MÄLARDALEN UNIVERSITY
SCHOOL OF INNOVATION, DESIGN AND ENGINEERING
VÄSTERÅS, SWEDEN

Thesis for the Degree of Bachelor of Science in Computer Science

THE INFLUENCE OF BITCOIN ON ETHEREUM PRICE PREDICTIONS

André Caldegren
acn15007@student.mdh.se

Examiner: Ning Xiong
Mälardalen University, Västerås, Sweden

Supervisor: Miguel León Ortiz
Mälardalen University, Västerås, Sweden

23 May 2018

Abstract

Cryptocurrencies are a cryptography based technology, that has increased massively in popularity in recent years. These currencies are traded on markets that specialize in cryptocurrency trade. There, you can trade one cryptocurrency for another, or buy one with real world money. These markets are quite volatile, meaning that the price of most cryptocurrencies swing up and down a lot. The largest cryptocurrency is Bitcoin, but there is also more than 1500 smaller ones, that goes by the name alternative coins, or altcoins. This thesis will try to find out if it is possible to make accurate predictions about the future price of the altcoin Ethereum, and also see if Bitcoin may have some influence over the price of the selected altcoin. The predictions were made with the use of an artificial neural network, an LSTM network, that was trained on labeled data from 2017. The predictions were then made in intervals of one hour ahead, six hours ahead, and one day ahead through early 2018. The predictions showed that it is possible to make somewhat accurate predictions about the future. The predictions that were made one hour ahead were more accurate than both the six hours ahead predictions and the full day ahead predictions. By comparing the loss rates of the neural networks that were only trained on Ethereum, with the loss rates of the networks that trained on both Bitcoin and Ethereum, is was made clear that training on both cryptocurrencies did not improve the prediction accuracies.

Table of Contents

1. Introduction

The area of cryptocurrencies are a rising subject within fields like computer science. Cryptocurrencies have had a massive increase in their popularity in the past years [1]. They are like a digital version of cash, with many of them striving for complete anonymity of their users [2]. The leading cryptocurrency being Bitcoin, with a market cap of about $144 billion, as of writing [11]. Many cryptocurrencies are open source, which has led to many new currencies being created. These smaller cryptocurrencies are called alternative coins, or altcoins for short. The coins may be traded directly between users, or by going through an exchange, or marketplace. Marketplaces often offer different coins that can be traded, and sometimes also offer the choice of buying cryptocurrency with real world money. However, since there is nothing behind the price of a cryptocurrency, except for the public opinions, the prices of cryptocurrencies are quite volatile.

Artificial neural networks is a group of algorithms that tries to replicate a brains functions, with neurons and weighted synapses, also called connections in this case. The weights determine how important the information from that neuron is in order to achieve the correct result. A simple neural network consists of some inputs going through their respective weights, being summed up, put through a special activation function, and then outputted as the results. More advanced neural networks, such as recurrent neural networks (RNNs), works better when the data is spread over time [3]. Some of the neuron outputs at this current timestep go back and becomes inputs in the next timestep. A more advanced version of a recurrent neural network is a long short-term memory network, or LSTM for short.

This thesis will focus on two questions: is it possible to predict the future price of the alternative coin Ethereum by the use of an LSTM network? And the main question: does the predictions improve when the network is allowed to train on both the market history of Ethereum, and of Bitcoin? The hypothesis here being that Bitcoin, since it is the largest cryptocurrency, has some type of influence over the smaller alternative coins. Several networks have been trained, two that predicts one hour into the future, two for six hours ahead, and lastly two for a full day ahead. Every interval has two neural networks, one only trained on the 2017 Ethereum market history, and one trained on both Bitcoin and Ethereum.

Neural networks such as a recurrent neural network or an LSTM network have been used before in solving problems like the ones that will be examined in this thesis. In a paper by Sean McNally, he tried to predict the future price of a single cryptocurrency, without any other information, with the use of both an RNN and an LSTM [3]. He proved that the LSTM network did provide better results than a regular RNN. RNNs have been used extensively in the area of stock market predictions. In a paper by Ken-ichi Kamijo and Tetsuji Tanigawa, a recurrent neural network was used to find patterns in the stock market, an important indication to where the price may be headed. In another paper by Emad W. Saad et al., an RNN, along with a time delayed network, was used to find profit opportunities on the stock market [4].

The method used to test these hypothesis's were the concept of falsification, as written by Zobel [5]. Furthermore, by making experiments and comparing the results with the predicted outcomes of the hypothesis's, it can be concluded whether or not the hypothesis were falsified. These experiments will be done with the use of neural networks, more precisely, a LSTM network. Some of these networks will train on only Ethereum, while others will train on both Bitcoin and Ethereum. The loss rates of these networks will then be compared, in order to find the most accurate one.

After the experiments were done, and the data was compiled into graphs, the results showed that it is possible to predict the future price. The predictions were shown to be better when they were not too far into the future. The one hour ahead model performed best, with the six hour model being second best and the one day ahead model being third. By comparing the loss rates of the different neural networks, the conclusion on whether training on Bitcoin improved the neural networks accuracy, came to the answer that it did not. Bitcoin did not improve the accuracy in any neural network model.

This report will start out with an explanation of what cryptocurrencies are and a bit about how they work in section 2. Furthermore, an explanation of different neural networks and the mathematics behind them will also be given, together with a learning method that will be used. In section 3, previous works in related areas will be analyzed. Section 4 will provide the questions, and limitations, on which this thesis is based on. The method for finding the answers is given in section 5. Section 7 will discuss the implementations that were needed in order to get the results. The results themselves will be shown in section 8, while a discussion about their value will be done in section 9. Lastly, in section 10 the conclusion is given, and in section 11, possible future work is provided.

2. Background

Since this thesis will encompass the area of cryptocurrency and machine learning, both will be explained in this section, giving the reader a basic understanding of the topics that will be further examined at a later stage. First, an explanation of what cryptocurrencies are and how they work will be given in section 2.1. Section 2.2 will explain a learning method that may be used in neural networks, among some other methods. In section 2.3, artificial neural networks will be explained in general, and recurrent neural networks will be examined in more detail together with long short-term memory networks.

2.1. Cryptocurrencies

Cryptocurrencies are a somewhat new technology that has exploded, both in popularity and economic value in, recent years [1]. Based on cryptography, what this technology aims to achieve is a decentralized currency. Cryptocurrencies can be compared to a digital version of cash, apart from cash not being decentralized [2]. By eliminating all the middle men, international transactions with a small to non-existing fee is made possible, compared to that of transferring money by the use of regular financial institutions [6]. Furthermore, by the elimination of middle men, cryptocurrencies effectively remove any institution, government, group or person from having full control. This makes most cryptocurrencies entirely decentralized.

Instead of relying on external actors to keep track of every transaction, cryptocurrencies rely on what is called a blockchain, also known as a distributed ledger [1, 2, 7]. The blockchain is one of the pillars of cryptocurrencies and is distributed to any user who has the cryptocurrency's software. The validation and authentication of transactions are left entirely to the users of the network to perform. When a new transaction is made between two users, workers on the network will validate and authenticate it, and insert it into the blockchain for everyone else on the network to see [8]. Since everyone knows every transaction, deceiving the system into giving someone unearned coins, or making two transactions with the same coin, at the same time, become impossible. Though there is often a small fee associated to making transactions, in the form of the cryptocurrency, this fee is used to award the workers who make the authentication process possible.

When a new cryptocurrency is released to the public, the control of that cryptocurrency is let go of by the creators, and the network is left to be ran by the users of that cryptocurrency. Furthermore, cryptocurrencies try to implement anonymity to the largest extent possible [9]. No transaction should be traceable to any single person. This is achieved by having anonymous wallets that can be created by anyone and are not connected to any person. These cryptocurrency wallets belong to the person that knows the password and has possession of the hardware on which the wallet is saved on.

Most cryptocurrencies are open source, allowing for anyone to understand how and why it works [10]. Furthermore, it allows for anyone to create their own cryptocurrency. This has resulted in the creation of hundreds of cryptocurrencies, the largest and with the most influence being Bitcoin with a value of $144 billion as of writing [11]. Every cryptocurrency that is not Bitcoin is categorized as an alternative coin, also known as an altcoin [10].

There are two ways of acquiring cryptocurrency: mining and trading. Mining works by a user offering their hardware's calculation power and, electricity to the network and in that becoming a worker [7]. In return, the worker receives some reward for their work and time. This is an option for most cryptocurrencies, though some coins do come pre-mined, meaning that the option of mining is not available. The other method of acquiring cryptocurrency is by trading for it. This may be done in person, buying from a company or by using a trading site, also called an exchange or may be referred to as a marketplace, where users are able to buy cryptocurrency with regular money, or with another cryptocurrency.

There are many different cryptocurrency markets in existence. What these marketplaces share is a certain volatility in the cryptocurrencies that are being traded [12-14]. Since there is nothing physically tied to a cryptocurrency, the only thing that affects its value is the perception that the public have of it [12, 15]. This have lead some researchers to try and categorize the current opinions on a cryptocurrency based on online community posts, to be able to predict the future price with some success [16].

2.2. Supervised Learning

Supervised learning is the process of making the neural network learn by providing it with examples. These examples are labeled in a way were the correct answers for a set of inputs is already provided. While supervised learning can be used for many different algorithms, this section will focus mainly on how it is applied to neural networks.

By training on labeled data sets, the neural networks can adapt their future predicted outcomes to reflect reality better. If the neural network does not correctly guess the output of an input set, the training algorithm will look at how much the guessed answer and the correct answer differs. The algorithm will then adjust the prediction function to provide better results in the future. The goal of this process is to achieve correct guesses as often as possible. When the training phase is done, data without a given answer will be put through the network, and a prediction will be made for the user.

The data sets used in this thesis do contain both inputs and the correct outputs. A majority part of this data should be used for training, while another part is used for testing how accurate the network is. Instead of using the results from running the neural network on the testing data to train, the predictions will be compared to the labeled results from the data sets to provide the accuracy of the network.

However, there are two main alternatives when data labeled with the correct answer is not available: unsupervised learning and reinforced learning. In unsupervised learning, the neural network, and probably also the creators, does not know what the correct output is from the outset. Because of this, there is no way to evaluate the accuracy of the network. In reinforced learning, as the name implies, behaviors is reinforced by providing the network with reward for guesses that lead to good results and punishing the network for guesses that lead to bad results. But since this thesis works with data that have the correct data for the training outputs, these methods will not be examined further.

2.3. Artificial Neural Networks

Artificial neural networks, also called ANNs, are a class of algorithms that tries to replicate the functions of the human brain [17]. By creating a model using neurons, often in multiple layers with an input layer, one or more hidden layers and an output layer, and synapses, also called connections, the artificial neural network can simulate a tiny part of a brain. Figure 1 presents a single neuron with synapse connections. If the network has more than a single hidden layer, it is classified as a deep neural network. Every layer in a neural network has an arbitrary number of neurons. Every neuron is connected to each neuron at the next level by weighted synapses. The weight of these synapses are what affects the end result. Neural networks employ backpropagation to improve their results on problems that have not yet been seen by the network. By using a learning algorithm, the neural network is able to update the synapse's weight, in such a way that the future results will be more precise. Aleksander and Morton [18] defines a neural network by the following two points:

1. *Knowledge is acquired by the network through a learning process.*
2. *Interneuron connection strength known as synaptic weights are used to store the knowledge.*

While there are a lot of different types of artificial neural networks, this section will focus only on feed forward neural networks, recurrent neural networks and long short-term memory networks, which is a special type of a recurrent neural network.

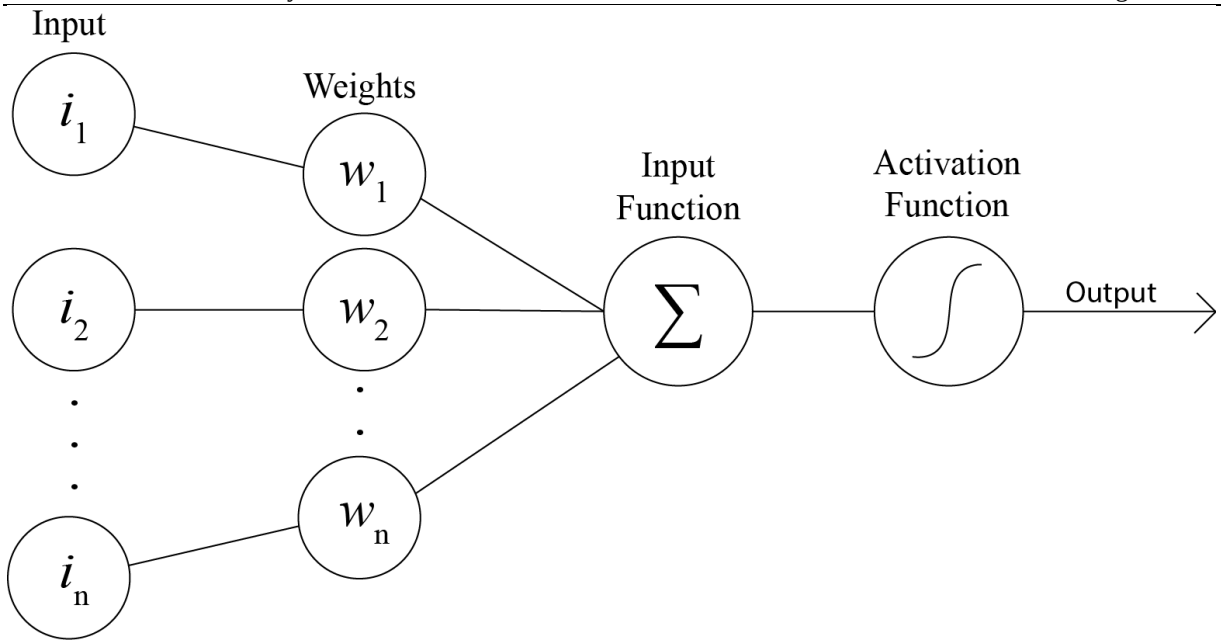

Figure 1: Single Neuron

2.3.1. Feed Forward Neural Networks

A simple form of a neural network is a feed forward neural network, seen below in figure 2. These networks take one or several inputs and feeds them through the synapses to the first hidden layer, then through the remaining hidden layers if any exists, and finally through to the output layer [19]. In figure 2, the neural network takes three inputs, passes it to a single hidden layer containing three neurons, and then passes that to the two outputs. These steps of passing information through a neural network is called a forward pass. As for most neural networks, there may be any number of hidden layers with any number of synapses in each, any number of input neurons and any number of outputs.

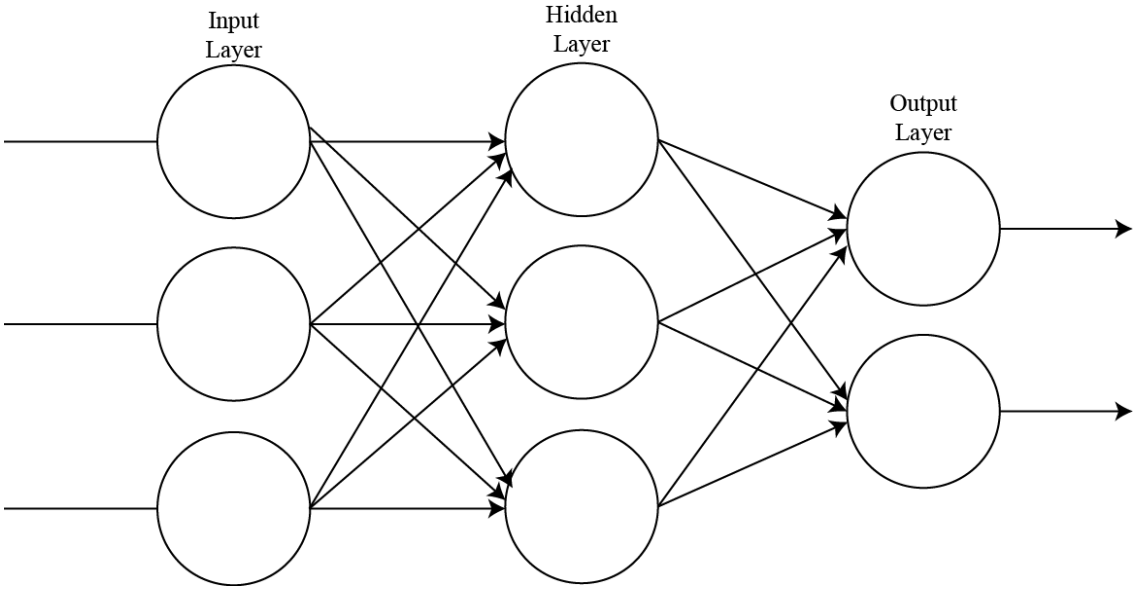

Figure 2: Feed Forward Neural Network

Feed forward neural network excel in classifying data but is not as good at making predictions where the relationship of the data is spread across time. Below in figure 3, a multi layered, deep, feed forward neural network is presented. This has two layers instead of one. As can be seen, every neuron in the first hidden layer is connected to every neuron at the second layer.

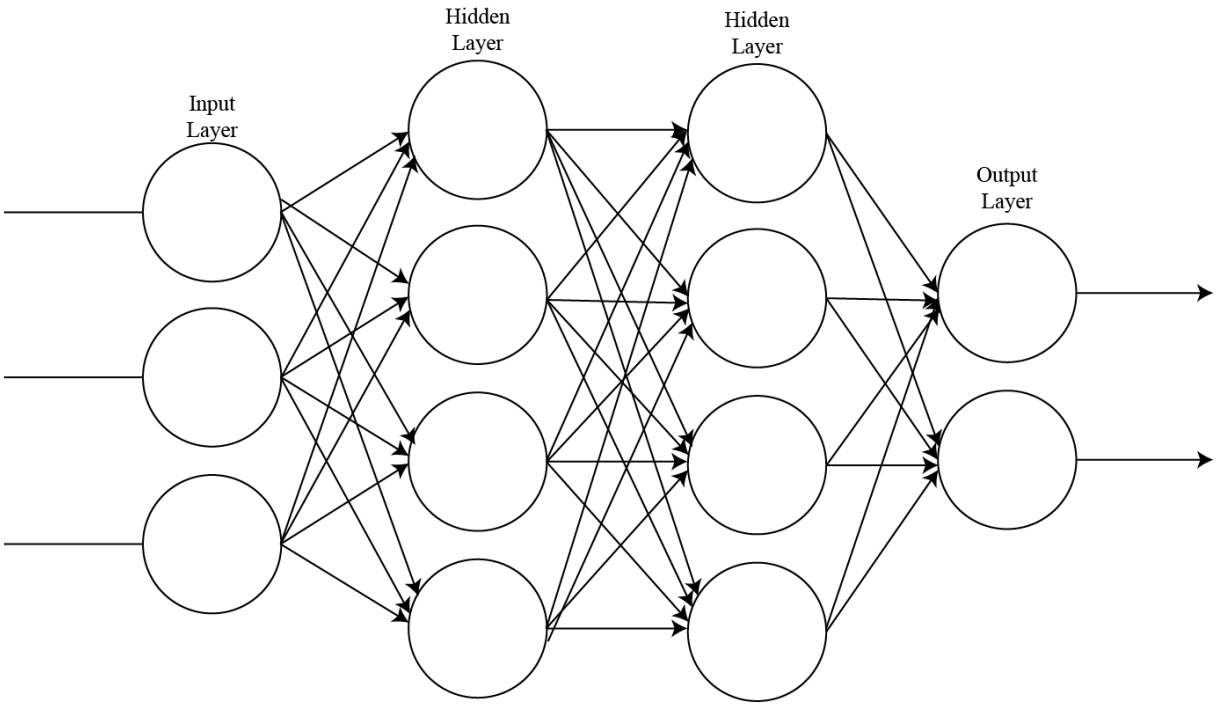

Figure 3: Multi-layer Feed Forward Neural Network

Figure 1 provides a clearer representation of how the individual neuron operates on the provided data. Both the input function and the activation function work together to create the output and provide it to every connected neuron in the next layer. The equation for this is:

$$y = f\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$

Where y is the output that will be fed forward to the next layer of neurons. x_i may be either the input that has been given to the neural network at the beginning, if this is the first layer in the network, or it is the output that was given from the neurons in the past layer at index i. w_i is a weighted synapse. Every x_i and w_i are multiplied together and added with the bias weight w_0. This is then passed through the activation function f, as seen in figure 1.

There are plenty of activation functions to choose from, some of the most common functions are shown in figure 4. The purpose of an activation function is to give an output from a neuron, within the limits of the activation function that was chosen. Worth noting is that an activation function is applied every time a neuron emits an output, and not only at the last output stage of the entire neural network.

The six activation functions shown in figure 4 are: two sigmoid functions, tanh, relu, softplus and gaussian. What these functions aim to achieve is to decide if the neuron should be fired or not. In the simplest activation functions, there is basically only “on” or “off”, as seen in figure 5, depicting the step activation function. The step function either outputs a 0 or a 1, though other similar functions may output -1 or 1 instead. More advanced activation functions, such as the sigmoid functions, provides the neural network with more nuanced outputs. Sigmoid for example, takes the input and returns a real, non-negative, output.

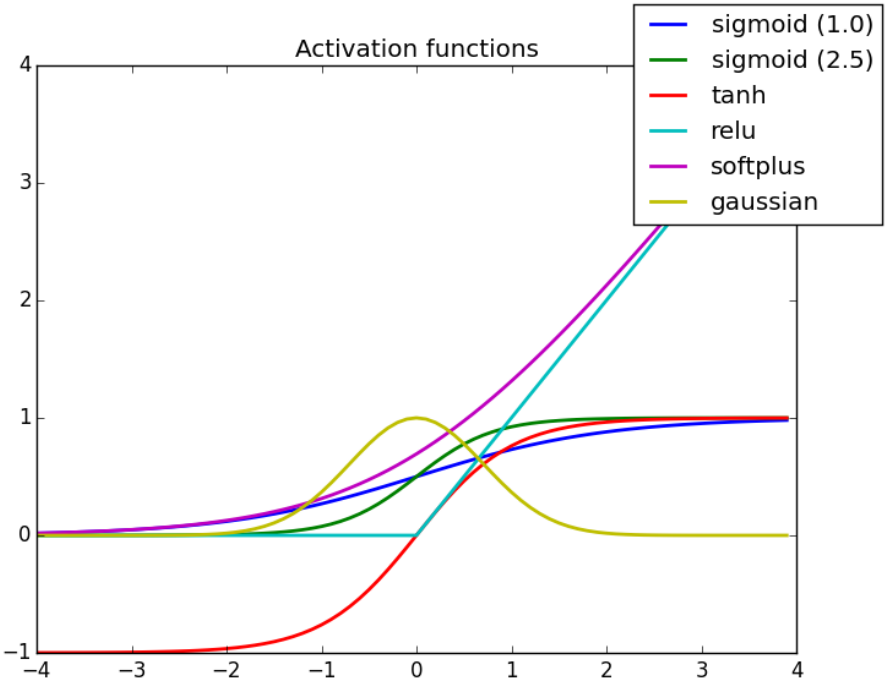

Figure 4: Activation Functions [20]

The three main activation functions from figure 4 are: sigmoid, tanh and relu. The three equations for these functions are:

$$sigmoid(t) = \frac{1}{1 + e^{-t}}$$

$$\tanh(t) = \frac{e^{t} - e^{-t}}{e^{t} + e^{-t}}$$

$$relu(t) = \begin{cases} 0, & t < 0 \\ t, & t \geq 0 \end{cases}$$

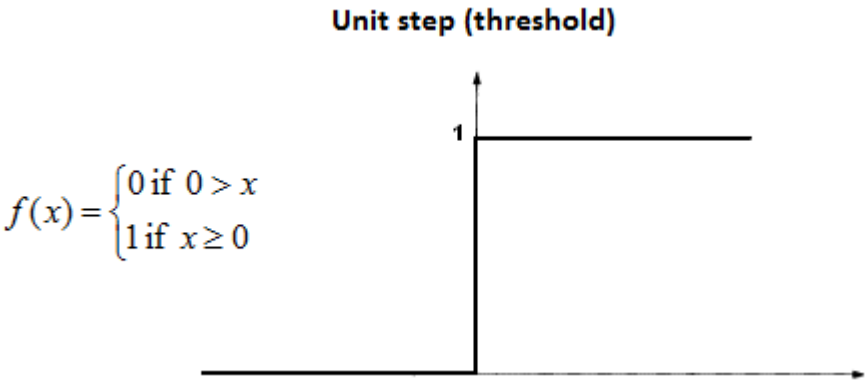

Figure 5: Step activation function [21]

If the neural network has been fully trained, the final output is returned to the user. On the other hand, if the neural network is currently being trained, the process called backpropagation is initiated. Assuming that the network is built for supervised learning, instead of returning the output data to the user, the data is compared to the expected answer that is given in the data set. The difference between the results and the expected answer will be used to update the weighted synapses to better provide a function that correlates to reality in a more suitable way. The amount that the weights are supposed to change by are given by the following equation:

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

This difference is added to the weight with the indexes j, i. η in this equation is the learning rate. The learning rate controls how much the weights can change during one training case. This is needed so that the weight values do not swing back and forth too much, making them volatile. It limits the amount of change possible, making the change both slower, but also more reliable. If the weights were to have huge differences from one training case to the next, the best value for a weight may be in between one of those swings, and therefore missed. Explained in the equations below are the remaining terms of the above-mentioned equation. δ is the error term, and x_{ji} is the input from the last layer. The error term at the output layer is explained below by the following equation:

$$\delta_j = e_j o_j (1 - o_j)$$

The error term is calculated a bit different for every layer that is not the output layer. In both cases, the term o_j is the output calculated at j. This is explained in the equation below:

$$\delta_j = -(1 - o_j) o_j \cdot \sum_{k \in Downstream(j)} \delta_k w_{kj}$$

The downstream in this equation refers to the neurons whose inputs include the output of the neuron at j. In pseudocode the entire process can be described by a few lines:

1. Do the forward pass to calculate the output
2. Calculate the output layers error term
3. Calculate the hidden layers error terms, from right to left
4. Update the weights
5. Repeat from step 1 until the error term is acceptably low

2.3.2. Recurrent Neural Networks

Recurrent neural networks (RNNs) differs from normal feedforward network in that they have an internal memory. As seen in figure 2 below, some of the outputs from the hidden layer goes back and becomes input. This makes them an appropriate choice for time series predictions. They have been used extensively in areas like stock market prediction [4, 22-24]. Therefore, recurrent neural network may be a better suited choice, compared to feed forward neural networks, when the relationship that is examined exists over time.

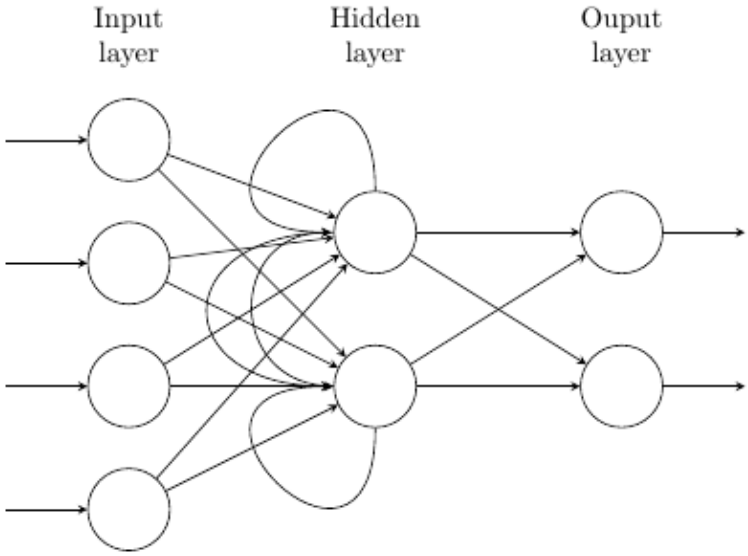

Figure 6: Recurrent Neural Network [25]

In figure 6 above, a simple recurrent neural network can be seen. Some of the outputs from the hidden layer goes back and becomes inputs for the same layer, only in the next timestep. Below in figure 7 an even simpler recurrent neural network can be seen to the left. On the right the unfolded version is presented. The unfolded version shows how the network actually works over time.

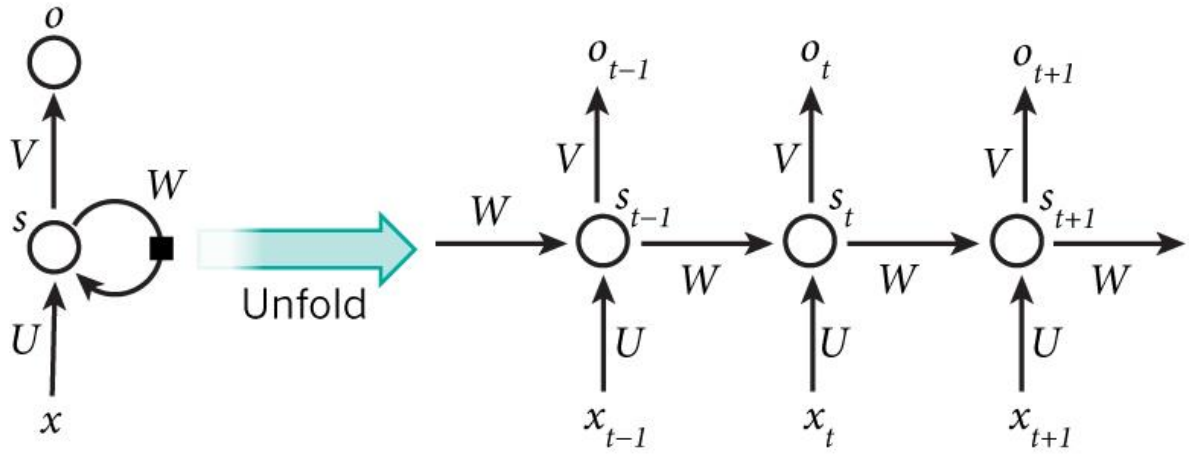

Figure 7: Unfolded RNN [26]

Recurrent neural networks have a type of memory. This memory works, as seen above in figure 6, because of the outputs from the hidden layer, that goes back and becomes input to the same layer once more. This memory can technically remember any range of the past. But in reality, this becomes harder the longer the time sequence that is being analyzed is. The RNN tends to forget over time and these relationships vanish.

Recurrent neural network has a forward pass, just like feed forward neural networks, where data is inputted and passed through the network, resulting in one or several outputs. Though some of the inputs will come from past executions of the network. This forward pass is explained by the following equation for the hidden neurons:

$$a_h^t = \sum_{i=1}^{I} w_{ih} x_i^t + \sum_{h'=1}^{H} w_{h'h} b_{h'}^{t-1}$$

In this equation, I is the number of input neurons and H is the number of hidden neurons. x_i^t is the value of the unit i at the time t and w is the value of the weight. $b_{h'}^{t-1}$ is the activation of a_h^t at the time $t-1$ by the activation function θ_h, which can be explained by the following equation:

$$b_h^t = \theta_h(a_h^t)$$

For the output neurons, the equation differs. The variables have the same representation as they have in the forward pass of the hidden layers. Instead of using the above equation, the following equation is used:

$$a_k^t = \sum_{h=1}^{H} w_{hk} b_h^t$$

Recurrent neural network implements backpropagation through time for learning and updating the weighted synapses. This is basically the same method as is used in regular feed forward neural networks. Again, the main difference is that the recurrent neural network does not only look at what has happened at the specific time, but also goes back in history. The backpropagation of an RNN can be explain though the following equations, starting with how the error term is calculated:

$$\delta_h^t = \theta'(a_h^t)(\sum_{k=1}^{K} \delta_k^t w_{hk} + \sum_{h'=1}^{H} \delta_{h'}^{t+1} w_{hh'})$$

$$\delta_j^t \stackrel{\text{def}}{=} \frac{\partial O}{\partial a_j^t}$$

The amount of which the weights are to be changes is calculated using the previously mentioned error term. Worth noting is that the same weights are used through every timestep. The following equation is used for summing the sequence and then using it to update the weights:

$$\Delta w_{ji} = -\alpha(\frac{\partial O}{\partial w_{ji}})$$

One problem with recurrent neural networks comes to light when learning the network. When teaching the network through backpropagation over time, the error term at the output layer diminishes, and reaches 0. This is called the vanishing gradient problem.

2.3.3. Long Short-Term Memory (LSTM)

Long Short-Term Memory tries to solve the problem of vanishing gradient in recurrent neural networks [27]. They work with recurrent connections, in the same way that a regular RNN does, with one main difference: they incorporate different blocks of memory. These memory blocks are composed of different gates: the forget gate, input gate, and output gate to, as the name may suggest, write, read, and reset. The gates allow the network to more precisely decide what to remember and what to forget. This solves the vanishing gradient problem since information can be stored and recalled over a longer time period, compared to that of a regular recurrent neural network [28].

LSTMs has both a forward pass and a backwards pass, just like regular feed forward neural networks, and recurrent neural networks. The forward pass is described by the following equations, starting with the input gate. All of the following equations are from the paper “Supervised Sequence Labelling with Recurrent Neural Networks” by the author Alex Graves [29].

2.3.3.1. Forward Pass

The equation for the input gates:

$$a_l^t = \sum_{i=1}^{I} w_{il} x_i^t + \sum_{h=1}^{H} w_{hl} b_h^{t-1} + \sum_{c=1}^{C} w_{cl} s_c^{t-1}$$

$$b_l^t = f(a_l^t)$$

f is, as before, the activation function that has been chosen for the neural network. a_l^t is the output that will be provided for the timestep t. a_l^t is passed through the activation function f and becomes the final output b_l^t. w_l^i are the weighted synapses between the networks input and the input gate and x_i^t is the actual value, and s_c^{t-1} is the state of the cell c at the timestep $t-1$. The summation of I is the number of all inputs, H is the number of cells in the hidden layer and C is the memory cells [29].

The equation for the forget gates:

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} b_h^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1}$$

$$b_\phi^t = f(a_\phi^t)$$

a_ϕ^t is the output of this gate, at this current timestep t. Weights are given by the variables $w_{i\phi}$, $w_{h\phi}$ and $w_{c\phi}$. x_i^t is the actual value at timestep t. And the state of the cell c is given by s_c^{t-1}. f is the activation function, same as for the input gate. The final output after the activation function is represented by the variable b_ϕ^t. The summations I, H and C represents the same values as in the case of the input gate.

Equation for the cells:

$$a_c^t = \sum_{i=1}^{I} w_{ic} x_i^t + \sum_{h=1}^{H} w_{hc} b_h^{t-1}$$

$$s_c^t = b_\phi^t s_c^{t-1} + b_l^t g(a_c^t)$$

Where a_c^t is the sum of the two summations at timestep t for cell c. w_{ic} are the weights from the input i to the cell c. x_i^t is the actual value, at timestep t. w_{hc} represents the weights between the recurrent connection to a memory blocks' output, from a previous timestep to the cell c. While b_h^{t-1} is the actual value. The summations I and H represents the same values as in the case of the input gate.

Equation for the output gates:

$$a_\omega^t = \sum_{i=1}^{I} w_{i\omega} x_i^t + \sum_{h=1}^{H} w_{h\omega} b_h^{t-1} + \sum_{c=1}^{C} w_{c\omega} s_c^t$$

$$b_\omega^t = f(a_\omega^t)$$

a_ω^t is the output sum of the three summations, for the output gate ω. b_ω^t is the final output of the output gate after the activation function has been applied to a_ω^t. $w_{i\omega}$ is the weight between the input and the output gate, while x_i^t is the value. $w_{h\omega}$ is the weight of the recurrent connection and b_h^{t-1} is the value of the previous timestep $(t-1)$. s_c^t is the state of the cell.

Equation for cell outputs:

$$b_c^t = b_\omega^t h(s_c^t)$$

b_c^t is the final output of the cell. In this equation $h(s_c^t)$ is an activation function applied to the state of cell c at timestep t.

2.3.3.2. Backward pass

In the backward pass, the same theory as with the backward pass from feed forward neural networks, are applied; with the addition that the pass is done through time as well. The equations for this is given below. While most of the variables represents the same things in both the forward- and backward pass, some variables are changed to make differences clearer. ϵ is used to represent the output, instead of having the variable a.

General definitions:

$$\epsilon_c^t \stackrel{\text{def}}{=} \frac{\partial O}{\partial b_c^t}$$
$$\epsilon_s^t \stackrel{\text{def}}{=} \frac{\partial O}{\partial s_c^t}$$

Equation for cell outputs:

$$\epsilon_c^t = \sum_{k=1}^{K} w_{ck}\delta_k^t + \sum_{g=1}^{G} w_{cg}\delta_g^{t+1}$$

For the cell outputs, K represents the number of outputs, and G represents the number of inputs given from the hidden layer. w_{ck} represents the weights from the output k to cell c. δ is a variable that represents the error term, when given as δ_k^t that represents the error term of the output k at the timestep t. w_{cg} is the weights from cell c to the input g. δ_g^{t+1} is the error term at the next timestep $(t+1)$ for input g.

Equation for output gates:

$$\delta_\omega^t = f'(a_\omega^t)\sum_{c\,=\,1}^{C} h(s_c^t)\epsilon_c^t$$

For the output gates, the summation of C represents all the memory cells. And δ_ω^t is the error term for the output gate ω at this timestep t. $h(s_c^t)$ is the application of an activation function on the state of cell c at timestep t.

Equation for states:

$$\epsilon_s^t = b_\omega^t h'(s_c^t)\epsilon_c^t + b_\phi^{t+1}\epsilon_s^{t+1} + \omega_{cl}\delta_l^{t+1} + \omega_{c\phi}\delta_\phi^{t+1} + \omega_{c\omega}\delta_\omega^t$$

This equation gives the state ϵ_s^t to the cell s at timestep t. The equation incorporates parts from most other equations to form this state.

Equation for cells:

$$\delta_c^t = b_l^t g'(a_c^t)\epsilon_s^t$$

δ_c^t is the error term for cell c at timestep t. $g'(a_c^t)$ is an activation function, g' applied to a_c^t, the value of cell c at timestep t.

Equations for forget gates:

$$\delta_\phi^t = f'(a_\phi^t)\sum_{c\,=\,1}^{C} s_c^{t-1}\epsilon_s^t$$

δ_ϕ^t is the error term for the forget gate ϕ at t. The summation C is, as before, a representation of all the memory cells.

Equation for input gates:

$$\delta_l^t = f'(a_l^t) \sum_{c=1}^{C} g(a_c^t)\epsilon_s^t$$

δ_l^t is the error term for gate l. Both f' and g are activation functions.

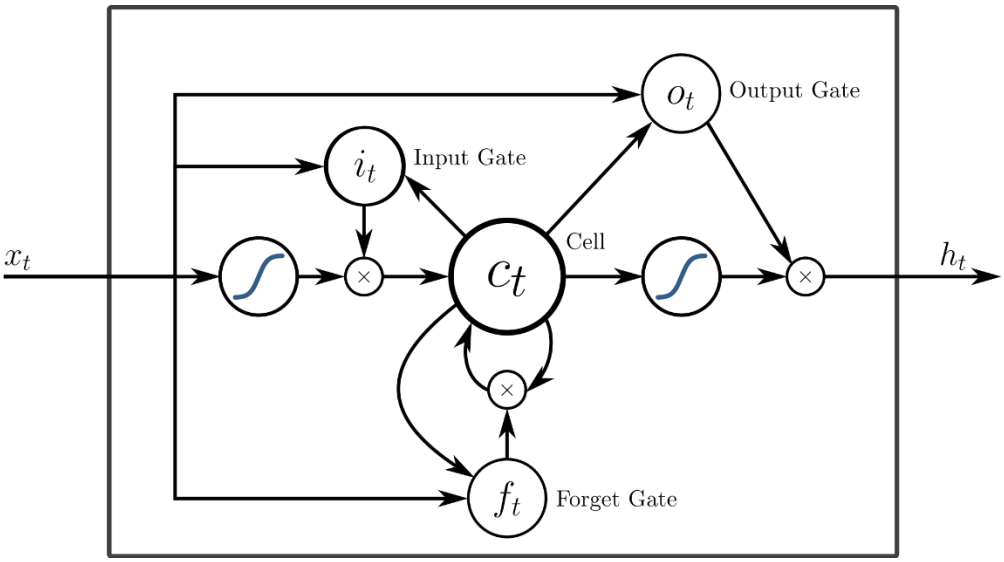

Figure 8: LSTM Cell [30]

The figure 8, above, represents a LSTM memory cell. Complete with an input gate, a forget gate and an output gate.

3. Related Work

Some research has previously been done in the area of trying to predict the future price of different cryptocurrencies. Different approaches have been used such as looking at community posts, trying to analyze the general public's opinion of a cryptocurrency, and therefore gaining a sense of the prospect of said cryptocurrency. Others have tries to analyze only the past market history of the selected cryptocurrency and tried to make predictions on that. This section will look at the research that has been done using the two previously mentioned strategies, but also look at how researchers have approached predictions of stock markets.

3.1. Analyzing the community to make predictions of the future

In the paper "Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies" from 2016, Young Bin Kim et al tries to predict fluctuations of the cryptocurrencies Bitcoin, Ethereum and Ripple [16]. By analyzing relevant user's comments and replies on community posts and assigning a tag, indicating if the post is positive or negative toward the cryptocurrency.
Working off the hypothesis that user comments and posts may have an influence on cryptocurrency prices, they started crawling for such data in the relevant communities. The tags assigned to these posts ranged from very negative, negative, neutral, positive and very positive. They used an algorithm called VADER to achieve this.
In the end, they were able to conclude that the number of comments and posts made in communities influenced the number of transactions that were to be made. Furthermore, their method proved useful in cryptocurrency trading. They also found a lag from when users made their posts, and when the market followed. The lag had a range of between 6 or 7 days.

In a paper by Martina Matta et al. a relation between Bitcoin and the number of tweets or web searches in explored [31]. By analyzing the number of tweets and the number of Google searches, on the key word "Bitcoin", the authors were able conclude that there existed a relationship. When the number of tweets and searches on the cryptocurrency went up, the price followed. Furthermore, by doing sentiment analysis on the tweets, the authors were able to conclude that many positive tweets may have a correlation to future price increments.

3.2. Analyzing past market history

Several attempts at predicting the future of multiple cryptocurrency markets have been made without looking at anything other than data that can be gathered from the markets themselves, with different amounts of success. A common tactic in achieving good results seems for most to utilize some type of a neural network algorithm, accompanied by supervised learning. Some have tried to use simple feed forward neural networks, whilst others take the recurrent neural network approach, sometimes with the added bonus of LSTM. The data that is being used varies from paper to paper, while some studies only focus on the past market history pricing of a single cryptocurrency, other take a more inclusive approach by including market open, close, high and low prices as well. This section will examine some these papers whom in some way have made attempts at predicting the future market. The methods, results and conclusions from the papers will be summarized in this section.

In the paper "Predicting the price of Bitcoin using Machine Learning", author Sean McNally attempts to predict the future price of the cryptocurrency Bitcoin [3]. Furthermore, he wants to see with what accuracy the predicted price's direction can be predicted by a neural network. The neural network in question would come to be both a recurrent neural network and an LSTM network.
When designing the recurrent neural network, McNally chose to find the best design of the model by using both a grid search and a Bayesian optimization. This allowed him to fine tune the network, until the best parameters, those that would provide the most accurate results, where found. The LSTM network also used Bayesian optimization to find the optimal parameters where it was possible to do so.

After running these neural networks, he could conclude that the LSTM provided him with a higher accuracy, rather than the regular recurrent neural network. While the LSTM was better, he was not able to reduce the error any lower than 8.07% when testing on the validation data. The overall accuracy for the LSTM was 52.78%. For the RNN the accuracy dropped to 50.25%. However, it should be noted that these numbers of accuracy are not for future pricing. McNally only trained on and validated on already existing market data.

In the paper "Bayesian regression and Bitcoin", the authors Devavrat Shah and Kang Zhang utilize Bayesian regression for creating a Bitcoin trading strategy [32]. This strategy predicts the future average price of Bitcoin over a 10 second interval. If this average reaches certain thresholds, they either buy, sell or hold their assets.

The data was collected over a duration of a few months, with an interval of 10 seconds between each data point. With the collected data, Shah and Zhang created three data vectors, one that contained the data with a time-length of 30 minutes, one with 60 minutes and one with 120 minutes time-length. When a prediction is to be made, these three vectors where used for the Bayesian regression.
For their results they simulated their strategy over a one-and-a-half-month period. They were able to get an approximate 89% return over 50 days of the simulation. This equaled a profit of 3362 yuan on a 3781 yuan investment. The strategy even performed well when the market was at its most volatile.

Zhengyao Jiang and Jinjun Liang trained a neural network model on a year's worth of Bitcoin price data, where every data entry was at an interval of 30 minutes [33]. The network used was a convolutional neural network. While they got positive results with the help of their model, a passive aggressive mean reversion algorithm gave a better return.

In another paper, by the authors Huisu Jang and Jaewook Lee, an empirical study on the performance of a Bayesian neural network in predicting Bitcoin's future price was made [34]. The Bayesian network was compared to other networks such as linear regression and other. The Bayesian network proved to provide good results, better than those of the linear regression. In a similar paper by Alex Greaves and Benjamin Au, where they compared neural networks with other methods such as linear regression, Greaves and Au found that a neural network model with two hidden layers proved to be best [35]. This shows that several types of neural networks can achieve better results than a linear regression model.

3.3. Stock market predictions

Predictions of the stock markets has been something that, if done correctly, could provide a stock market trader with greater insight into what might happen next. This has lead financial institutions, private persons and researches to spent time and effort trying to come up with some prediction methods that works. Some of the researched areas are fundamental analysis, technical analysis and data mining. The stock markets and cryptocurrency markets have similarities between them that might translate from one to another. This section will focus on how neural networks have been used on mined data to predict stock market trends and might happen to a stock in the future, in the name of increasing returns.

The authors Ken-ichi Kamijo and Tetsuji Tanigawa took a recurrent neural network approach in trying to recognize stock market price patterns in their paper "Stock Price Pattern Recognition – A Recurrent Neural Network Approach" from 1990 [23]. After gathering and analyzing stock data for the past three years, they were able to find triangle patterns, which, according to the, is important indications of where the stocks might be headed.

In the paper "Stock Market Prediction Using Artificial Neural Networks" Birgul Egeli et al. uses artificial neural network to predict the Istanbul stock exchange's future [24]. They used both an multi layered perceptron structure to their network, as well as a generalized feed forward structure. Supervised learning was used to train the networks over a data set of 417 days of price history. In conclusion, it was determined that both structures performed better than a moving average, and that generalized feed forward performed the best.

Emad W. Saad et al. used time delay neural networks and recurrent neural networks in their paper "Advanced Neural Network Training Methods for Low False Alarm Stock Trend Prediction" to find profit opportunities on the stock market, with as few false alarms as possible [4]. With the time delay neural network they were able to predict 30 opportunities for Apple's stock out of 94 with 0 false alarms, 37 opportunities for IBM out of 117 with 8.1 false alarms and 29 opportunities for Motorola out of 108 with 13.8 false alarms. With the recurrent neural network, they were able to predict 36 with 2.8 false alarms, 78 with 1.3 false alarms and 39 with 5.1 false alarms respectively.

In another paper by T. Kimoto et al., several prediction models were tested on the Tokyo Stock Exchange [36]. With the use of a simple neural network, strucured in three layers: the input layer, a single hidden layer and an output layer, the authors aimed to predict future stock market prices. After running simulations with their neural network model over a period of 33 months, they could conclude that the prediction model has achieved accurate predictions that would lead to good profits.

4. Problem Formulation

The cryptocurrency market can be extremely volatile. This poses several questions. Is it possible to predict a cryptocurrency's, an altcoin more specifically, future value? By looking at the past market history of the specified altcoin, can a prediction be made on what the price will be in 10 minutes, an hour or even several days in advance?

Additionally, would this prediction improve by not only looking at the altcoins past market history, but also looking at the history of the leading cryptocurrency, Bitcoin? Since Bitcoin is the largest cryptocurrency, and is often used to buy smaller, alternative coins, the hypothesis is that Bitcoin has some influence over the price of other cryptocurrencies. So what improvement in the predictions, can be seen if both market histories are considered, instead of only looking at one? The following questions will be investigated:

• Based on the past market history, is it possible to predict the future market price of Ethereum?

• Can predictions of Ethereum's future price be improved by also looking at the past price of Bitcoin?

The main motivation for this work is to see if there is a simple way of improving price predictions without going as far as analyzing whole communities.

4.1. Limitations

It is not certain that this thesis will result in a neural network that can predict the actual future price. If deemed too hard to accomplish or impossible to achieve, the focus might be switched to making the neural network predict if the price is about to go up, down or stay at about the same level.

The software to be implemented will only focus on making predictions about the future. It cannot make any decisions at all by itself, meaning that it will not be able to make trades on its own, like a trading bot. It is only supposed to be used as a tool for possible investors to make predictions about where the market may be headed.

Since there are many different cryptocurrencies, it may be possible that the influence on a selected cryptocurrency may not only come from Bitcoin but may also come from other cryptocurrencies as well. This thesis will only take the influence of Bitcoin on the selected alternative coin into consideration. Future work could be done to research the correlation of multiple cryptocurrencies.

In what way a neural network should be built, for achieving the best results will not be examined in much detail. Instead, this thesis will only focus on finding out if it is possible to get better results with a single type of neural network, a LSTM network. The parameters used in that network will not be examined in a lot of detail either, since that is not really the goal. Better parameters may provide better results, but that is left to future research. Furthermore, which type of neural network that provides the best results is not examined, this is also left for future research to find out.

5. Method

Justin Zobel wrote in the book “Writing for Computer Science” about the concept of falsification. Falsification is the principle that any hypothesis made should have the potential of being false [5]. This implies that no hypothesis may be vague or untestable. Therefore, a hypothesis must be concrete enough, as to be tested. Furthermore, Zobel claims that no hypothesis can be proven true, no matter the amount of scientific evidence. A hypothesis is only able to be proven false.

During this thesis, falsificationism will be used. By first setting up a reasonable and testable hypothesis, and then figure out a way of performing tests to try and see if the hypotheses still hold up, or if it can be considered false. A prediction of what these tests will result in if the hypothesis were to be validated will be done. After analyzing what tests seem reasonable, the experiments would begin. The result of the experiments would yield a result that will be used for making conclusions about the hypothesis. In short, the four following steps will be adhered to:

- Hypothesis
- Predictions
- Experiment
- Results

Applied to this thesis, the hypothesis is that Bitcoin will improve the predictions of the selected alternative coin. If this hypothesis is true, an increase in the neural networks certainty and correctness of the predictions made, would be shown. The experiment that is needed to test this are a comparison of the predictions with only the alternative coins market history, and one with both the alternative and Bitcoins past market history. By comparing these predicted future prices with the markets real prices, the level of correctness can be examined and compare between the two methods.

A short study of the related areas will be done. It will describe where the current research is at the moment and what their results have been. By analyzing these previous works, it may be possible to get a better understanding of how to achieve good results in this thesis and understand what tools should be used, and what to avoid.

Several neural networks will be required. At least one with the market history of only the alternative coin, and one with both the market history of the alternative coin and Bitcoin. Having different neural networks that are trained on different ranges of history may be a good option to analyze how far back into the history it is worth looking. Some neural network may be trained on the past day, another one that is trained on the past few weeks, and a final one that is trained on the full market history.

To analyze how well the neural networks can predict the future price, a software that visualizes the market and our future predictions will be required. This software should have multiple options on what neural network it should display. Options should be presented to the user, allowing them to make choices on what they want visualized. For example, if the market history of Bitcoin should be considered, how far back on past market history the neural network should be trained and so on. This software will do the actual predictions and visualizations.

Another software that is responsible for the training on the data sets will be required. This software will train several neural networks that the previously mentioned visualization and prediction software can utilize. The software should continuously train on newly attained data from the markets. Acquiring the new data will be handled by a server.

A script for attaining the data set will be required. It is important that the data comes from the same market, both for Bitcoin and for Ethereum. This software should be able to fetch data through different time ranges, and also capture data at different intervals. The data entries need to be checked, so that there is one matching Ethereum entry for every matching Bitcoin entry, and vice versa.

6. Ethical and Societal Considerations

As this thesis does not involve any data collected from regular users, and only looks at data provided by the marketplaces, no ethical standpoint is taken into consideration. And since the data is offered by market providers without personal information about its users attached, no infringement on people's personal integrity will be made.

7. Implementation

In this section, the implemented software will be examined and explained in detail. These implementations are what have made fetching data sets, teaching the neural network and making predictions possible. All of the code is written in the programming language python, with the Keras library for making the creation of neural networks less cumbersome. We will start with taking a look at some of the libraries that were suitable for this thesis, and examine the Keras library in more detail, in section 7.1.

The implementation consists of three separate scripts: *nn.py*, *predict.py*, *evaluate.py* and *gather_data.py*. *nn.py* contains the code for teaching the neural network and the saving the model, *predict.py* loads an already existing model and makes predictions on it, *evaluate.py* evaluates the performance of the neural network, and *gather_data.py* gathers data for *nn.py* to train on.

7.1. Neural Network Libraries

Implementing a neural network is time consuming and requires extensive knowledge. Therefore, several libraries have been created to help users with fast and easy prototyping. This section will discuss some of these frameworks.

TensorFlow is an open-source library that was released in November of 2015 by the Google Brain Team [37]. It is available in official capacity for python and C, and in unofficially for C++, GO and Java. It is able to run on a single CPU or GPU, or if there is enough resources, it can run on more than one, for improved speed. In general, it is used for machine learning, often with the use of neural networks.

Keras is a API that uses TensorFlow as a backend [38]. The sole purpose of the Keras API is to make prototyping neural networks as easy as possible. By building on an already existing backend, prototyping neural networks even faster and easier than by using only TensorFlow. It is possible to use different backends other than TensorFlow. For example, a backend with either Theano or CNTK is possible. However, for this thesis, the recommended TensorFlow backend was used.

7.2. Fetching data

Fetching of the data was done in the script *gather_data.py*. This script utilized the GDAX API to gather data from the marketplace Coinbase at selected intervals. This in return, creates a comma delimited csv file. The csv file contains entries for every timestep from the start time to the end time. Each entry contains a UNIX timestamp, the price information of both Bitcoin and Ethereum as gathered from the GDAX API. Furthermore, the timestamp is translated into 12 ones and zeroes representing the month of the year, 31 ones and zeros representing the date, and lastly 24 ones and zeroes representing the hour of the day. While the time may not have been used in this particular thesis, it may be useful in the future. The dates where transformed into ones and zeroes by using the python library datetime and through array operations.

The GDAX API only allows for 300 data entries to be fetched at a time. The python script takes care of this if the time interval is of a size where more than 300 entries would be returned. Furthermore, the API blocks requests if they are made within too short of a time between each other. The script recognizes this as well, and when a request is dismissed, the script sleeps for one second before trying again.

Since data for both Bitcoin and Ethereum cannot be fetched with the same request, the script has to do one request for every interval of Bitcoin information, and a separate one for Ethereum. If the timestamps for some entry in one or the other sets does not have a matching entry in the other set, that entry is discarded by the script.

These parameters are possible to adjust, if needed, though it requires that the user has some coding skills:

- start_time: A UNIX timestamp which tells the API where the user wants to start fetching data from. However, it is not assured by the API that this timestamp is used exactly as provided.

- end_time: A UNIX timestamp which tells the API where to stop fetching data. This is also not assured by the API to be used precisely.

- tz: A time zone from the pytz python library, this can be changed if the user wants to, but should be set to the same time zone as the servers are located in, from which the data will be pulled.
- filename: the filename of the csv file that will be created by the script, this file will not be sorted.
- filenameSorted: the filename of the final csv file that will be used by the neural network to train. All data in this file will be sorted by the UNIX timestamp.
- firstRate: the wanted output currency for a cryptocurrency, for example "BTC-USD" gives the Bitcoin price in USD.
- secondRate: same as firstRate, these two should be different. In this script, firstRate is set to "BTC-USD", while secondRate is set to "ETH-USD".
- sleepTime: in seconds, how long the script should sleep if a request is blocked by the API.
- interval: the interval at which to get data entries, in seconds. By GDAX API restrictions, this value can only be set to one of these options: 60, 300, 900, 3600, 21600 or 86400. This corresponds to: one minute, five minutes, fifteen minutes, one hour, six hours, and one day.

7.3. Learning

The script *nn.py* contains the code for building a neural network model and training it on a csv file. The model itself may be changed however the user wants. However, this requires more extensive programming knowledge and knowledge of the library Keras. The following parameters can be changed without a lot of previous knowledge from the user:

- filenameTraining: The csv filename from which the data set used for training the neural network will be taken from.
- filenameValidation: The csv filename from which the data set used for validating the neural network will be taken from.
- windowSize: the number of data entries that will be used to predict a single output. For example: if windowSize is set to 24, and the data entries are an hour apart from one another, a full day of previous data will be used to predict a single hour ahead.
- neurons: the number of neurons that will be in the hidden layer.
- activationFunction: this is the activation function that will be used for all the layers. It is currently set to "relu".
- filenameModel: the filename for the model. It will be saved in the h5 file format that Keras can load at a later stage, or in another program.

The following parameters can be set for the actual training method call to Keras:

- optimizer: currently set to "adam" but can be changed to whatever the user desires and is supported by Keras.
- epochs: the number of epochs that will be ran, in other word, how many times the neural network will train on the data set.
- batch_size: how many data entries that will be taken by the neural network at a time. Default is 32.
- x: the values that will be put through the network, and predictions be made out of.

- y: the correct answers to every entry in x.
- verbose: if set to 0, Keras will not output any information on how the training is progressing. If set to 1, Keras will output as much information as possible, including a progress bar. If set to 2, Keras will only output text when it reaches a new epoch.
- callbacks: here a list of functions can be provided. These functions are applied at a certain stage of the training process.
- validation_split: the amount of data that should be used for validation. Given as a floating value between 0 and 1, representing a percentage. This is not set since the data provided to the functions is already split in two.
- validation_data: provided a tuple of x and y data, this is the data that will be used to evaluate the loss of the network, at each end of an epoch.
- shuffle: true or false, should the data be shuffled. This should be set to false.
- class_weight: this can be used to give more weight to certain classes of samples during testing.
- sample_weight: optional weights for samples. However, this is not currently being used.
- initial_epoch: The epoch that you want your neural network training to start at. Currently this is not set, which means 0, but it a user wants to continue a training session from a certain epoch, this may be good.
- steps_per_epoch: The number of batches in an epoch.
- validation_steps: The number of batches to use for validation.

The functions that this script uses are:

- getBTCSum: this may be used if the user wants to normalize their values.
- normalize: Used if the user wants to normalize their cryptocurrency price information.
- getData: provided a filename, this function gets the data from a csv file, and inputs it into a list.
- splitData: given a percentage, this function will split a provided list of data entries into two lists. One for training and one for validation.
- createInputs: given a list of the data that should be inputted through the network, this function arranges a list in the correct way for Keras.
- createOutputs: given a list of data, this function creates the correct outputs that should be predicted by the neural network.

7.4. Predicting

In the script *predict.py*, predictions on a data set can be made, provided an already existing model of a neural network. First off, the input data from a csv file is read into a list. From this list, two separate lists are created: one for inputs into the neural network, and one with the actually correct answer. After that, a previously created and trained neural network model is loaded. Predictions can now be made with the data, and model. After running the data through the neural network, the script will output the guessed answer, and also provide the user with a graph where the predicted values and the correct values are shown.

The adjustable parameter for the prediction script are:

- filename: The path to the csv file, from which the inputs that are to be passed through the neural network, at which it can be found.
- windowSize: this needs to be set to the same value that was set for set loaded model while it was training.

The parameters that can be modified in the Keras predict method are:

- batch_size: how many data entries that will be taken by the neural network at a time. Default is 32.
- x: The input data that will be passed through the neural network and is used to make the predictions.
- verbose: 0 for silent, 1 for everything including a progress bar, 2 for only the most important parts.
- steps: the number of batches to run before being done with the predictions.

7.5. Evaluation

The evaluate script takes a data set and an already created and trained neural network and runs it through the Keras evaluate function, which provides the user with the loss rate of the neural network. So, given a data set and a network, this will provide the user with a general sense of how accurate the network actually is at making the predictions. The following parameters may be changed in the script:

- filename: the path to the csv file where the data set is taken from.
- modelPath: the path to the neural network model that will be loaded into the script and evaluated.

The following parameters may be changed by the user for the Keras evaluate functions:

- x: this is the data set that will be fed through the neural network for predictions to be made on.
- y: these are the correct answers that the evaluate function will compare the predictions to.
- batch_size: should be set to the same value as in the previous scripts. This corresponds to the amount of data entries that will be put through the network at the same time.
- verbose: 0 for silent, 1 for everything, 2 for the most important messages.
- sample_weight: this is optional and is not really used in the script. However, it tells the function if it should weigh some samples heavier than others.
- steps: number of steps that will be taken before the evaluation is done. This is currently set to the default of None.

8. Results

The resulting neural network was a LSTM network with a single hidden layer with 16 neurons. A graph of one of the neural networks can be seen in figure 9 below. This particular figure is of the neural network that was used to predict the hourly price, by using both the price of Bitcoin and of Ethereum for its predictions.

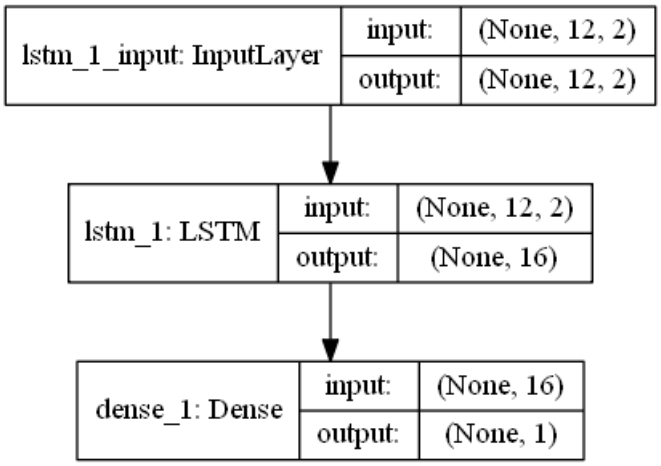

Figure 9: LSTM network

The parameters for the different neural networks were mostly the same, though some differences were necessary. These are the parameters that were the same for every neural network:

- Input: the input for the training was always done on data from 2017, though the amount of data may differ because of the different time intervals (one hour, six hours, and one day).
- Output: the expected output taken from the 2017 data, the amount of data differs here as well, for the same reasons as for the input.
- Neurons: 16.
- Epochs: 3000, though not all of the epoch may have been run through, because of early stopping.
- Batch_size: 32.
- Shuffle: False.
- Optimizer: adam, this optimizer has a learning rate of 0.1%.
- Loss: mean squared error (mse).
- Callbacks:
    - Early stopping:
        - Monitor: val_loss.
        - Min_delta: 0.
        - Patience: 100, this is how many epochs the software will wait before terminating the training if no better val_loss has been achieved.
        - Mode: auto.
    - ModelCheckpoint:
        - Monitor: val_loss.
        - Save_best_only: True.
        - Save_weights_only: False.
        - Mode: auto.
        - Period: 1.

What differed between the neural networks where, except for the amount of data, the window size. For predicting one hour a window size of 12 where used, for six hours and a full day a size of 2 were used.

8.1. Predictions

The predictions that were made one hour ahead resulted in the graph below, when the neural network was only allowed to train on the price of Ethereum. Here the predictions can be seen as the blue line, while the actual values are marked in orange. Predictions that were made one hour ahead with only Ethereum training are show in figure 10, and training with both Bitcoin and Ethereum are show below in figure 11.

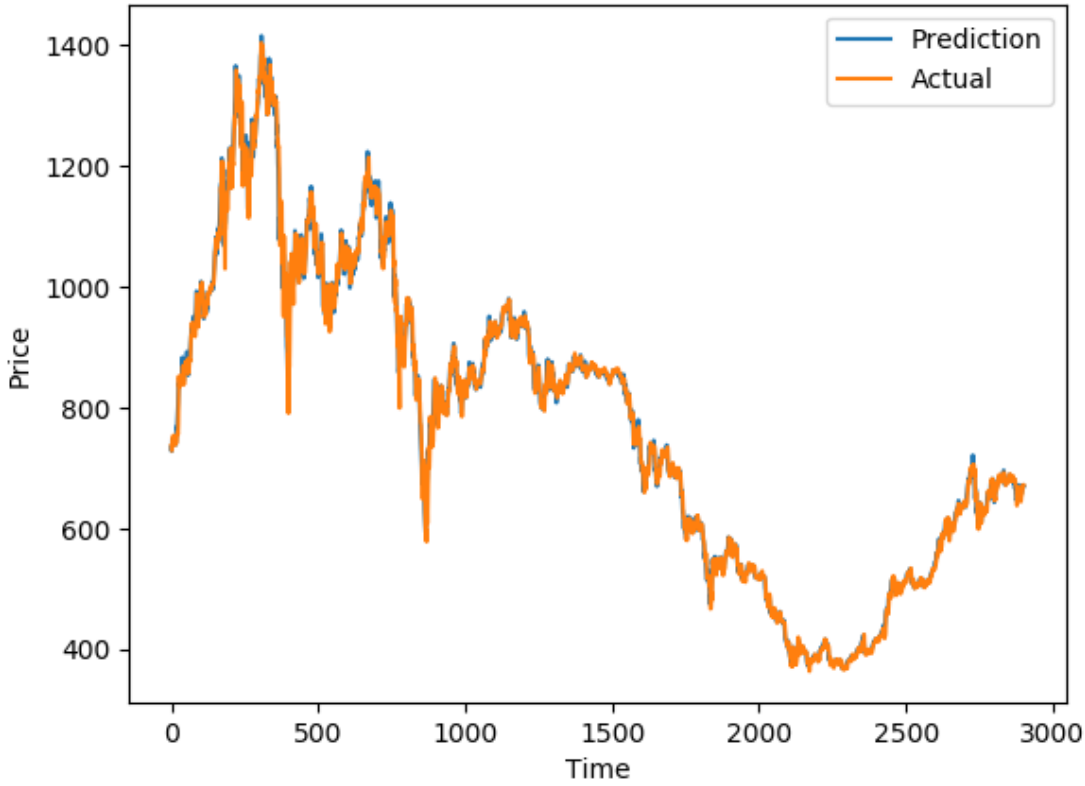

Figure 10: predict 1 hour ahead, only Ethereum training

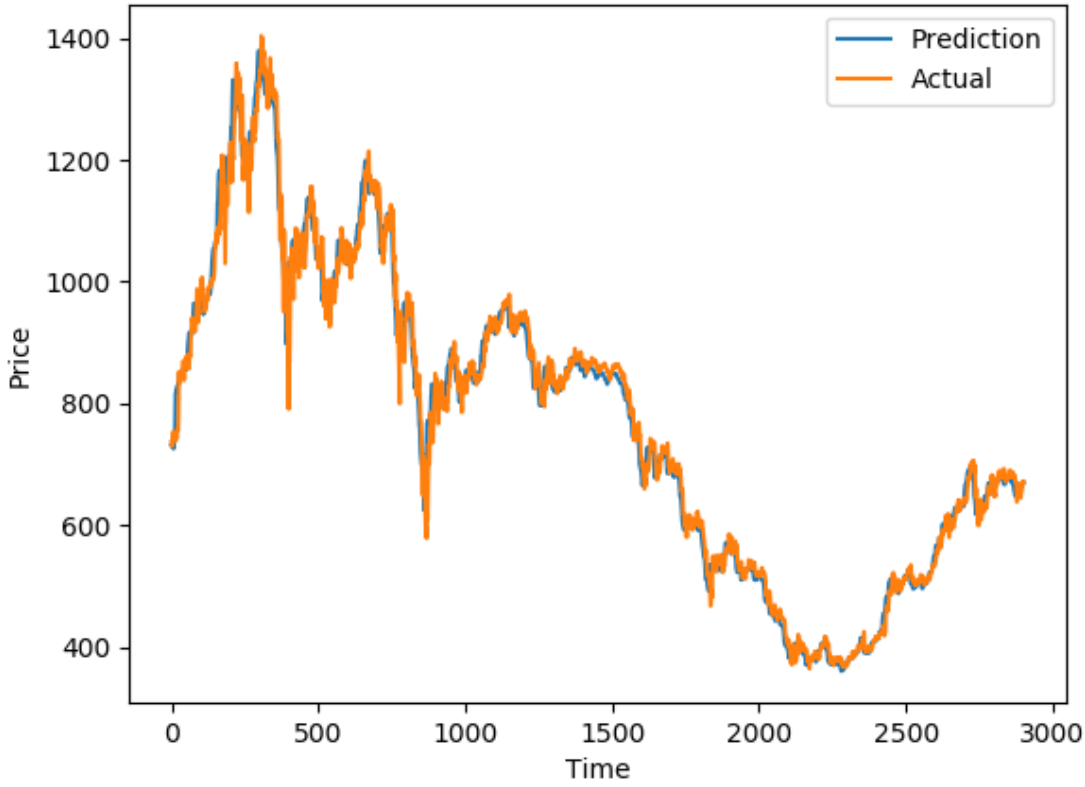

Figure 11: predict 1 hour ahead, Bitcoin & Ethereum training

For the six hours ahead predictions, twelve hours of previous data were fed through the network to predict a single price value. The training used a 2017 data set where the entries where spaced six hours apart. The figure below shows the predicted prices over 2018, when only trained on Ethereum.

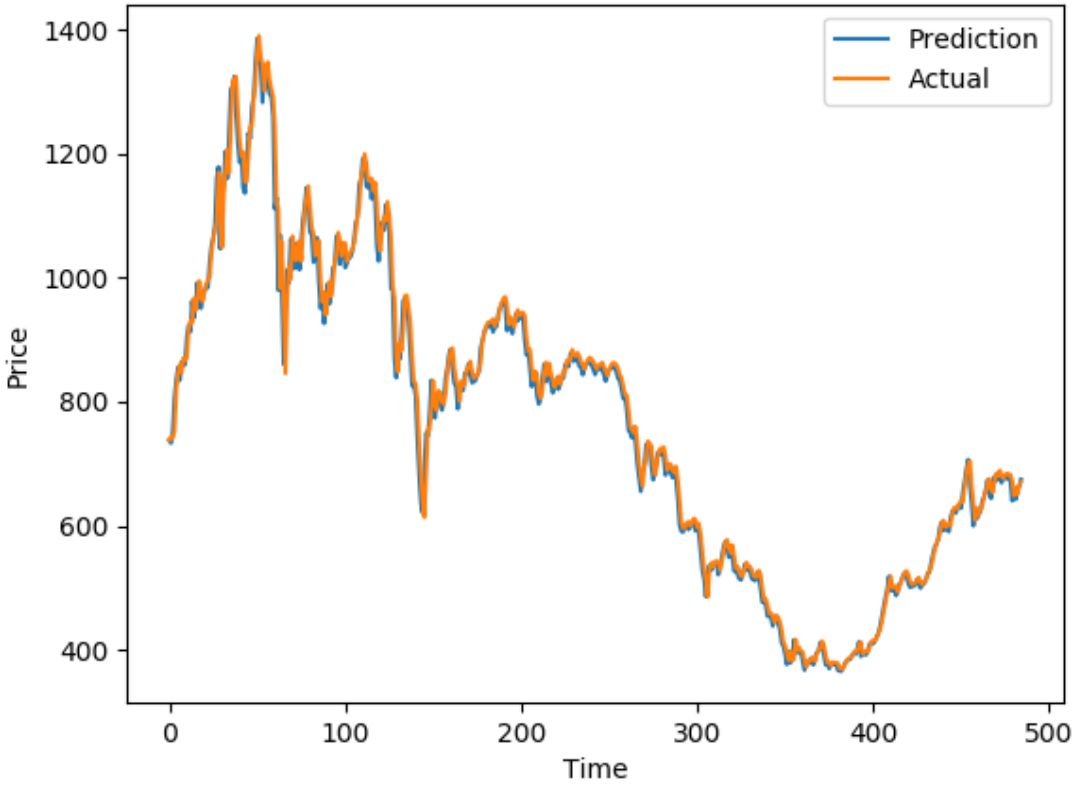

Figure 12: predict 6 hours ahead, only Ethereum training

The graph below shows the results from the neural network training on both Bitcoin and Ethereum and the predicting the prices for the 2018 data set.

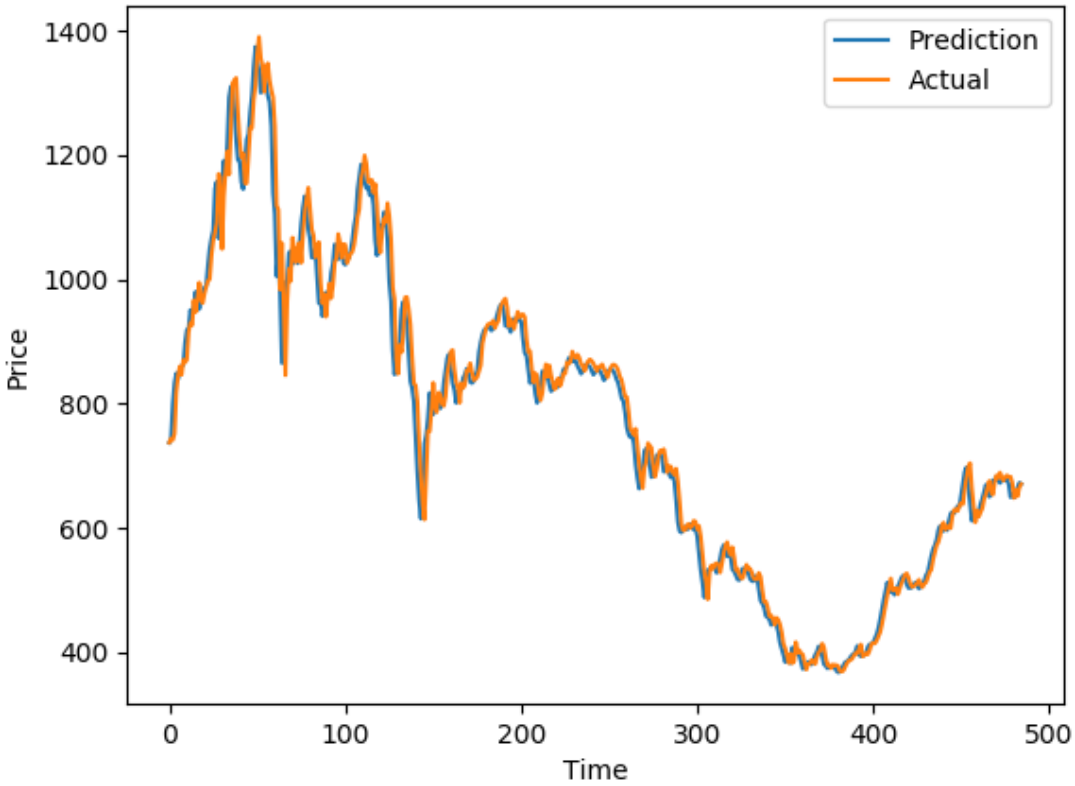

Figure 13: predict 6 hours ahead, Bitcoin & Ethereum training

The following graphs are for predictions made one day ahead at a time. These neural networks were trained on a 2017 data set where the interval was one day apart. The window size, in other word the number of data entries used to predict a single value, was two. The following results are from predicting over the 2018 data set.

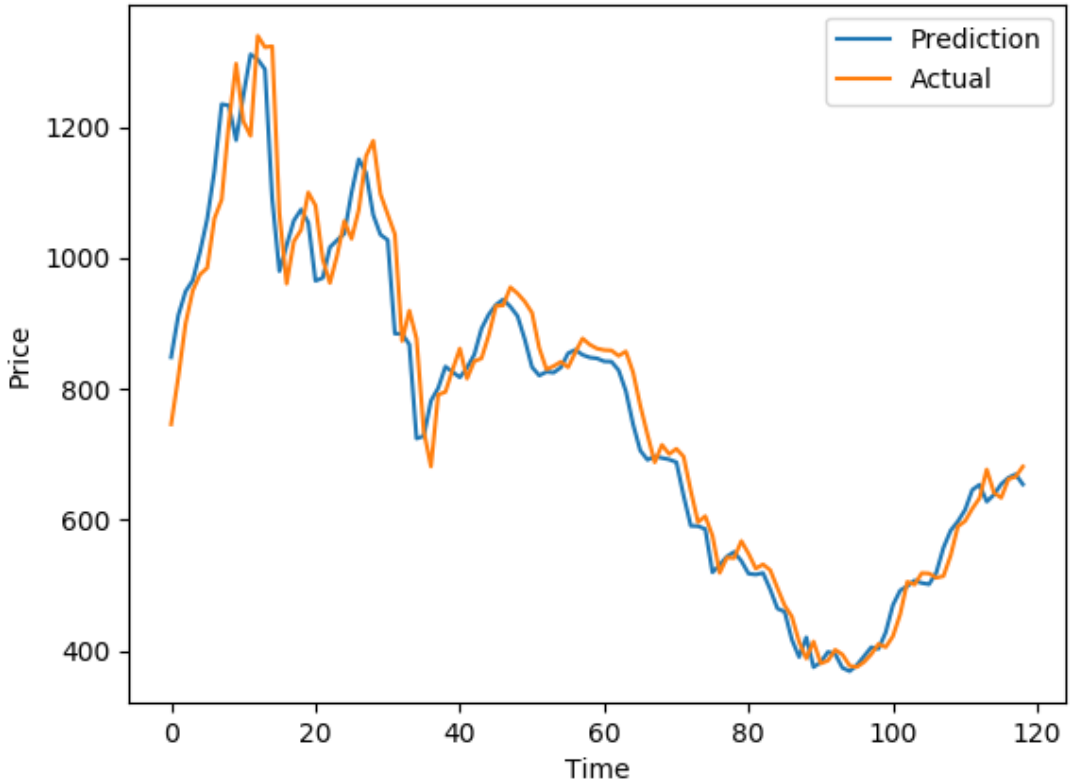

Figure 14: predict 1 day ahead, only Ethereum training

In the figure below, the predictions made by the network being trained on both cryptocurrencies can be seen.

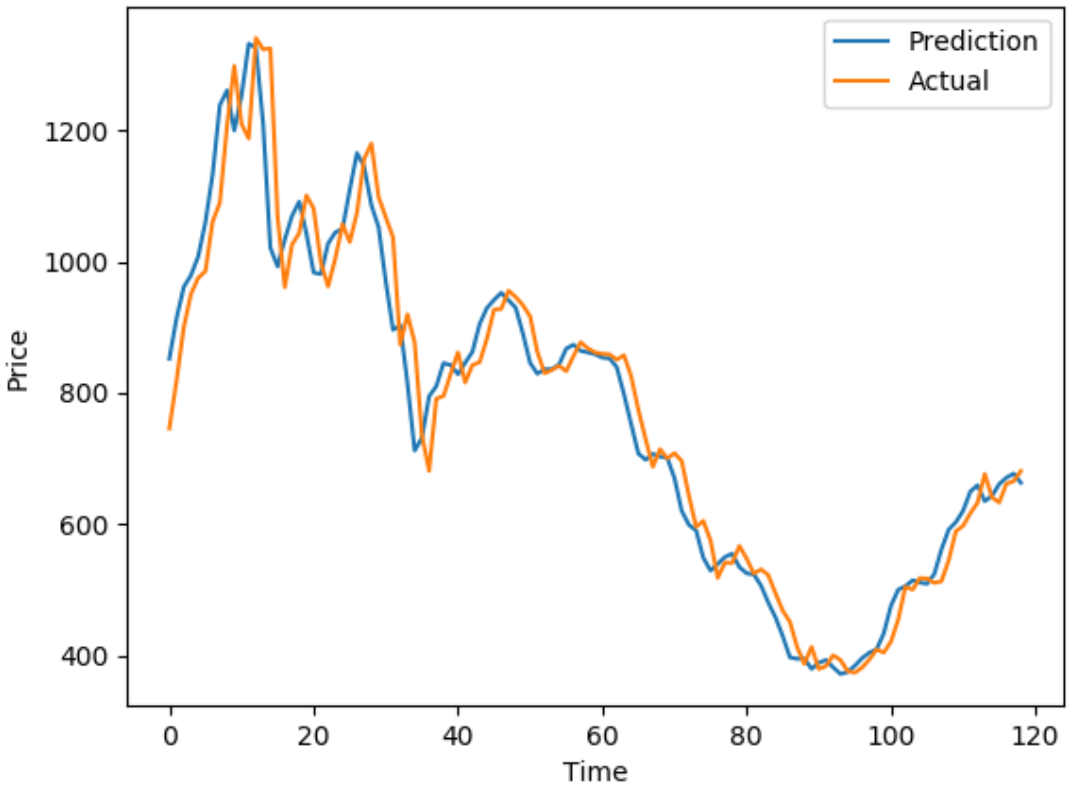

Figure 15: predict 1 day ahead, Bitcoin & Ethereum training

8.2. Comparison of loss rates

One way to measure how good a neural network is at making predictions, is by measuring the loss rate. The loss measured here is mean squared error, which is basically calculated by taking the mean of the squared error rates. The lower this is, the better the network is at predicting. All of the losses were calculated on the 2018 data set. In the table below, a comparison of the loss rates between learning on only Ethereum and on Bitcoin and Ethereum, for all neural networks can be seen. Note that all loss values were rounded up to the nearest two decimals.

	Ethereum	**Ethereum & Bitcoin**
Predict 1 hour	245.37	895.35
Predict 6 hours	640.47	1235.17
Predict 1 day	2861.07	3534.51

Table 1: Loss rates

As it can be seen in table 1, the loss rates did in fact not improve when Bitcoin was used for training along with Ethereum. As seen in the figures provided in the section, the predictions are quite accurate for both the training done with only Ethereum, and the ones done with both Bitcoin and Ethereum.

9. Discussion

In this section, the results that were presented in section 8 will be discussed. Section 9.1 will discuss the outcome of the predictions that were made by the neural networks. Section 9.2 will discuss how the inclusion of Bitcoin during training affected the prediction results.

9.1. Predictions

By looking at the graphs in figures 10-15 in the result section, a clear correlation between the actual values and the predicted values can be seen. This shows that is should in fact be possible to predict the price of Ethereum. Even though the graphs depicting predictions made with both Bitcoin and Ethereum seem less accurate than the ones with only Ethereum, both training models have resulted in good predictions.

By comparing the different graphs, it is possible to see that they become less accurate the further into the future that the predictions are made. The predictions made for one hour ahead are clearly more reliable than the ones made for a whole day ahead. This may be because of the amount of data that the neural network had to train on. The data set for training consisted of data from 2017, but with different intervals between the data entries depending on how far into the future predictions were to be made. Because of this, less data was available for the one day ahead model, than there was for the one or six-hour models. With more data it may be possible to provide better results for the one day ahead predictions. The problem with that, is that Ethereum did not really take off in price until early 2017.

The predictions that were made by the neural network that were trained on one day intervals shows that correct guesses, on where the price is headed, is made before the actual price fluctuations happens on the real price. This can be seen most clearly in figures 14 and 15, where the line representing the predicted price dives, followed by the actual price dropping just a bit later. The same can be seen in the rises of the price. Though the predictions are not as accurate in this model as the ones from the other models, the correlations are the clearest here.

Improving the parameters in the neural network may improve these graphs even further. It may be the case that more hidden layers, or more neurons in each layer, would provide a significant advantage. Other options, such as regular recurrent neural networks, may also give good results. This thesis did not set out to prove what parameters, or what type of neural network, give the best results. And therefore, it may be entirely possible that another set of these options would give other results, for better or worse.

No experiments on the amount of data needed to make a single prediction, also known as the window size, were done in this thesis. While the window size parameter that were chosen for these neural networks did turn out fine, the accuracy of the predictions may improve if more data is considered. On the other hand, looking back too far may not result in any improvements at all. Since cryptocurrencies price's is only based on what people think about that cryptocurrency at the moment, the history may not offer much detail in where the price is headed.

One of the goals of this thesis were to see if it was at all possible to predict the future prices of the cryptocurrency Ethereum, based on past market history. The graphs have shown that this is quite possible. Both in short-term, and in a longer term.

9.2. Result of different training data

In table 1, it becomes obvious that with the current implementations of the neural networks, the predictions do in fact not improve with the use of both cryptocurrencies. The use of only Ethereum provides better results than with both Bitcoin and Ethereum. This loss increase is most evident in the one hour ahead predictions, where the loss is about 3.65 times larger when using both cryptocurrencies, than when only using Ethereum. The loss increase for the six hours ahead version is about 1.93, rounded up to the nearest two decimal points. And for the one day ahead predictions, the loss increased by about 1.24 times.

There may be several reasons to why this is occurring. One reason may be that more training is needed for the neural network to realize how Bitcoin influences the price of Ethereum. Since there is more data to analyze, it may take longer for the network to figure out the connection between the two cryptocurrencies. More training or other parameters for the network may solve this, but it may also improve the predictions for the networks that look at only Ethereum if the same number of epochs were to be ran on all networks. If it takes more epochs, and therefore more computing power and time, to make the Bitcoin and Ethereum predictions as good as the predictions made on only Ethereum, it can be questioned if it is actually worth it.

Another explanation may be that there is in fact no lag. Meaning that, when a price drop, or a price spike, happens in Bitcoin, it happens to Ethereum at the exact same time. By looking at the graphs 16 and 17 below, that shows the prices of Bitcoin and Ethereum separately over 2018, taken from CoinMarketCap [11], a clear correlation between the prices can be seen. But if there is no lag it would not be possible for a neural network to make any claims on what the future Ethereum price may be, just because of Bitcoin.

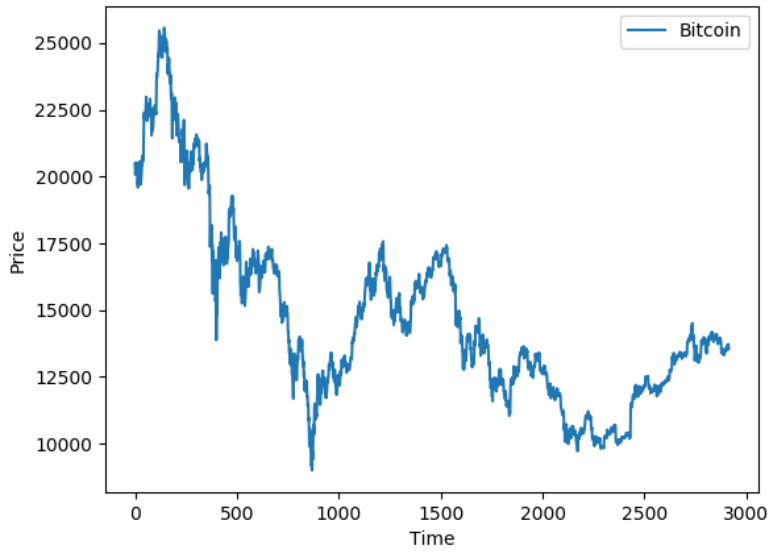

Figure 16: Bitcoin Price for 2018

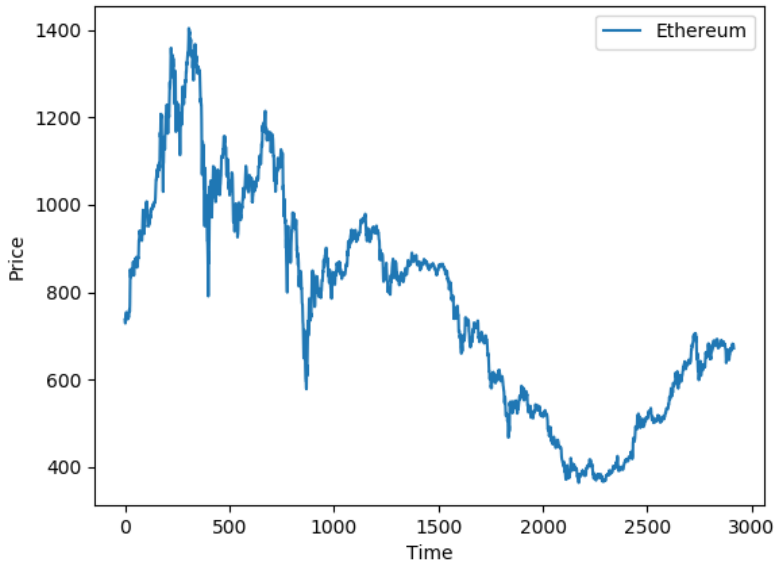

Figure 17: Ethereum Price for 2018

This may have something to do with the fact that there is nothing behind a cryptocurrency, no company or such in other terms. The value is entirely based on what users think about the cryptocurrency, and those thoughts may be based on news articles or posts in communities [16]. This may mean that when bad news hits the cryptocurrency world, all cryptocurrencies are affected at the same time, or at least with such a small lag that it is not possible for this neural network to see the change that is going to happen.

10. Conclusions

In this section, the conclusions of the results to the two questions that this thesis set out to answer, will be provided. The two questions that this thesis set out to answer were:

- Based on the past market history, is it possible to predict the future market price of Ethereum?
- Can predictions of Ethereum be improved by also looking at the past price of Bitcoin?

By looking at the past history of the cryptocurrency Ethereum, it can be concluded that it is possible to predict the future price. By analyzing the graphs 10-15, in section 8.1, a clear resembles between the actual prices and the predicted prices can be seen. The predictions that were made one hour ahead proved to be more accurate than both the predictions that were made six hours ahead, and the predictions that were made one full day ahead. While the six hours ahead predictions proved better than the full day predictions. Though it may be possible that this discrepancy in accuracy may have been caused by the decreasement of data available to the neural network.

By comparing the loss rates of the neural networks that were only trained on Ethereum, with the networks that was trained on both Bitcoin and Ethereum, it can be concluded that no improvement of the prediction accuracy was acquired by training on them both. This is made evident in table 1, in section 9.2. The graphs 10-15 in section 8.1 however, shows that the method of training on both cryptocurrencies, still gives adequate results, though not as good as when only training on only Ethereum.

11. Future Work

Future possible research on this subject extends far and wide. One area of expansion is testing the hypothesis for other cryptocurrencies rather than only for Ethereum. As the case may be that the result achieved in this paper only applies for Ethereum, more research is needed to see if the findings applies across the board. Furthermore, it is possible that cryptocurrencies other than Bitcoin could be used as the basis for the predictions for improved results. This needs further research to prove or disprove.

Since there are many different types of neural networks, it cannot be said for certain that an LSTM network is the best choice, at least that is not proven in this thesis. More research is needed to conclusively say what type should be used for this problem, and other problems alike. Neural networks such as a feed forward neural network, or a regular recurrent neural network may provide better, or worse results, than the LSTM network chosen for this thesis provides.

The settings, such as the number of hidden layers, number of neurons in each layer or for how many epochs the neural network should be trained, used in this thesis may not be optimal. More research on what the optimal settings are for this specific problem when using a LSTM neural network. Even though some research has been done in this thesis relating to how far back one should train the neural network, no certain answer has been presented. This is another area that should be explored in more depth if future research were to build upon what has already been done. An alternative here could be some sort of genetic algorithm.

The software created in this thesis work could be improved upon. It should be easier for a user to fully utilize the software, with less previous coding knowledge than what is required at the moment. In general, the UI should be more user friendly, making it easier for any user to select how far into the future they want their predictions to go. Furthermore, it should be easier to select how far back the neural network should go when training. This can even extend so far as to let the user choose which cryptocurrencies they want to make predictions on.

The chosen architect of the neural network model should be research in more depth. LSTM networks may not be the optimal artificial neural network for work like this. The case may be that regular recurrent neural network, or a simple feed forward neural network, perform better for this specific task. Some research has been done in this area, as explained in the related works section, but no studied research has tested enough architectural model to accurately provide proof of what the optimal choice is. Future research could try to examine several architectures with the goal of finding the optimal one for problems specifically like this.

12. References

[1] S. Dziembowski, "Introduction to Cryptocurrencies," presented at the Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, Colorado, USA, 2015. Available: https://dl.acm.org/citation.cfm?doid=2810103.2812704

[2] R. Böhme, N. Christin, B. Edelman, and T. Moore, "Bitcoin: Economics, Technology, and Governance," *Journal of Economic Perspectives*, vol. 29, no. 2, pp. 213-38, 2015.

[3] S. McNally, "Predicting the price of Bitcoin using Machine Learning," Dublin, National College of Ireland, 2016.

[4] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, "Advanced neural network training methods for low false alarm stock trend prediction," in *Neural Networks, 1996., IEEE International Conference on*, 1996, vol. 4, pp. 2021-2026 vol.4.

[5] J. Zobel, *Writing for computer science*. Springer, 2015.

[6] Y. M. Kow and X. Ding, ""Hey, I know what this is!": Cultural Affinities and Early Stage Appropriation of the Emerging Bitcoin Technology," presented at the Proceedings of the 19th International Conference on Supporting Group Work, Sanibel Island, Florida, USA, 2016. Available: https://dl.acm.org/citation.cfm?doid=2957276.2957279

[7] A. Hayes, "What factors give cryptocurrencies their value: An empirical analysis," 2015.

[8] A. Serapiglia, C. P. Serapiglia, and J. McIntyre, "Crypto currencies: core information technology and information system fundamentals enabling currency without borders," *Information Systems Education Journal*, vol. 13, no. 3, p. 43, 2015.

[9] J. Lansky, "Possible State Approaches to Cryptocurrencies," *Journal of Systems Integration*, vol. 9, no. 1, p. 19, 2018.

[10] A. S. Hayes, "Cryptocurrency value formation: An empirical study leading to a cost of production model for valuing bitcoin," *Telematics and Informatics*, vol. 34, no. 7, pp. 1308-1321, 2017/11/01/ 2017.

[11] "All Cryptocurrencies | CoinMarketCap", Coinmarketcap.com, 2018. [Online]. Available: https://coinmarketcap.com/all/views/all/. [Accessed: 08- Apr- 2018].

[12] J. Bouoiyour and R. Selmi, "What Does Crypto-currency Look Like? Gaining Insight into Bitcoin Phenomenon," 2014.

[13] L. Catania, S. Grassi, and F. Ravazzolo, "Predicting the Volatility of Cryptocurrency Time–Series," 2018.

[14] S. Y. Yang and J. Kim, "Bitcoin Market Return and Volatility Forecasting Using Transaction Network Flow Properties," in *2015 IEEE Symposium Series on Computational Intelligence*, 2015, pp. 1778-1785.

[15] M. Polasik, A. I. Piotrowska, T. P. Wisniewski, R. Kotkowski, and G. Lightfoot, "Price Fluctuations and the Use of Bitcoin: An Empirical Inquiry," *International Journal of Electronic Commerce*, vol. 20, no. 1, pp. 9-49, 2015/09/15 2015.

[16] Y. B. Kim *et al.*, "Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies," *PLOS ONE*, vol. 11, no. 8, p. e0161197, 2016.

[17] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[18] I. Aleksander and H. Morton, *An Introduction to Neural Computing*. International Thomson Computer Press, 1995.

[19] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4-22, 1987.

[20] "Activation Functions - Tanh vs. Sigmoid", Orngunnarsson.blogspot.se, 2018. [Online]. Available: https://orngunnarsson.blogspot.se/2017/04/activation-functions-tanh-vs-sigmoid.html. [Accessed: 18- May- 2018].

[21] S. Sharma, "What the Hell is Perceptron? – Towards Data Science", Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53. [Accessed: 22- May- 2018].

[22] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1456-1470, 1998.

[23] K. Kamijo and T. Tanigawa, "Stock price pattern recognition-a recurrent neural network approach," in *1990 IJCNN International Joint Conference on Neural Networks*, 1990, pp. 215-221 vol.1.

[24] A. Birgul Egeli, "Stock market prediction using artificial neural networks," *Decision Support Systems*, vol. 22, pp. 171-185, 2003.

[25] S. Johnson, "Drawing an unfolded recurrent neural network", TeX - LaTeX Stack Exchange, 2018. [Online]. Available: https://tex.stackexchange.com/questions/364413/drawing-an-unfolded-recurrent-neural-network. [Accessed: 20- May- 2018].

[26] "Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs", WildML, 2018. [Online]. Available: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/. [Accessed: 18- May- 2018].

[27] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310-1318.

[28] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115-143, 2002.

[29] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Berlin Heidelberg, 2012.

[30] "Long short-term memory", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory. [Accessed: 18- May- 2018].

[31] M. Matta, I. Lunesu, and M. Marchesi, "Bitcoin Spread Prediction Using Social and Web Search Media," in *UMAP Workshops*, 2015.

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

[33] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," in *2017 Intelligent Systems Conference (IntelliSys)*, 2017, pp. 905-913.

[34] H. Jang and J. Lee, "An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information," *IEEE Access*, vol. 6, pp. 5427-5437, 2018.

[35] A. Greaves and B. Au, "Using the Bitcoin Transaction Graph to Predict the Price of Bitcoin," *No Data*, 2015.

[36] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka, "Stock market prediction system with modular neural networks," in *1990 IJCNN International Joint Conference on Neural Networks*, 1990, pp. 1-6 vol.1.

[37] "TensorFlow", TensorFlow, 2018. [Online]. Available: https://www.tensorflow.org. [Accessed: 08- Apr- 2018].

[38] "Keras Documentation", Keras.io, 2018. [Online]. Available: https://keras.io/. [Accessed: 08- Apr- 2018].