception that signals can flow both forward and backwards in an iterative manner. To facilitate this another layer known as the context layer is added. In addition to passing input between layers, the output of each layer is fed to the context layer to be fed into the next layer with the next input. In this context, the state is overwritten at each timestep. This offers the benefit of allowing the network to assign particular weights to events that occur in a series rather than the same weights to all input as with the MLP. This results in a dynamic network. The length of the temporal window in a sense is the length of your networks memory. As is discussed in the related work section, it is an appropriate technique for a time series prediction task (5) (12). While this addresses the temporal issue faced with a time series task, vanishing gradient can still be an issue. In addition, some research has found that while RNN are capable of handling long-term dependencies, in practice they often fail to learn due to the difficulties between gradient descent and long term dependencies (13) (14). An example of an RNN can be seen below in figure 2. Note the context layer is represented as z-i.

**Figure 2: Recurrent Neural Network**




[5] Deeplearning4j: `http://deeplearning4j.org/lstm`

### 1.5.1 Long Short Term Memory

Long short term memory (LSTM) units address both of these issues. Developed by Hochreiter et al. (15), they allow the preservation of the weights that are forward and back-propagated through layers. This is in contrast to the Elman RNN in which the state gets overwritten at each step. They also allow the network to continue to learn over many time steps by maintaining a more constant error. This allows the network to learn long term dependencies. A LSTM cell contains forget and remember gates which allow the cell to decide what information to block or pass based on its strength and importance. As a result, weak signals can be blocked which prevents vanishing gradient. A comparison of a regular RNN and LSTM can be seen below in figure 3. Note the addition of the gates on the right and left of the LSTM block.

**Figure 3: Long Short Term Memory**[6]

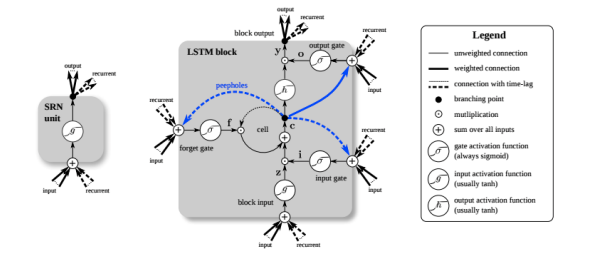
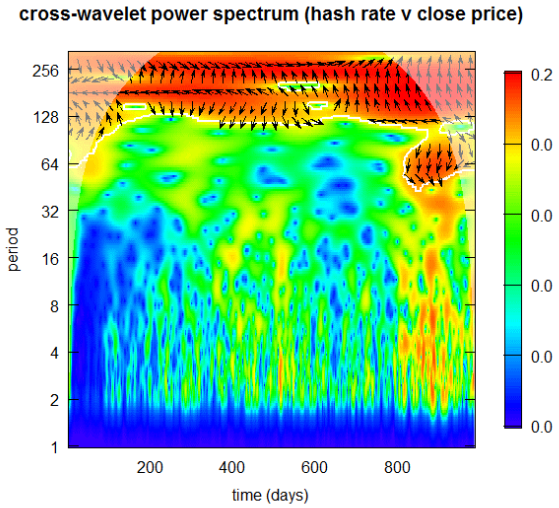



*Figure 1.* Detailed schematic of the Simple Recurrent Network (SRN) unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network.

## 1.6 Time Series Analysis:

Based on the abundance of literature on the topic, the ARIMA model is a popular approach to time series forecasting (16)(17). However, these models rely on linear assumptions regarding the data. Due to the highly non-linear nature of the Bitcoin price it was not expected to perform well. The data was differenced to make it stationary to meet the necessary assumptions. As a result, the ARIMA model will not form an integral part of this research. However, a forecast was made using an ARIMA model for comparison purposes with the final neural network models in the evaluation section. Wavelets and the Wavelet Discrete Transform (WDT) were also considered in the analysis of the time series analysis domain. Wavelets are a mathematical function useful for analysing and breaking down signals (18). A time series can be considered in the same regard as a signal. The intention was to investigate using Wavelets as an input to the neural network model. This was not possible due to time constraints but could be addressed in future work. However, Wavelet coherence analysis was used to analyse correlation between variables in the dataset on a temporal scale. This is a useful exploratory analysis exercise in analysing relationship between variables over time (19). An example of one of the Wavelet Coherence analysis spectrum graphs can be seen below in figure 4. This graph represents the cross correlation between Bitcoin and the Hash rate over time. From the graph it is apparent that Bitcoin closing price is correlated with the network hash rate on a long term horizon.

**Figure 4: Wavelet Power Spectrum**




[6]Deeplearning4j: `http://deeplearning4j.org/lstm`

## 2 Related Work

Research on predicting the price of Bitcoin using machine learning algorithms specifically is lacking. Shah et al. (20) implemented a latent source model as developed by Chen et al. (17) to predict the price of Bitcoin. The model received an impressive 89 percent return in 50 days with a Sharpe ratio of 4.1. The Sharpe ratio examines performance of an investment while adjusting for its risk. One point of note from this study is that the period selected for testing by the user enjoyed growth of 33 percent using a buy and hold strategy alone. Attempts were made to re-create this study independently which were unsuccessful . One reason provided for this was that the original authors hand selected 20 patterns observed in their clusters which were similar to those they had seen in trading books i.e. the head and shoulders. This highlights the danger of cherry-picking data to obtain good results.

Geourgoula et al. (21) investigated the determinants of the price of Bitcoin while also implementing sentiment analysis using support vector machines. The author found that the frequency of Wikipedia views and the network hash rate had a positive correlation with the price of Bitcoin. Sentiment has also been utilised as a predictor of Bitcoin in other research. Matta et al. (22) investigated the relationship between Bitcoin price, tweets and views for Bitcoin on Google Trends. The author found a weak to moderate correlation between Bitcoin price and both Google Trends views and positive tweets on Twitter. The author found this to be proof that they can be used as predictors. However, one limitation of this study is that the sample size is only 60 days. Sentiment was considered as a variable. However, with a variable of this nature the question of trust arises. Misinformation can be spread through various social media channels such as Twitter or on message boards such as Reddit. In what is known as a pump-and-dump scheme, an investor looking to take advantage could spread misinformation through the channels discussed in order to buy at an artificially deflated price or sell at an artificially inflated price (23). 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. Another study by the same authors (24) implemented a similar methodology except instead of predicting Bitcoin price they tried to predict trading volume. This study found Google Trends views to be strongly correlated with Bitcoin price. This data sample covered a period of just under one year. This data source was considered for implementation. However, as only the past years worth of data is available from Google this variable was not deemed suitable. Some papers have utilised Wavelets to find similar results (25)(19). Kristoufek found a positive correlation between search engine views, network hash rate and mining difficulty with the price of Bitcoin in the long run using Wavelet coherence analysis of Bitcoin price. Wavelets are useful in exploring cross correlation between time series as they have a temporal dimension. As a result, one can see correlation between variables at specific points in time.

Greaves et al. (26) analysed the Bitcoin Blockchain to predict the price of Bitcoin using SVM and ANN. The author reported price direction accuracy of 55 percent with a regular ANN. They concluded that there was limited predictability in Blockchain data alone as price is technically dictated by exchanges whose behavior lies outside of the realm of the Blockchain. 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. Similarly, Madan et al. (27) attempted to predict Bitcoins price using Blockchain data. They implemented SVM, Random Forests and Bino-

mial GLM using data from the Blockchain. The author reported accuracy of over 97 percent. One limitation of this study is the results were not cross-validated. As a result they may have overfit the data and one cant be sure if the model will generalize.

In terms of a machine learning task, predicting the price of Bitcoin can be considered analogous to other financial time series prediction tasks such as forex and stock prediction. The idea for using ANN for such a task is not a new notion. Since the discovery of the back propagation algorithm (28) much research was undertaken in the area which concluded that ANN are suitable for modelling and forecasting nonlinear time series (29)(30). Several bodies of research implemented the MultiLayer Perceptron for stock price prediction (2)(31). White found limited value in their IBM stock price prediction MLP due to a lack of data. This study utilized a trial and error network parameter search process. One limitation of this approach is that there is no guarantee that a global maximum has been found. Bergstra et al. (32) found this random search process to be more effective than grid search. This is due to random searches being able to find as good or better models within a small fraction of the computation time. What the author doesnt recognize is that it is difficult to know when an optimal model has been found. For this reason, grid search is favoured over random search in this research where Bayesian optimization is not suitable (33). This approach uses Bayesian regularisation to search the feature space for an optimal solution. The fitness is calculated for each model and the fittest model is kept. An example of a Bayesian optimiser is the Python library Hyperopt[7]. This approach is also implemented. This should reduce the chances of overfitting the model which is a common problem in the literature. Often models perform well on training data but not on unseen data.

Another approach to reducing the risk of overfitting is dropout regularisation as outlined by Wager et al. (34) This approach is considered a noising scheme that controls for overfitting by corrupting training data. This increases the generalisability of the model and as a result should perform better on unseen data. Another factor to consider when trying to predict time series data is that the time when certain features appear may contain important data. One limitation of the MLP is it can only learn an input to output mapping that is static, and thus it has no notion of order in time. As a result, the only input it considers is the current example it has been exposed to (35). In contrast, the output from each layer in a RNN is stored in a context layer to be looped back in with the output from the next layer. In this sense one may consider that the network gains a memory of sort as opposed to the MLP. The length of the network is known as the temporal window length. Giles et al. (36) found that the fact that the temporal relationship of the series is explicitly modelled by the internal states can contribute significantly to the models effectiveness. Rather et al (37) took this approach in predicting stock returns. They incorporated a RNN in a hybrid model with a genetic algorithm for optimization of weight selection. In this case the author reported successful results and credited this to optimal network parameter selection. Optimisers such as RMSprop are suitable for recurrent neural networks in terms of weight selection and updates.

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. Gers et al. (12) implemented a LSTM for a time series prediction task. The author found that the LSTM performed as well as the RNN for this task. This type of

[7]Hyperopt: https://github.com/hyperopt/hyperopt

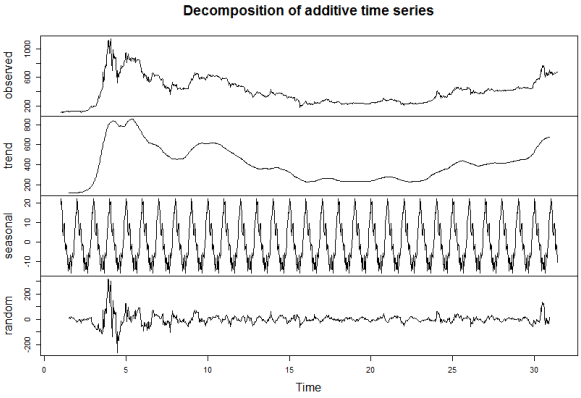
model is implemented here also. One limitation in training both the RNN and LSTM is the significant computation required. For example, a network of 50 days is comparable to training 50 individual MLP models. Since the development of the CUDA framework by NVIDIA in 2006, the development of applications that take advantage of the extremely parallel capabilities of the GPU has grown greatly including the area of machine learning. Steinkrau et al. (38) reported over three times faster training and testing of its ANN model when implemented on a GPU rather than a CPU. Cantanzaro et al. (39) 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 ran on a CPU. In addition, training time was nine times greater for the CPU implementation. Ciresan et al. (40) 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, the LSTM model is implemented on both the CPU and GPU. The performance of both implementations is analysed in the results section.

# 3 Methodology

This body of research follows the CRISP data mining methodology [8]. This is an incremental approach made of up four levels of tasks which are completed in an iterative manner. As a result, analogously it also reflects an Agile methodology (41).

The dataset ranges from the 19th of August 2013 until the 19th of July 2016. A time series plot of this can be seen in figure 5 below in the observed window. Data from previous to August 2013 was excluded as the author feels that due to the immaturity of the network, its not an accurate representation of the network at present. In addition to the Open, High, Low, Close (OHLC) data from CoinDesk and the difficulty and hash rate taken from the Blockchain, features were engineered to provide the dataset with more predictive data to learn from. Data was standardised in R using the scale function. This transforms the dataset to give it a mean = 0 and SD = 1. Before the dataset was standardised and given as input to the neural network it performed poorly. Standardisation was chosen over normalising the data for example between 0 and 1 as this methodology is robust to most activation functions.

**Figure 5 Decomposition of Time Series**




## 3.1 Feature Engineering

Feature engineering is the art of extracting useful patterns from data to make it easier for machine learning models to perform its prediction. It can be considered one of the most important skills to achieve good results for prediction tasks (42). Wind (43) investigated the behaviour of consistent top performers in Kaggle data mining competitions. The findings were that feature engineering is often the most important part. It is quite a subjective process requiring domain knowledge to be effective. It is also considered an art. Engineered features should represent what one is trying to teach the network.

[8] CRISP-DM 1.0: https://www.the-modeling-agency.com/crisp-dm.pdf

However, chosen features must be evaluated to ensure they improve your datasets predictability. The addition of technical analysis indicators offers the benefit of describing and quantifying trends which may exist in the price. The rationale for this is quantitative technical indicators provide a numerical description of previous trends in the data which can be used as an attribute in a machine learning algorithm for both regression and classification. Several papers in recent years have included indicators including the Simple Moving Average (SMA) for machine learning classification tasks (44) (45). An example of one such technical indicator is a simple moving average (SMA). This records the average price over the previous x number of days.

$$SMA = \frac{p1 + p2... + px}{n}$$

Fortunately, not much feature engineering is required for deep learning models as the hidden layers learn non-linear hierarchical features and in the case of the LSTM to detect very long term dependencies in sequential data (42). The reason one would still be required to engineer features is that there is no way of knowing what non-linear features the model has learned. When feature engineering, one may explicitly choose features which are thought to be important. It is accepted that feature engineering is a subjective process highly individual to the task at hand (43). As a result, utilising domain knowledge is necessary. The rationale of selecting a SMA is that it can allow a model to more easily recognise trends by smoothing the data. It is a popular technical indicator. In addition to the SMA a de-noised closing price was created in R. This involved decomposing the time series into trend, seasonal and noise components as can be seen in figure 5 above. The importance of features can be evaluated using various feature evaluation methods.

## 3.2 Feature Evaluation

Features must be evaluated once selected. The reason for this is dealing with too large of a feature set will considerably increase training time. In addition, machine learning algorithms can suffer from decreased accuracy if the number of variables is significantly higher than the optimal number(46). Several methods of feature evaluation exist including filter based selection and wrapper based selection. Filter based selectors filter features based on a particular statistical property of the feature e.g. correlation. Wrapper based methods perform a heuristic search of solutions to a classifier.

The Boruta algorithm in R is one such wrapped based methods. This algorithm is a wrapper built around the random forest classification algorithm. This is an ensemble classification 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. This extra randomness provides you with a clear view on which attributes are important (47). All features were deemed important to the model based on the random forest, with 5 day and 10 days the highest importance among the tested averages. The de-noised closing price was one of the most important variables also.

The dimensionality reduction technique of principal component analysis (PCA) was also explored. The result was four principal groups in which all attributes belonged to. The results of the PCA was not included in the final model as computation wasnt an issue and the original data performed reasonably well.

## 3.3 RNN

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. As mentioned in the related work section manual grid search and Bayesian optimisation are 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 (48).

### 3.4 LSTM

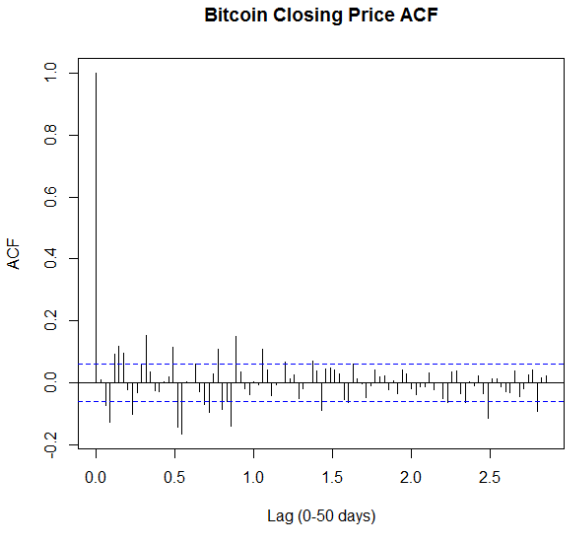
Similar to the RNN, Bayesian optimisation was chosen for selecting parameters for this model where possible. This is a heuristic search method which works by assuming the function was sampled from a Gaussian process and maintains a posterior distribution for this function as the results of different hyperparameter selections are observed. One can then optimise the expected improvement over the best result to pick hyperparameters for the next experiment (49). The performance of both the RNN and LSTM network are evaluated on validation data with significant overfitting measures in place. Dropout is implemented in both layers. In addition, an early stopper is programmed into the model to prevent overfitting. This stops the model if its validation loss doesnt improve for 5 epochs.

## 4 Implementation

### 4.1 RNN

The first parameter for selection was the temporal length window. As suggested by supporting literature (14) these type of networks may struggle to learn long term dependencies using gradient based optimisation. However, this could not be assumed. An autocorrelation function (ACF) was ran for the closing price time series. This assesses 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. The ACF for closing price can be seen below in figure 6. 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 days and says 34, 45 and 47. 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.

**Figure 6: ACF**




Learning rate is the parameter that guides your stochastic gradient descent (SGD) i.e. how your network learns. Momentum updates your learning rate to avoid getting stuck in local minima and attempt to move to global minimum of the function (50). However, several optimisers are available to improve this process. The optimser RMSprop improves on SGD with momentum as it keeps a running average and as a result is more robust to information

loss (51). RMSprop keeps a running average of its recent gradient magnitudes. The next gradient is divided by the average so that gradient values are loosely normalised. The number of hidden layers and hidden nodes was the next parameter choice. Other optimisers such as Adagrad and Adadelta are available but the Keras documentation recommends the RMSprop for recurrent neural networks. According to Heaton, one hidden layer is enough to approximate the vast majority of non-linear functions (52). Two hidden layers were also explored and were chosen as they achieved lower validation accuracy. 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. As 20 nodes performed sufficiently well this was chosen for the final model. The activation function are non-linear stepwise equations that pass signals between layers. The options explored were Tanh, and ReLu which can be seen below..

$$Tanhx = \frac{sinhx}{coshx} = \frac{e^x - e^- x}{e^x + e^- x}$$

$$ReLU = f(x) = max(0, x)$$

These functions perform better on different types of tasks so it is important to test several. 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. Number of epochs tested ranged from 10 to 10000. Too many training epochs can result in overfitting. Overfitting is a common issue with neural networks. To reduce the risk of overfitting dropout can be implemented as discussed in the related work section. Optimal dropout between 0.1 and 1 was searched for both layers with .5 dropout the optimal solution for both layers. An early stopper was also implemented in the code. This Keras callback method stops the training of the model if its performance on validation data did not improve after 5 epochs. This helps to prevent overfitting. Generally the RNN converged between 20 and 40 epochs with early stopping. LSTM models converged between 50 and 100 epochs

## 4.2 LSTM

In terms of temporal length, the LSTM is considerably better at learning long term dependencies. As a result, picking a long window for this parameter was less detrimental for the LSTM as the RNN. 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.

Two hidden LSTM layers were chosen. For a time series task two layers is enough to find non-linear relationships among the data for a time series task. Three and four layers were tested but didnt improve validation performance. Several layers can be required for tasks such as image recognition when the number of features is significantly large. 20 hidden nodes were also chosen for both layers as per the RNN model. The Hyperas library[9] 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 performed the best for this task as indicated by the Keras documentation. The LSTM model activation functions werent changed as the LSTM has a particular sequence of activation functions within its cell made up of tanh and sigmoid activa-

[9]Hyperas: https://github.com/maxpumperla/hyperas

tion functions for the different gates within the cell.

LSTM models converged between 50 and 100 epochs with early stopping. LSTM models converged between 50 and 100 epochs in comparison to the RNN. This indicates the LSTM continues to learn for longer than the RNN. Batch size represents how often to updates weights using RMSprop. This was found to have a greater affect on execution time than accuracy. This may be due to the relatively small size of the dataset.

In terms of temporal length, the LSTM is considerably better at learning long term dependencies. As a result, picking a long window for this parameter was less detrimental for the LSTM as the RNN. This process followed a similar process to the RNN in which autocorrelation lag was used as a guideline.

Two hidden LSTM layers were chosen. For a time series task two layers is enough to find non-linear relationships among the data for a time series task. Three and four layers were tested but didnt improve validation performance. Several layers can be required for tasks such as image recognition when the number of features is significantly large. The Hyperas library was used to implement the Bayesian optimisation of the network parameters. The optimiser searched the optimal model in terms of the number of hidden layers and batch size. Batch size represents the number of data points to train on before updating weights. Similarly as above, dropout was implemented along with early stopping to prevent overfitting.

## 4.3 Model Comparison

A confusion matrix representing the ratio of true/false and positive/negative classifications is used to derive the ratings metrics. The formulae for the four metrics can be seen below. Accuracy can be defined as the total number of correctly classified predictions. This metric can be misleading for example when there is an imbalanced dataset. As a result, the metrics sensitivity, specificity and precision are also analysed. Sensitivity represents how good a test 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.

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{FP + TN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Accuracy = \frac{TP + TN}{P + N}$$

$$RMSE = \sqrt{(xi - yi)^2}$$

## 4.4 Validation

Holdout validation is used by default in Keras. The last 20 percent of the dataset is withheld for validation while the model trains on the remaining 80 percent of the date. This is the only method natively supported in Keras. K-fold cross validation was investigated through the Python Scikit-Learn library. However, it was not implemented due to the significant training times that result from it. Training times are k times greater than that for holdout. Sliding window validation was tested but performed poorly in a Theano version of the RNN implementation. As a result it was not implemented for the LSTM.

Holdout validation is used by default in Keras. The last 20 percent of the dataset is withheld for validation. The model trains on the remaining 80 percent. K-fold cross validation was investigated. However,

it was not implemented due to the significant training times that result from it. Sliding window validation was considered but not implemented due to time constraints.

### 4.5 ARIMA

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 as found by auto.arima from the R forecast package.[10]

## 5 Evaluation

### 5.1 Model Evaluation

As can be seen in table 1 LSTM achieved the highest accuracy while the RNN achieved the lowest RMSE. The ARIMA prediction performed poorly in terms of accuracy and RMSE. This was to be expected. 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. This contributed to the specificity and precision being so high (specificity,precision= 1). This does not necessarily suggest good performance.

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 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. Both models were combined to create a network with one LSTM and one RNN layers. This performed marginally worse to the pure LSTM model but required less training time. The ARIMA model appears to perform well based on sensitivity, specificity and precision. However, on analysis of the high RMSE of the model its apparent its forecast was poor. The ARIMA model forecast followed a stable path with very little variance. The fact that it appears to have performed well in terms of specificity and precision accuracy appears to have been down to chance as it failed to recognise any trends in the data and predicted the price would rise in general.

The figures listed are those that are thought to represent the generalizability of the model as appropriate prevention measures were taken in terms of overfitting. Dropout was considerably high on both hidden layers with the training dataset being shuffled in groups based on temporal window length. In addition, an early stopper was programmed into the model to stop training if the validation loss didnt improve for 5 epochs. Turning this off and it is possible to attain considerably better results but this is as a result of overfitting.

## 5.2 Performance Evaluation

The CPU utilised was a Intel Core i7 2.6GH/z. This is a relatively high specification processor for a laptop. 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. Performance can be seen in table 1 above. The GPU considerably outperformed the CPU as can be seen in figure 7. 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 is even greater.

The LSTM and RNN were also compared in terms of total training time on both the CPU and GPU combined as can be seen in figure 8. The LSTM took 3.1 times longer to train than the RNN with the same network parameters. One reason for this 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.

**Figure 7: CPU vs GPU**

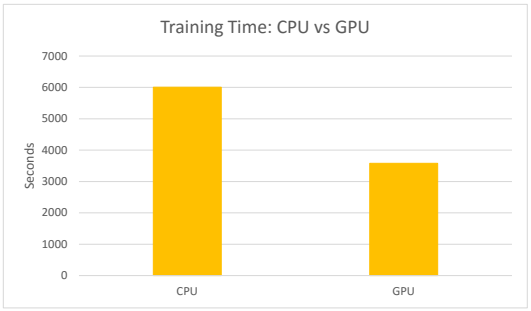



**Figure 8: LSTM vs RNN**

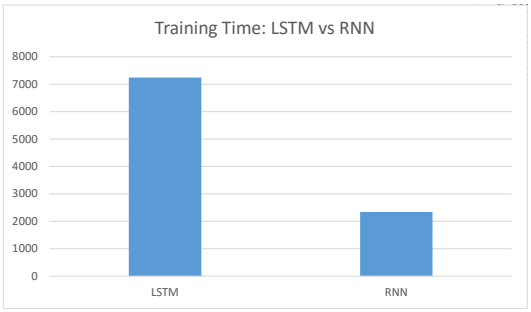



Table 1: Model Results

| 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% | 1 | 1 | 50.05% | 53.74% |

Table 2: Performance Comparison

| Model | Epochs | Intel Core i7 6700 2.6GH\z | NVIDIA GeForce 940M 2GB |
|---|---|---|---|
| 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 |

# 6 Conclusion and Future Work

Deep learning models such as the RNN and LSTM are evidently effective learners on training data with the LSTM more capable for recognising longer-term dependencies. However, a high variance task of this nature make it difficult to transpire this into impressive validation results. As a result it remains a difficult task. There is a fine line to balance 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 couldnt 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 there was not significant difference in the results of both. 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 on the GPU as opposed to the CPU. This confirmed the findings indicated by the related work. It appears from the results that Andrej Karpathy was correct in his article on the unreasonable effectiveness of recurrent neural networks (5). They can reduce validation error sufficiently low for a difficult task like this one. Looking at the task from purely a classification perspective one may be able 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 on streaming data would improve the model. Sliding window validation is an approach not implemented here but this may be explored for future work. One problem that will arise is that the data is inherently shrouded in noise.

Wavelets offer an interesting approach to time series analysis due to the way they break down a signal on a temporal level to get rid of noise. Some research exists in merging Wavelets with the MLP flavour of Neural Networks (53). To the best of the authors no implementation exists knowledge of a more advanced neural network like a LSTM or RNN being integrated with Wavelets. This is to be explored in future work. 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 from. The dataset utilised contained 1066 time steps representing each day. 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, par-

allelisation of algorithms is not limited to GPU devices. Field Programmable Gate Arrays (FPGA) are an interesting alternative to GPU devices in terms of parallelisation. Under some circumstances machine learning models have been show to perform better on FPGA than on a GPU (54). This warrants further investigation for deep learning models.

## Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Jason Roche for accompanying me on this deep journey of mine. I learned a tremendous amount from you in this short space of time. Thank you for believing in me and making me believe in myself and my abilities. I would also like to thank my girlfriend Katie and my parents for their unfaltering support and counsel. Finally, thank you to Dr. Simon Caton, Jonathan McNicholas and Patrick McHale for your technical advice and guidance when called upon.
)

## References

[1] 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.

[2] 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.

[3] 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.

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

[5] A. Karpathy, "The unreasonable effectiveness of recurrent neural networks," *Andrej Karpathy blog*, 2015.

[6] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

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

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

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

[10] B. G. Malkiel and E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *The journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[11] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

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

[13] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

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

[15] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," *Advances in neural information processing systems*, pp. 473–479, 1997.

[16] V. Ş. Ediger and S. Akar, "Arima forecasting of primary energy demand by fuel in turkey," *Energy Policy*, vol. 35, no. 3, pp. 1701–1708, 2007.

[17] 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.

[18] I. Daubechies *et al.*, *Ten lectures on wavelets*. SIAM, 1992, vol. 61.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] M. Matta, I. Lunesu, and M. Marchesi, "The predictor impact of web search media on bitcoin trading volumes."

[25] 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.

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

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

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by back propagation," *Parallel distributed processing: exploration in the microstructure of cognition*, vol. 1, 1986.

[29] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, "Predicting the future: A connectionist approach," *International journal of neural systems*, vol. 1, no. 03, pp. 193–209, 1990.

[30] Z. Tang, C. de Almeida, and P. A. Fishwick, "Time series forecasting using neural networks vs. box-jenkins methodology," *Simulation*, vol. 57, no. 5, pp. 303–310, 1991.

[31] 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.

[32] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[33] J. L. Ticknor, "A bayesian regularized artificial neural network for stock market forecasting," *Expert Systems with Applications*, vol. 40, no. 14, pp. 5501–5506, 2013.

[34] S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," in *Advances in neural information processing systems*, 2013, pp. 351–359.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.

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

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

[44] 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.

[45] 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.

[46] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

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

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

[49] 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.

[50] 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.

[51] 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.*

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

[53] L. Wang, K. K. Teo, and Z. Lin, "Predicting time series with wavelet packet neural networks," in *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, vol. 3. IEEE, 2001, pp. 1593–1597.

[54] 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.