FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

# Assignment of bachelor's thesis

| | |
|---|---|
| **Title:** | Analyze and reconstruct order book to develop a model to predict cryptocurrency assets price. |
| **Student:** | Mikhail Lyashenko |
| **Supervisor:** | Ing. Stanislav Kuznetsov |
| **Study program:** | Informatics |
| **Branch / specialization:** | Knowledge Engineering |
| **Department:** | Department of Applied Mathematics |
| **Validity:** | until the end of summer semester 2022/2023 |

## Instructions

The purpose of this work is to analyze the reconstructed order book to develop a model for predicting cryptocurrency assets prices.

1. Introduction to cryptocurrency assets and choosing the assets for our experiments.
2. Reconstructing of order flows matching of the order books.
3. Analyzing and selecting the best approach of reconstructing order flows with minimum loss of information about the market.
4. Preprocessing of order flows and prepare datasets.
5. Build models based on neural networks, such as LSTM and GRU.
6. Provide experiments and evaluate models.
7. Test solution on online market prediction.

---

*Electronically approved by Ing. Karel Klouda, Ph.D. on 6 September 2021 in Prague.*

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

# Analyze and reconstruct order book to develop a model to predict cryptocurrency assets price.

***Mikhail Lyashenko***

Department of Applied Mathematics
Supervisor: Ing. Stanislav Kuznetsov

May 12, 2022

# Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 12, 2022 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology


## Citation of this thesis

Lyashenko, Mikhail. *Analyze and reconstruct order book to develop a model to predict cryptocurrency assets price.*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

# Abstrakt

Tato práce analyzuje informace uvedené na trhu s kryptoměnami, zkoumá, které nejlépe funguje při predikci časových řad dat na trhu: LSTM-RNN nebo GRU- RNN a pokouší se předpovídat změnu ceny na trhu u tří párů kryptoměn: BTC/USDT, ETH/USDT a DOGE/USDT.
Hlavní přínosy práce jsou zejména následující:

1. Analyzuje trh kryptoměn a identifikuje některé silné rysy knih příkazů,
2. Ukazuje skutečný potenciál RNN při predikci časových řad,
3. Zdůrazňuje skutečný význam dat, na kterých jsou modely trénovány.

**Klíčová slova** Limit Order Book, Transakce, RNN, LSTM, GRU, DNN

# Abstract

This thesis analyzes the information given in the cryptocurrency market, explores which works best in time series prediction of the data in the market: LSTM-RNN or GRU-RNN and attempts to predict price change in the market for three cryptocurrency pairs: BTC/USDT, ETH/USDT and DOGE/USDT.

In particular, the main contributions of the thesis are as follows:

1. Analyzes cryptocurrency market and identifies some strong features of limit order books,
2. Shows a true potential of RNNs in time series prediction,
3. Emphasizes a true importance of the data, on which models are trained.

**Keywords** Limit Order Book, Transactions, RNN, LSTM, GRU, DNN

# Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

In this chapter, we go through motivation behind the thesis, introduce the reader to the overall theme, state the problem and define the goals.

## 1.1 Motivation

Motivation behind this thesis is to inspire anyone, who is reading this thesis, to make researches in the world of cryptocurrency as a whole and continue the work of predicting the price of cryptocurrencies. This is an ambitious project, considering its relative novelty. It may not have given the results, that were hoped for, but it surely gave a contribution for future work.

## 1.2 Introduction to the theme and problem Statement

As the world evolves, also does its financial side. In the year 2009, the first decentralized was introduced by pseudonymous developer Satoshi Nakamoto. It was Bitcoin, which now has an estimated market capitalization of 747 billion dollars[1]. At first decentralization, that was brought by , was what interested people about it. Removing enormous control of banks and huge organization. It is the technology, that also has a great security, which increases, as more people get involved. Then in the year 2011, a first altcoin[2] was born, called Namecoin. Compared to Bitcoin, many altcoins brought a completely different spectrum of functionality, from faster transactions to technology platforms intended to help manage assets and data easily and inexpensively. One of the first altcoins and now the second biggest  in terms of market capitalization is Ethereum[3], which is most popular

[1] Bitcoin market capitalization on April 2022.

[2] Alternative .

[3] Ethereum market capitalization on April 2022 - 348 billion dollars.

because of its smart contracts functionality. One of the phenomenons in financial terms are memecoins, which usually has some humorous background and does not have any utility. Dogecoin is one of them, which takes the $10^{\text{th}}$ place in terms of market capitalization[4]. The question stands: is it possible to predict prices of , as a fairly new thing? Bitcoin, Ethereum and Dogecoin, these three completely different are going to a part of this research. Machine learning is used for making predictions, in particular recurrent neural networks, such as LSTM and GRU. The research is going to be made solely on the market. Order flow is going to be the key thing in this thesis: decision stands between limit order books and executed transaction, what works best for the prediction?

## 1.3 Goals of the Bachelor Thesis

1. Collect data and analyze it.
2. Build LSTM and GRU to predict future volumes.
3. Build deep neural network to predict price based on volumes.
4. Combine step 2. and 3. in a pursue to predict price based on predicted volumes.

## 1.4 Structure of the Bachelor Thesis

The thesis is organized into chapters as follows:

1. *Introduction*: Gives a gentle introduction to the subject and set out goals of the thesis.
2. *Background and State-of-the-Art*: Introduces the necessary theoretical background and surveys the current state-of-the-art.
3. *Overview of Our Approach*: States approach, used in the thesis.
4. *Main Results*: Describes results main results, considering volume prediction and results, considering future price predictions.
5. *Conclusions*: Summarizes results received, designates contribution of the thesis and proposes approaches for future research.

[4]Dogecoin market capitalization on April 2022 - 2 billion dollars.

Chapter 2

# Background and State-of-the-Art

In this chapter two main parts are mentioned. In theoretical background is gathered all the information needed to understand the subject of the work. In Previous Results and Related Work will be mentioned all the articles, which helped in writing of this work.

## 2.1 Theoretical Background

### 2.1.1 Artificial Neural Networks

Time-series data is a consecutive collection of observation made in time. Order flow in this work is meant as an overall information in the market, meaning both limit order books and executed transactions. Limit order book is a collection of buy and sell orders on different levels, meaning the highest buy order is on the top of buy orders and the lowest sell order is on the bottom of sell orders. Limit order book is update every tick, which is the measure of minimum upward or downward movement in the price of an asset. Limit order books are believed to be the makers of the price, because the price lays between the highest buy order and the lowest sell order.
Recurrent neural network (RNN) [1] is a class of artificial neural networks, which characteristic feature is the transfer of signals from the output layer or hidden layer to the input layer. Basic architecture of an RNN can be seen on the Figure 2.1.

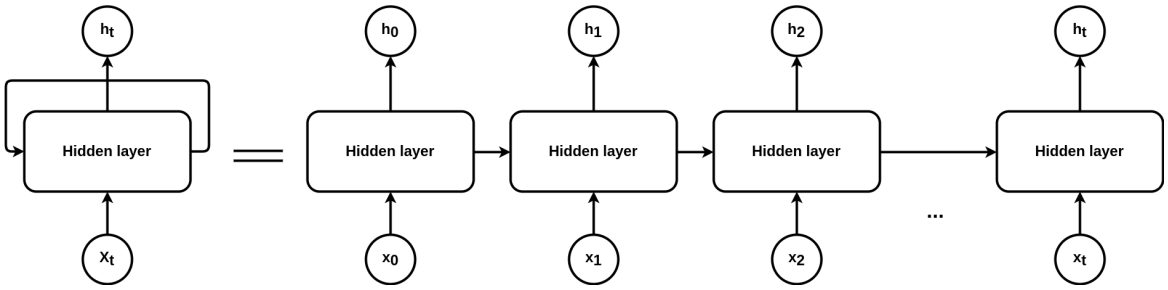



Figure 2.1: Basic architecture of RNN.

RNN is commonly used for time series predictions. Backward propagation of errors is used to train the model, it defines a strategy for selecting neural network's weights using gradient optimization methods. The task is to solve the problem of minimizing total losses on all the training samples:

$$Q(W) = \sum_{t=0}^{n-1} L_t(W) \to \min_W, \tag{2.1}$$

where:

- $W$ - weight vector of a neural network
- $n (n \in \mathbb{N}_0)$ - number of training samples
- $L_t(W)$ - loss function on sample $x_t$ of index $t$

On each step $t$ of learning neural network is given an input vector $x_t$, for which an output $y_t$ is expected. The weights for neurons of the hidden layer (or hidden layers) are calculated for the input vector $x_t$. Output vector $k_t (k_t \in R^n)$ of the hidden layer is calculated like this on each step $t$:

$$k_t = \sigma_k(W_x \cdot x_t + W_k \cdot k_{t-1} + b_k) \tag{2.2}$$

where:

- $x_t (x_t \in \mathbb{R}^L)$ is an input vector on each step $t$,
- $W_x (W_x \in \mathbb{R}^{L \times K})$ is a matrix of weights for the input vector,
- $W_y (W_y \in \mathbb{R}^{K \times K})$ is a matrix of weights for the hidden layer,
- $b_k (b_k \in \mathbb{R}^K)$ is a shift vector,
- $\sigma_k$ is an activation function for the hidden layer.

| **Function name** | **Function** | **Value Range** |
|---|---|---|
| Sigmoid | $g(z) = \frac{1}{1+e^{-z}}$ | $(0, 1)$ |
| Tanh | $g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$ | $(-1, 1)$ |
| ReLU | $g(z) = max(0, z)$ | $(0, +\infty)$ |
| LeakyReLU | $g(z) = max(0.1 \cdot z, z)$ | $(-\infty, +\infty)$ |

Table 2.1: Table of activation functions.

There are four commonly used activation functions: Sigmoid, Tanh, ReLU and ReLU alternative, called LeakyReLU, which are depicted in Table 2.1. The memory implemented in RNN network is short, because at each learning step information in the memory is mixed with new information, that way it is completely overwritten after a few steps.

Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) is a particular type of a recurrent neural network. However, simple recurrent neural networks can only account for the network's recent past states. This problem is called the problem of long-run dependence or the vanishing gradient problem. In theory, this problem can be solved by choosing the right network parameters, but the problem of choosing these parameters is still unsolved. LSTM networks are designed to solve that kind of problem and allow to detect both long and short data patterns and partially eliminate the problem of vanishing gradient. LSTM architecture in extended view can be seen on the Figure 2.2 [2].

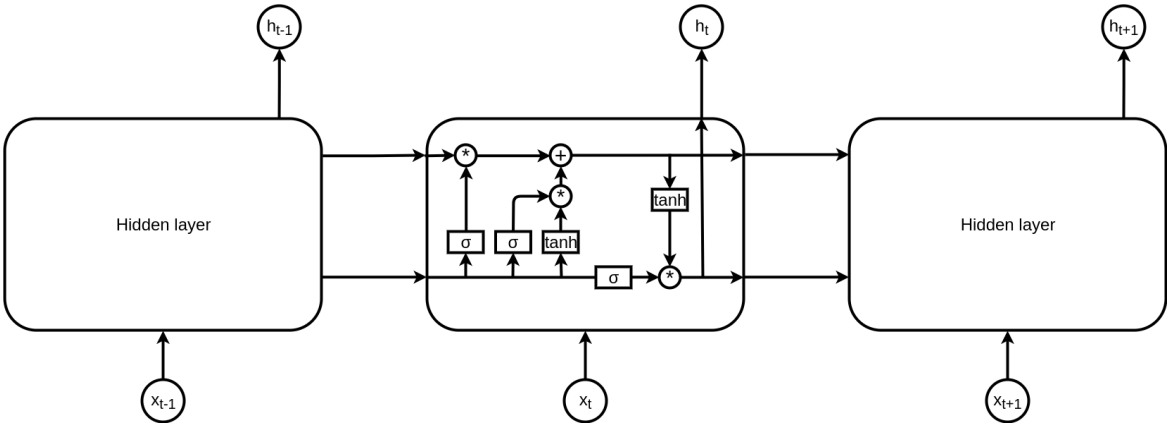



Figure 2.2: LSTM architecture, including cell view.

LSTM cell works like this:

1. Lower horizontal line on Figure 2.2 given as an input for a cell represents previous hidden state $h_{t-1}$. It proceeds as an input for a sigmoid function, that works as forget filter. Output is given as one of the values: 0 or 1, where 0 means "forget". This part of LSTM cell is called a **Forget Gate**. The equation for **Forger Gate** is:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{2.3}$$

where:

- $t$ - timestep
- $f_t (f_t \in \mathbb{R}^K)$ - **Forget Gate** at t
- $x_t (x_t \in \mathbb{R}^L)$ - input vector
- $h_{t-1} (h_{t-1} \in \mathbb{R}^K)$ - previous hidden state
- $W_f (W_f \in \mathbb{R}^{K \times L})$ - Weight matrix for input gate $x_t$
- $U_f (U_f \in \mathbb{R}^{K \times K})$ - Weight matrix for previous hidden state $h_{t-1}$
- $b_f (b_f \in \mathbb{R}^K)$ - connection bias

2. Next sigmoid function, which gets $x_t$ as an input, decides what values should be changed. Tanh function makes a vector of new values, that could be added to a new cell state $C_t$, which is represented by the upper horizontal line on Figure 2.2. This part of LSTM cell is called an **Input Gate**. It produces two values:

a)
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{2.4}$$

b)
$$\tilde{C}_t = tanh(W_C x_t + U_C h_{t-1} + b_C), \tag{2.5}$$

where:

- $t$ - timestep
- $i_t (i_t \in \mathbb{R}^K)$ - **Input Gate** at t
- $W_i (W_i \in \mathbb{R}^{L \times K})$ and $W_C (W_C \in \mathbb{R}^{L \times K})$ - weight matrices for input vector $x_t$
- $U_i (U_i \in \mathbb{R}^{K \times K})$ and $U_C (U_C \in \mathbb{R}^{K \times K})$- weight matrices for previous hidden state $h_{t-1}$
- $\tilde{C}_t$ - value generated by tanh
- $b_i (b_i \in \mathbb{R}^K)$ and $b_C (b_C \in \mathbb{R}^K)$ - connection biases

3. To update the old state the output from the **Forget Gate** $f_t(f_t \in \mathbb{R}^K)$ is multiplied with the the old state $C_{t-1}(C_{t-1} \in \mathbb{R}^K)$ and then its added with the multiplies outputs from the "Input Gate": $i_t(i_t \in \mathbb{R}^K)$ and $\tilde{C}_t(\tilde{C}_t \in \mathbb{R}^K)$. This produces a new LSTM state $C_t(C_t \in \mathbb{R}^K)$.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.6}$$

4. Output from the previous LSTM cell together with the input $x_t$ goes to sigmoid function and gets multiplied with the tanh function of the new state, which produces the output of the current LSTM cell. This part of the cell is called the **Output Gate**. In equations:

   a)
   $$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{2.7}$$

   b)
   $$h_t = o_t * tanh(C_t), \tag{2.8}$$

   where:

   - $t$ - timestep
   - $o_t(o_t \in \mathbb{R}^K)$ - **Output Gate** at $t$
   - $W_o(W_i \in \mathbb{R}^{K \times L})$ - weight matrix for input vector $x_t$
   - $U_o(U_o \in \mathbb{R}^{K \times K})$ - weight matrix for previous hidden state $h_{t-1}$
   - $b_o(b_o \in \mathbb{R}^K)$ - connection biases
   - $C_t(C_t \in \mathbb{R}^K)$ - new cell state

Gate recurrent network, in comparison with LSTM, does not have states and consists of only two gates: **Reset Gate** and **Update Gate**. The cell architure of GRU is given on the Figure 2.3.

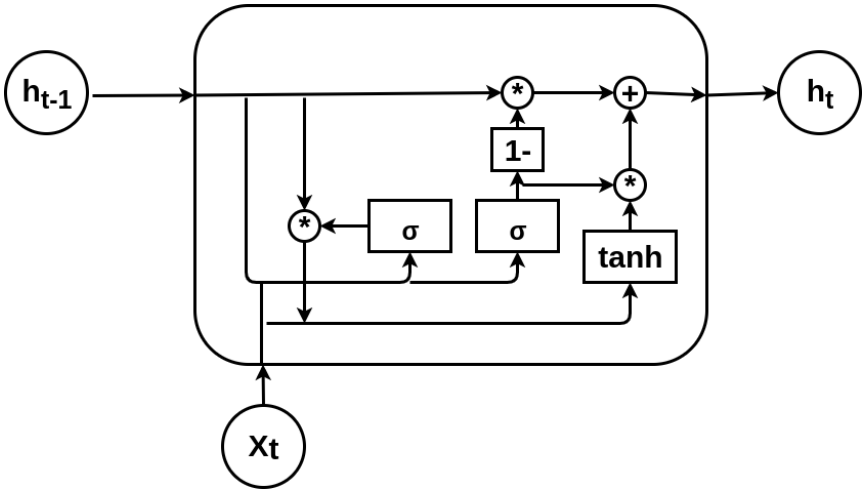



Figure 2.3: GRU cell architecture.

GRU cell works like this:

1. Input vector together with the output vector from the previous cell is fed to sigmoid function. This is the result of the **Update Gate**. In equation:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{2.9}$$

where:

- $t$ - timestep
- $z_t (z_t \in \mathbb{R}^K)$ - **Update Gate** at $t$
- $x_t (x_t \in \mathbb{R}^L)$ - input vector
- $h_{t-1} (h_{t-1} \in \mathbb{R}^K)$ - output vector from previous cell
- $W_z (W_z \in \mathbb{R}^{K \times L})$ - weight matrix for input vector $x_t$
- $U_z (U_z \in \mathbb{R}^{K \times K}$ - weight matrix for output vector from previous cell $h_{t-1}$
- $b_z (b_z \in \mathbb{R}^K$ - connection bias

2. Similar to the previous step is made in the **Reset Gate**.

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \tag{2.10}$$

where:

a) $r_t (r_t \in \mathbb{R}^K)$ - **Reset Gate** at $t$

b) $W_r (W_r \in \mathbb{R}^{K \times L})$ and $U_r (U_r \in \mathbb{R}^{K \times K})$ - weight matrices

c) $b_r$ - connection bias

3. Then an intermediate value $\tilde{h_t}$ of output vector is counted, which works as a decision maker in term of what information is to be discarded from the previous step.

$$\tilde{h_t} = tanh(W_h x_t + r_t \odot U_h h_{t-1} + b_h), \tag{2.11}$$

where:

- $\tilde{h_t} (h_t \in \mathbb{R}^K)$ - intermediate value of output vector
- $W_h (W_h \in \mathbb{R}^{K \times L})$ and $U_h (U_h \in \mathbb{R}^{K \times K})$ - weight matrices
- $b_h$ - connection bias

4. Output vector $h_t (h_t \in \mathbb{R}^K)$ is counted using intermediate value of output vector and output of the **Update Gate**:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h_t}, \tag{2.12}$$

Deep neural networks (DNN) can be used both for regression and classification problems. Regression problem is the the problem, when linear value is expected as an output and classification problem is when a set of predetermined values is expected.

### 2.1.2 Optimizer

Adaptive Momentum Estimation (Adam) [3] is going to be used in a final stage in a training of the models. It is an optimization algorithm for gradient descent. In contrast to the classical stochastic gradient decent, it does not maintain a single learning rate during training, but adapts. Adam algorithm can be described as a combination of Adaptive Gradient Algorithm (AdaGrad) [4] and Root Mean Square Propagation (RMSProp) [5]. In RMSProp learning rates are adapted based upon the first moment (mean), whereas in Adam the second moment of the gradients (the uncentered variance) is also used. The formulas used in Adam optimizer are:

1.
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{2.13}$$

2.
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{2.14}$$

where:

- $m_t$ - moment, which helps in faster convergence of the loss function
- $v_t$ - second moment
- $\beta_1$ and $\beta_2$ - decay rates
- $g_t$ - gradient function

Usually $m_t$ and $v_t$ are initialized by zero vectors. To solve this kind of a problem following upgrades are made:

1.
$$\tilde{m_t} = \frac{m_t}{1 - \beta_1}, \tag{2.15}$$

2.
$$\tilde{v_t} = \frac{v_t}{1 - \beta_2} \tag{2.16}$$

The final parameter update formula look like this:

$$\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt{\tilde{v_t}} + \epsilon} \tilde{m_t}, \tag{2.17}$$

where:

- $\mu$ - learning rate
- $\epsilon$ - a small constant to avoid division by 0

### 2.1.3 Stationarity

Time-series data can be stationary or non-stationary. Members of non-stationary data are dependent on each other, meaning that the trend can be extracted from the data. To check the data stationarity Augmented Dickey Fuller (ADF) [6] test is used. To understand ADF, firstly unit root test should be observed. Having an autoregressive model:

$$y_t = \sum_{i=0}^{n-1} a_i y_{t-i} + \varepsilon_t \tag{2.18}$$

If all $a_i$ are equal to 1, then the autoregressive process is considered to be non-stationary. Otherwise if each $|a_i|$ is less, than 1, then the autoregressive process is stationary.
The autoregressive equation given before can be written like this:

$$\Delta y_t = \delta y_{t-1} + \varepsilon_t, \tag{2.19}$$

where:

- $\delta = a - 1$
- $\lambda y_t = y_t - y_{t-1}$

Three versions of a Dickey-Fuller test exist:

1. Without a constant and a trend:

$$\Delta y_t = \delta y_{t-1} + \varepsilon_t \tag{2.20}$$

2. With constant, without trend:

$$\Delta y_t = \alpha + \delta y_{t-1} + \varepsilon_t \tag{2.21}$$

3. With constant and linear trend:

$$\Delta y_t = \alpha + \gamma t + \delta y_{t-1} + \varepsilon_t \tag{2.22}$$

The null hypothesis stays the same: if $b$ is equal to 0, then it refers to non-stationarity. For each of the tested regressions corresponding critical values of DF-statistic exist. If DF-statistic lies to the left of the critical value (critical values are negative), then null hypothesis is rejected and and the process is considered stationary.
Augmented Dickey-Fuller test is a slight alternation of classic Dickey-Fuller test, one more term is included $\beta \Delta y_{t-1}$. So the equation with constant and without trend would look like this:

$$\Delta y_t = \alpha + \delta y_{t-1} + \beta y_{t-1} + \varepsilon_t \tag{2.23}$$

## 2.2 Previous Results and Related Work

One of the RNNs applications is time series prediction. There are works, that try to predict future cryptocurrency's price, based on the price itself[7]. Despite the fact, that some of this works get pretty solid results, this method is not going to be used in this thesis, because studies [8] show, that there is a significant amount of information stored in Limit Order Books. In the article [9] limit order books are used for prediction of stock price movements. LSTM gets a solid F1-score[5], which fluctuates between 61 and 66 percent, depending on the value of prediction horizon. In the course of the work, time series data are going to be used. In the book [10] the author says:
*"If we fit a stationary model to data, we assume our data are a realization of a stationary process. So our first step in an analysis should be to check whether there is any evidence of a trend or seasonal effects and, if there is, remove them."*
That is why all time series are going to be checked on stationarity and if they are not stationary a method from the book "Forecasting: principles and practice" [11] called differencing is going to be applied on the dataset to reduce trend and seasonality. Before feeding the input to the neural network it is strongly advised to scale the data. The author of "Neural Networks for Pattern Recognition" [12] says:
*"In practice it is nearly always advantageous to apply pre-processing transformations to the input data before it is presented to a network. Similarly, the outputs of the network are often post-processed to give the required output values."*
That is why normalization is applied to input and output. During training the neural networks some regularization techniques are used. In the journal paper "Dropout: A Simple Way to Prevent Neural Networks from Overfitting"[13] the author describes how much of an improvement gave the Dropout method, during which some nodes are dropped. As another regularization method Batch normalization is used, in the book "Deep Learning" [1] the author emphasizes, that it *"can have a dramatic effect on optimization performance"*. In the deep learning models used further in the work RuLU activation functions are used, as the author of a previously mentioned book [1] highlights the fact, that in modern neural networks it is recommended to use ReLU as an activation function.

[5]F1-score is a harmonic mean of the precision and recall.

Chapter 3

# Overview of Overall Approach

## 3.1 Collection of data

All the data were collected from the Binance Exchange Api. Data were collected for three cryptocurrencies: Bitcoin(BTC), Ethereum(ETH) and Dogecoin(DOGE). All were collected in relation to the stablecoin Tether(USDT), which is made as an equivalent of 1 US dollar. Data were collected from midnight of 29.01.2022 to midnight of 05.02.2022, a total of seven days. For each pair BTC/USDT, ETH/USDT, DOGE/USDT were collected their prices, limit order books and executed transactions. This information was collected in time interval of approximately 30 seconds. PostgreSQL was used as the database for storing the data.
In the Table of prices of cryptocurrencies 3.1 there are three columns:

- **time** - time, when data in the corresponding row were collected,
- **price** - the actual price at **time**,
- **symbol** - cryptocurrency's pair abbreviation

| | price | symbol | time |
|---|---|---|---|
| **0** | 37652.92 | BTCUSDT | 2022-01-29 00:00:58 |
| **1** | 2552.12 | ETHUSDT | 2022-01-29 00:04:29 |
| **2** | 2562.01 | ETHUSDT | 2022-01-29 00:07:59 |
| **3** | 37877.25 | BTCUSDT | 2022-01-29 00:10:59 |
| **4** | 37928.22 | BTCUSDT | 2022-01-29 00:14:29 |

Table 3.1: Prices.

In the Table of limit order books 3.2 there is a total of 7 columns:

- **time** - time, when data in the corresponding row were collected
- **lastUpdatedId** - auto-generated id by Binance Exchange for the corresponding data row
- **side** - stores one of two values bid/ask, shows, if this was an order to sell(ask) or an order to buy(bid)
- **symbol** - cryptocurrency's pair abbreviation,
- **level** - how deep is the order placed in the limit order book at **time**,
- **price** - price, which was bid/ask,
- **VELOCITY** - volume of cryptocurrency bid/ask at **price**

| | lastUpdateId | price | VELOCITY | side | time | symbol | level |
|---|---|---|---|---|---|---|---|
| **0** | 14294173908 | 2542.32 | 0.6191 | bid | 2022-01-29 00:00:19 | ETHUSDT | 57 |
| **1** | 14294173908 | 2547.48 | 0.8196 | ask | 2022-01-29 00:00:19 | ETHUSDT | 57 |
| **2** | 14294173908 | 2542.28 | 0.4104 | bid | 2022-01-29 00:00:19 | ETHUSDT | 58 |
| **3** | 14294173908 | 2547.52 | 5.3882 | ask | 2022-01-29 00:00:19 | ETHUSDT | 58 |
| **4** | 14294173908 | 2542.26 | 0.1104 | bid | 2022-01-29 00:00:19 | ETHUSDT | 59 |

Table 3.2: Limit order books.

| | id | price | qty | quoteQty | time | isBuyerMaker | isBestMatch | symbol |
|---|---|---|---|---|---|---|---|---|
| 0 | 1239975947 | 37716.57 | 0.00086 | 32.436250 | 2022-01-29 00:00:07.804 | true | true | BTCUSDT |
| 1 | 1239975948 | 37716.57 | 0.00078 | 29.418925 | 2022-01-29 00:00:08.266 | true | true | BTCUSDT |
| 2 | 1239975949 | 37716.58 | 0.00076 | 28.664601 | 2022-01-29 00:00:08.273 | false | true | BTCUSDT |
| 3 | 1239975950 | 37716.57 | 0.00586 | 221.019100 | 2022-01-29 00:00:08.296 | true | true | BTCUSDT |
| 4 | 1239975951 | 37716.57 | 0.00086 | 32.436250 | 2022-01-29 00:00:08.296 | true | true | BTCUSDT |

Table 3.3: Executed transactions.

Transaction are executed automatically by Binance Exchange. It means, that if for a bid order is found a match on the side of ask, in terms of the same price, the transactions is automatically executed. In reality, it is not possible to find two orders on different sides with the same price and same volume, that is why orders can be partially executed. In the Table of executed transactions 3.3 8 columns are represented:

- **time** - time, when data in the corresponding row were collected,
- **symbol** - cryptocurrency's pair abbreviation,
- **price** - price of 1 unit of cryptocurrency at **time**,
- **id** - auto-generated id by Binance Exchange for the corresponding data row,
- **isBuyerMaker** - strores true or false, true, if row represents buyer's side, false - seller's side,
- **isBestMatch** - transaction is executed for the reason of finding the best match on the other side of limit order book. True, by default,
- **qty** - volume of transactions, how much of cryptocurrency is bought/sold
- **quoteQty** - **qty** $\times$ **price**

## 3.2 Analyzing and aggregating data

This part's main purpose is to find correlation between price and limit order book/transactions information, depending on what gives the best result.
The amount of collected data is too big to read it all at once to code memory. That is why data is read from the database in batches. One batch is representing one day. Groupers were applied on each table. Data were grouped by time, time intervals: 15min, 60min, 120min, 180min, 240min, 300min, 360 min. Additionally, for each table were made their specific applies.
For limit order books table a total of 4 applies were created. The main thought behind all of them is that theoretically, if bid side dominates ask side in the limit order book, the

price should go up, and it should go down otherwise. One of the main feature of limit order books is that it stores orders on multiple levels, which can give a significant amount of information. This is how these applies work:

1.
$$\sum_{i=0}^{n-1} b_i - \sum_{i=0}^{n-1} a_i, \tag{3.1}$$

2.
$$\frac{\sum_{i=0}^{n-1} b_i}{n} - \frac{\sum_{i=0}^{n-1} a_i}{n}, \tag{3.2}$$

3.
$$\sum_{i=0}^{m-1} \frac{b_i}{l_i} - \sum_{i=0}^{m-1} \frac{a_i}{l_i}, \tag{3.3}$$

4.
$$\frac{\sum_{i=0}^{m-1} \frac{b_i}{l_i}}{m} - \frac{\sum_{i=0}^{m-1} \frac{a_i}{l_i}}{m}, \tag{3.4}$$

where:

- $n (n \in \mathbb{N}_0)$ - the number of order book's snapshots
- $b_i$ - bid volume on index $i$
- $a_i$ - ask volume on index $i$
- $m (m \in \mathbb{N}_0)$ - number of bid/ask records
- $l_i$ - level on index $i$

Volume to level ratio is chosen to prioritize the information on the top of the limit order book.

On the table of executed transaction were applied only two aggregations:

1.
$$\sum_{i=0}^{n-1} b_i - \sum_{i=0}^{m-1} s_i, \tag{3.5}$$

2.
$$\frac{\sum_{i=0}^{n-1} b_i}{n} - \frac{\sum_{i=0}^{m-1} s_i}{m}, \tag{3.6}$$

where:

- $n (n \in \mathbb{N}_0)$ - number of all buy transactions
- $m (m \in \mathbb{N}_0)$ - number of all sell transactions
- $b_i$ - buy volume at index $i$
- $s_i$ - sell volume at index $i$

Finally, two ways were chosen to capture the change of price on a corresponding time interval. First one is the most common one, which is also used in each cryptocurrency exchange platform. The equation is this:

$$\lambda p = \frac{p_l - p_f}{p_f} \times 100, \tag{3.7}$$

where:

- $\lambda p$ is the price increase
- $p_l$ is the last price on the time period, the newest one,
- $p_f$ is the first price on the time period, the oldest one

Based on the equation, price increase $\lambda p$ is positive, if price goes up, or negative otherwise. The second way to capture the price change is to find the difference between the maximum value of price and the starting price on the time interval. The equation for this is:

$$\lambda p = \frac{\max_{0 \leq i < n} p_i - p_f}{p_f} \times 100, \tag{3.8}$$

where:

- $\lambda p$ is the price change
- $i (i \in \mathbb{N}_0)$ is the index of price in time period
- $n (n \in \mathbb{N}_0)$ is the number of price values in time period
- $p_i$ is the price with index $i$

- $p_f$ is the first price in the time interval, the oldest one

To measure correlation between price change and volumes of limit order books/transaction Spearman method was used. The choice was made between Spearman and Pearson in favor of Spearman, because Pearson captures the linear relationship between two continuous variables, whereas Spearman evaluates the monotonic relationship. The range of the Spearman's function lays in the interval $[-1, 1]$, where 1 is perfect positive correlation, -1 is perfect negative correlation and 0 is no correlation at all. As a result, the best spearmans returns for each pair of cryptocurrencies were not higher, than 0.4, which is pretty low to make good predictions. That is why a custom measure of correlation was in introduced, which was finding correlation, based only on the sign of the variables. The equation for this:

$$c = \frac{\sum_{i=0}^{n-1} f(x_i, y_i)}{n}, \tag{3.9}$$

where:

- $i (i \in \mathbb{N}_0)$ is the index in time interval
- $n (n \in \mathbb{N}_0)$ is number of variables in time interval
- $x_i$ and $y_i$ are observed variables
- $f$ is a function, that returns 1, if the signs of observed variables are the same, and 0 otherwise

In comparison to the custom method, the Spearman formula is as follows:

$$r = 1 - \frac{6 \sum_{i=0}^{n-1} d_i^2}{n(n^2 - 1)}, \tag{3.10}$$

where:

- $n (n \in \mathbb{N}_0)$ - number of data points
- $d_i$ - difference in ranks of the element on index $i$

This measure of correlation gave far better result, compared to Spearman. The results for both methods are shown on Table 3.4.

| | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| Spearman | 0.407 | 0.270 | 0.063 |
| Custom correlation | 0.722 | 0.633 | 0.966 |

Table 3.4: Correlation results.

Based on the efficiency of the two submitted methods, aggregated values, that give the best results for custom method, are going to be used further. Results, for which the custom method gave the best score, were aggregated on prices and limit order books for each cryptocurrency pair. Aggregations, made in the process, are shown in the Table 3.5.

| | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| Time | 300 min | 60 min | 360 min |
| Price aggregation | $\frac{p_l - p_f}{p_f} \times 100$ | $\frac{p_l - p_f}{p_f} \times 100$ | $\frac{\max_{0 \leq i < n} p_i - p_f}{p_f} \times 100$ |
| Volumes aggregation | $\frac{\sum_{i=0}^{n-1} b_i}{n} - \frac{\sum_{i=0}^{n-1} a_i}{n}$ | $\sum_{i=0}^{n-1} b_i - \sum_{i=0}^{n-1} a_i$ | $\frac{\sum_{i=0}^{m-1} \frac{b_i}{l_i}}{m} - \frac{\sum_{i=0}^{m-1} \frac{a_i}{l_i}}{m}$ |

Table 3.5: Aggregations.

## 3.3 LSTM and GRU

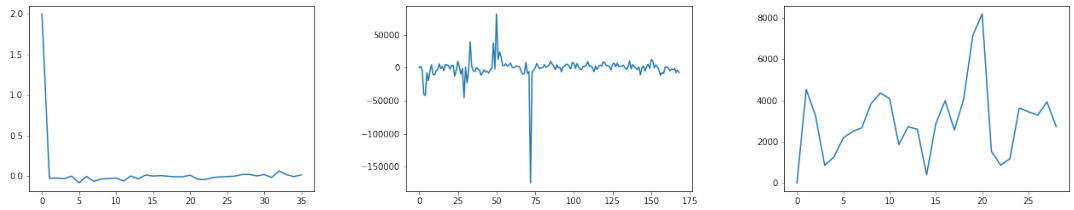
Primal goal of LSTM and GRU in this work is to predict future differences between volumes. Data generated on in section Analyzing and aggregating data are shown on the figure 3.1.

Figures 3.1a and 3.1b seem to have outliers, that increase variability in the data and decrease its statistical power, which may largely affect the learning process. It is clear, that for Bitcoin it is the first data point, that is why it is eliminated. However, for Ethereum it is a bit more complicated. Having

$$\mu = \frac{\sum_{i=0}^{n-1} x_i}{n}, \tag{3.11}$$

as mean of the whole dataset, and

$$\sigma = \frac{\sum_{i=0}^{n-1} x_i - \mu}{n}, \tag{3.12}$$



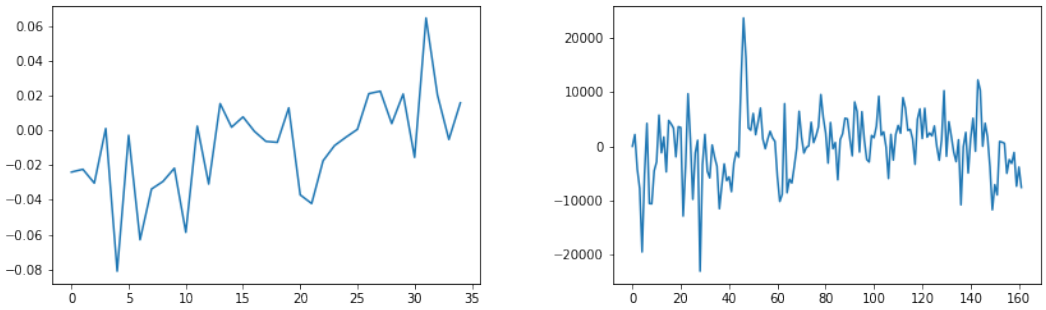

(a) BTC/USDT (b) ETH/USDT (c) DOGE/USDT

Figure 3.1: Aggregated volumes.

as standard deviation of the dataset, all values, which lays from mean further, that double standard deviation, are to get rid of. This means, that values, that goes to a new dataset, should meet the condition:

$$x_i - \mu < 2 * \sigma. \tag{3.13}$$

After applying these changes, BTC/USDT and EHT/USDT datasets look like this:




(a) BTC/USDT (b) ETH/USDT

Figure 3.2: Aggregated volumes with reduced outliers.

Because all the work is done on time series data and after aggregating the data, first index of aggregated prices refers to a time period, which is the same for the first index of aggregated volumes and so on, all indexes, that were removed from the volumes dataset by removing outliers are also removed from the prices dataset.
To get better results in terms of prediction, data on which the models are trained are advised to be stationary. Making data stationary means removing the trend, that can affect the training process. To check the data stationarity Augmented Dickey–Fuller test is applied, the null hypothesis for which stands, that the series has a unit root. The results of the test are shown in the Table 3.6. It is clear, that the null hypothesis can be rejected for both ETH/USDT and DOGE/USDT pairs, as their p-values are less, than 0.05 and test statistic is less, than any of the critical values. This can't be said about BTC/USDT pair, which means, that the dataset is not stationary.

Differencing method was applied to get rid of the trend. In result each member of the differed dataset looks like the difference between two neighbouring members of the original dataset. Augmented Dickey–Fuller test applied on the new dataset returns p-value equal to $9 \times 10^{-25}$ and test statistic less, than any of the critical values, which means, that the dataset is in fact stationary.
Before feeding data to the model, all datasets are normalized in the range $[-1, 1]$. For output $y_t$ there is an input $x_{t-1}, ..., x_{t-n}$, where $n (n \in \mathbb{N})$ is the number of steps back in time. Because of the cryptocurrencies' differences, meaning there applicative purposes and their background, there is going to be a separate model for each pair.

| | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| Test statistic | -1.99 | -3.93 | -4.03 |
| P-value | 0.29 | 0.002 | 0.001 |
| Critical value (1%) | -3.65 | -3.47 | -3.7 |
| Critical value (5%) | -2.95 | -2.88 | -2.98 |
| Critical value (10%) | -2.62 | -2.58 | -2.63 |

Table 3.6: Augmented Dickey–Fuller test's results.

Firstly, to choose the number of steps back in time, several models are constructed for each cryptocurrency pair. Three values, are observed: 2, 3, 4. Experiments made are made solely on LSTM-RNN models. The measurement is the loss on test data. Mean squared error is used to measure the loss, the equation for which is:

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(Y_i - \tilde{Y_i})^2, \tag{3.14}$$

where:

- $n$ - number of data points
- $Y$ - observed values
- $\tilde{Y}$ - predicted values

After finding the best architecture for the LSTM-RNN model, meaning number of layers and number of neurons in each layer, the results are shown in the Table 3.7. It is clear, that for BTC/USDT and DOGE/USDT 4 steps back work the best, whereas for ETH/USDT it is the number 2. But based on validation performance for DOGE/USDT, better number of steps back is 2. The work done further is going to use those parameters.

To optimize models performance Dropout is used as one of the most used regularization techniques. Using Dropout method during each iteration different set of nodes is used. It is used to reduce overfitting and eliminates situations, when the model is very dependent on some neurons.

| Number of steps back | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| 2 | 0.65 | 0.08 | 0.03 |
| 3 | 0.52 | 0.11 | 0.17 |
| 4 | 0.26 | 0.21 | 0.01 |

Table 3.7: Loss on test data using different number of steps back.

For optimization purposes Adam optimizer is chosen. Decay is used to finding a local minima, it gradually reduces learning rate. The default formula for this is:

$$lr_i = lr_0 \times \frac{1.0}{1.0 + decay \times i}, \tag{3.15}$$

where:

- $i (i \in \mathbb{N}_0)$ - number of the current iteration, considering, that the batch size is 1, there is $n \times e$ iteration, where $n (n \in \mathbb{N}_0)$ is the number of samples in training dataset and $e (e \in \mathbb{N}_0)$ is the number of epochs
- $lr_i$ - learning rate on iteration $i$
- $lr_0$ - initial learning rate
- $decay$ - decay value

After applying all the necessary changes to the LSTM-RNN models, results for each pair are shown in the Table 3.8.

For the purpose of comparison of final results between LSTM-RNNs and GRU-RNNs, GRU-RNNs are going to use the same train, validation and test data, meaning that the same number of steps back in time is used.

| | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| Final loss | 0.010 | 0.041 | 0.006 |
| Final validation loss | 0.168 | 0.038 | 0.416 |
| Loss on test data | 0.176 | 0.06 | 0.022 |

Table 3.8: LSTM results for each pair.

| | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| Final loss | 0.007 | 0.049 | 0.024 |
| Final validation loss | 0.219 | 0.039 | 0.530 |
| Loss on test data | 0.463 | 0.061 | 0.061 |

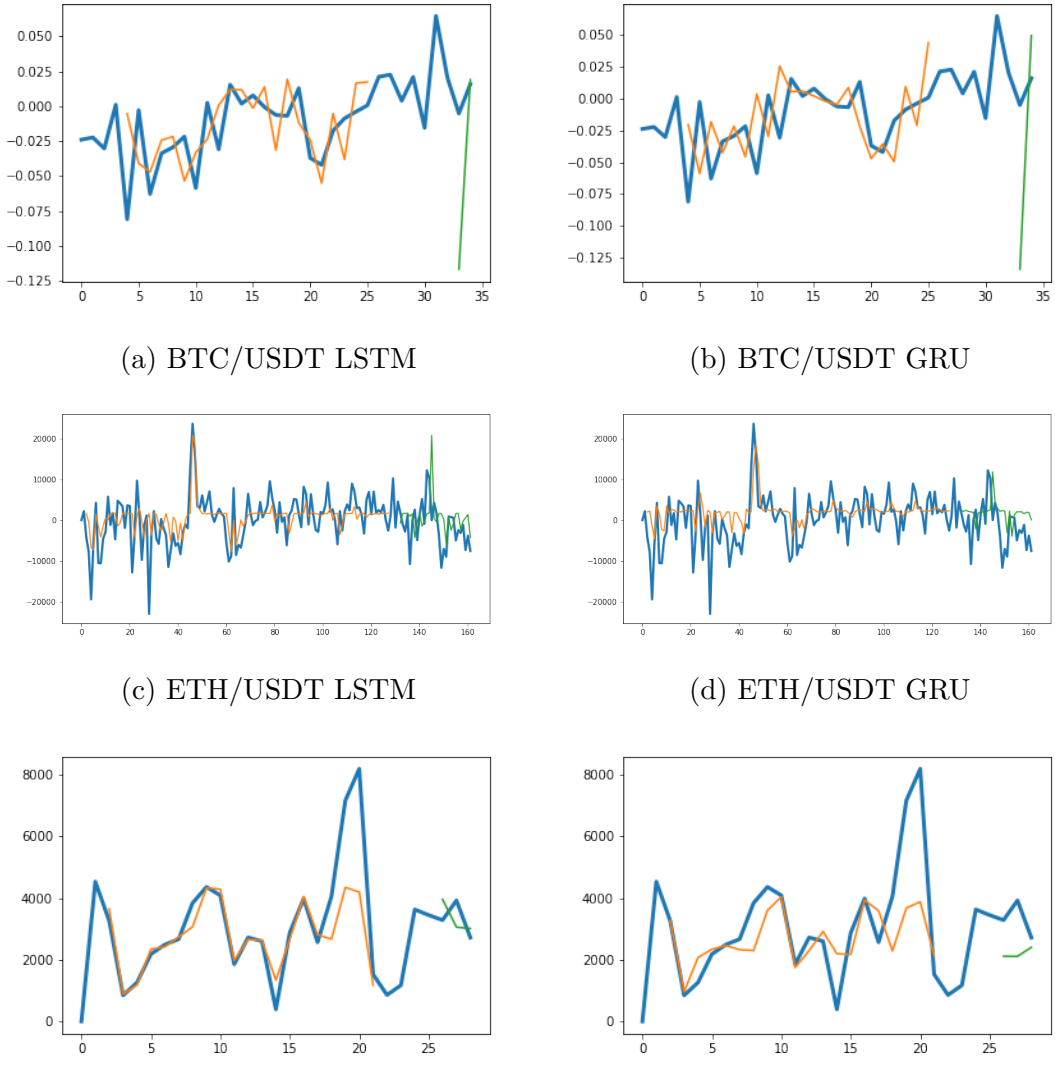
Table 3.9: GRU results for each pair.

| | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| LSTM-RNN | 4-8-4-1 | 4-8-4-1 | 16-8-4-2-1 |
| GRU-RNN | 8-4-2-1 | 4-2-1 | 4-8-4-1 |

Table 3.10: Networks architecture.

After settling down with architecture and applying all the optimization, mentioned earlier, the results received are shown in the Table 3.9.

For the purpose of avoiding overtraining and to capture the best models checkpoints were used. In the Table 3.10 final networks architectures are introduced. Each number represents number of neurons in the layer. Last layer is the Dense layer with linear activation function, because this is a regression problem and the purpose of LSTM-RNNs and GRU-RNNs is to predict future volumes. Between each LSTM/GRU layer is inserted a Dropout layer. The default architecture is used for LSTM/GRU cells, tanh is the activation function and sigmoid is used as a recurrent activation function.
Figure 3.3 compares the real values of each pair and predicted by LSTM and GRU models. Orange color represents predicted values on training data and green color represents predicted values on test data. BTC/USDT figures show that some values are off by a high margin, but the direction is captured properly. ETH/USDT seems to follow the real data, but predictions seems to be more gravitated to the zero value. DOGE/USDT subfigures display pretty close movement for both cases in terms of predicted values based on training data, whereas LSTM seems to be better in term of test predictions.
Considering that predicted values by LSTM-RNN seems to follow the real ones better than GRU-RNN and also having less error in the case of LSTM-RNN, LSTM-RNN shows better performance and seems to dominate GRU-RNN.



(a) BTC/USDT LSTM

(b) BTC/USDT GRU

(c) ETH/USDT LSTM

(d) ETH/USDT GRU

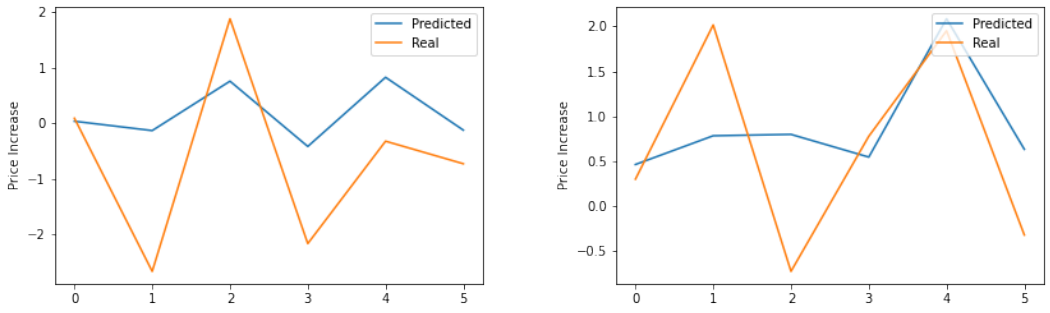
(e) DOGE/USDT LSTM

(f) DOGE/USDT GRU

Figure 3.3: Figures with real values and predicted values.

## 3.4 Price movement prediction

The purpose behind this section is to find the best model for price movement prediction. As a regression problem, deep neural networks(DNNs) were used. BTC/USDT volumes dataset and BTC/USDT price movement dataset were taken as input and output respectively. Two cases were explored: a neural network constructed for the input as it is and neural network constructed for the input, where each member of the input was in form $[x_{t-1}, x_t]$, meaning current value of volumes dataset and one before. Input and output were normalized in the interval of $[-1, 1]$. In the process of optimizing the model several regularization techniques were used, one of each was Dropout mentioned earlier in the work in the LSTM and GRU section. Batch normalization was also applied. The purpose behind this method is the same as the purpose of normalizing the input before training the model, it scales outputs of layers to have mean 0 and variance 1. The output of the layer is subtracted by the batch mean and then divided by the batch standard deviation before going to another layer. In this particular case, batch normalization did not give any better results, compared to networks without it. ReLU function was used as a default activation function for initial models. Final models had this architecture 1024-512-512-256-256-128-1, where numbers represent neurons in Dense layers. Last layer had a linear activation function, because of the regression nature of the problem. In both models LeakyReLU was used as an activation function in the first layers. Adam was chosen as an optimizer and decay was used during the learning process. Comparison between real and predicted values are shown in the figure 3.4. As can be seen the model with a single input performs better, it seem more flattened, than the actual values, but it captures the actual movement. Model with double input did not perform so good, and this can be explained in dependent features in the input.




(a) Single input. (b) Double input.

Figure 3.4: Results of DNN on BTC/USDT data.

It is known, that it is hard to make a solid regression model most of the time, and those experiments, mentioned before, proves it exactly right. But this kind of a problem can be interpreted as a classification problem. Price dataset can be interpreted in this way: **sell**, if the value is strongly positive, **buy**, if the value is strongly negative, and **hold**, if the value lays in a specified percent range. In the course of experiments, 1 percent is going to be used as the percent to **hold**. For weight initialization "he" initialization was used, so that the model would not get stuck in the learning process. Because it is a multi-class classification problem, softmax function is an activation function for the last layer. The equation of softmax function is:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=0}^{k-1} e^{z_j}}, \tag{3.16}$$

where:

- $\vec{z}$ -input vector
- $k(k \in \mathbb{N}_0)$ - number of classes in the multi-class classifier
- $e^{z_i}$ - standard exponential function

Softmax function returns probabilities for each class. Denominator in a function ensures, that all the output values of the function will sum to 1.
As a loss function categorical cross-entropy was used, which looks like this:

$$loss = -\sum_{i=0}^{n-1} y_i \times \log \tilde{y}_i, \tag{3.17}$$

where:

- $\tilde{y}$ - model output
- $y$ - target value
- $n(n \in \mathbb{N}_0)$ - output size

Final models were received in the course of learning with decay, where the model with the lowest validation loss was saved. Models performance on test data can be seen in the Table 3.11. The results show, that these models indeed can perform prediction enough for them to be useful, because any accuracy higher than 0.5 should help at least not lose money in the market. Model for DOGE/USDT pair got an astonishing 100 percent accuracy, but this can be explained by the low number of values in the test dataset.

| | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| Loss | 0.43 | 0.85 | 0.03 |
| Accuracy | 0.83 | 0.76 | 1.0 |

Table 3.11: Results on test data for multi-classification.

| | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| Accuracy | 0 | 0.47 | 1 |

Table 3.12: Accuracy score for each pair in a chain of predictions.

### 3.4.1 Prediction based on the volumes given by LSTM-RNNs

Change in volumes, working as an instrument for price prediction is predicted by LSTM-RNN models for each cryptocurrency pair. This should be already enough for an experienced trader, but this work also tries to predict the price change. This is achieved by making a chain of prediction. Predicted values from LSTM-RNN models are fed to the deep neural networks in pursue of predicting whether to buy, sell or hold. Results of this chain reaction are given in the Table 3.12. The accuracy scores given for BTC/USDT and DOGE/USDT pairs can be explained by the low number of samples in the test dataset. The ETH/USDT pair on the other hand shows the true nature of the final chain model.

Chapter 4

# Main Results

This section is considered to go over each particular step in the executed work and sum up the results received in the course of experiments with all the necessary explanations. Three main points of this work are:

1. Volumes change prediction using LSTM-RNN and GRU-RNN and finding the best model
2. Price movement prediction, based on volumes change
3. Price movement prediction, based on volumes change predicted by the best model, received in the step 1

, where the first one is the main purpose of this work.

## 4.1 Volumes prediction

In the chapter Overview of Overall Approach two recurrent neural networks were observed: LSTM-RNN and GRU-RNN. After finding the best time step back in time and tweaking the parameters of the network final models were received. Comparison between the two final networks for each cryptocurrency pair can be seen in the table 4.1, considering only the perfomance on test data.

| | BTC/USDT | ETH/USDT | DOGE/USDT |
|---|---|---|---|
| LSTM-RNN | 0.176 | 0.06 | 0.022 |
| GRU-RNN | 0.463 | 0.061 | 0.061 |

Table 4.1: Loss on test data in final models.

It is clear, that LSTM-RNN outperforms GRU-RNN, that is why LSTM-RNN is chosen as the best model for each cryptocurrency pair for volumes change prediction. Going through all the figures in figure 3.3 showing the real value and predicted one using LSTM-RNN these conclusions can be made only considering the performance on test data:

- For BTC/USDT pair the error is pretty big, but the movement is depicted perfectly
- In the case of ETH/USDT predicted values are a bit more flattened to the zero value, compared to the real ones, but all in all the pattern is followed. It also showed a really low MSE on test data, shown in the table 3.8
- LSTM-RNN showed the lowest MSE for DOGE/USDT, and the figure 3.3 also shows how close the values are, but unfortunately the pattern is not depicted properly for test data

## 4.2 Price prediction

Different kinds of models were used before coming to the final ones. Firstly, the problem was looked at as a regression type of problem, trying to predict the price change in percents. Two approaches were considered: model with an input of volumes change as it is and model with an input, where each member was in a form $[v_prev, v]$, meaning current volume change and the previous one. Turned out, that the model with an input of a single feature worked better. The double input was considered as an input with more information, but as it turned out, because of the same nature of the features this did not work as good as it was hoped to. Looking at the figure 3.4a it is seen, that the model depicts the movement perfectly, but the values are off by quiet a big margin. That is why it was decided to convert the task from regression problem to multi-class classification problem with 3 classes: buy, sell and hold. The optimal models were found for each cryptocurrency pair, the results of which can be seen in the table 3.11. Based on these results a conclusion can be made, that these models are good enough to make predictions, having the real volumes changes, and can be used to operate on the market with profit.

## 4.3 Price prediction, based on predicted volumes changes

Combining the results from step Volumes prediction and step Price prediction, a true complexity of the work's problem can be seen. Because of the short test dataset used in the case of BTC/USDT and DOGE/USDT cryptocurrencies, they show an abnormal minimum and abnormal maximum. DOGE/USDT model tends to outperform all other because of the output it is considered to create, which is either hold or sell. Also the volumes dataset is not really suitable for a research, as it depicts only positive momentum. The true nature of the final model is assumed to be shown in the ETH/USDT case. The result received is pretty close to 0.5, but still is not enough to use it in the market, cause it will only cause money losses.

# Chapter 5

# Conclusion

## 5.1 Summary

In the course of the executed work with mentioned data and tools these conclusions can be made:

1. LSTM-RNN tends to outperform GRU-RNN in predicting future volume changes
2. LSTM-RNN can be used to predict future volume changes, that can be useful for the trader in the market
3. Knowing the future volume change, DNN can be used to predict, what to do in the market with the accuracy bigger, than 50 percent, meaning, that this model won't lose money to its user
4. A chain of LSTM-DNN models constructed during the course of this work are not advised to be used in the market, because of its poor performance

The $4^{\text{th}}$ point can be justified by several points. Let's go over the aspects, that could most possibly affect this kind of behavior:

- To get the data to work with, the decision was made in the favor of collecting data, because there is no free distributed data of this kind. The process of data collecting was limited by the used machinery, the data were collected every 30 seconds, when the best possible data to be obtained are tick-by-tick data, which demands an enormous amount of power. The data were also collected only on the Binance Exchange. All together, this is only a small part of all the ordered and transferred data concerning cryptocurrency, when data is the main key in making a decent model. The quality of information can be seen in the step of finding correlation between the price change and volumes change, where principal method did not work and the need to resort to a custom method showed up.

- As the time interval, on which to conduct the work, was chosen to be 7 days, this led to some unexpected complications. In the course of aggregating data and removing outliers to make the dataset stationary small datasets were obtained for BTC/USDT and DOGE/USDT pairs, which could led to the model, that is in fact unsuitable for the market use. This also culminated in the small test data for both of the pairs.

- Generated dataset of volume changes turnes out to unsuitable, because it depicts only positive movement

- The poor performance of the final model can be also justified by a very novel nature of cryptocurrency itself. It is very volatile, which can lead to a very unpredictable behaviour. A very strong impact is sometimes played by the social part of the cryptocurrency, which is not usually seen in the limit order books or transactions at the first sight and can lead to a pitiable finale.

## 5.2 Contributions of the Thesis

This work perfectly depicts, that LSTM-RNN outperforms GRU-RNN in the problem of volume changes prediction and an assumption can be made, that it is generally more suitable for time series predictions in the cryptocurrency market. Results say, that there is a huge possibility to predict, what to do on the market, using DNN model. It also shows, how much of an importance plays the data in a pursue of getting the good results. There should be payed a lot of attention to the data collection in attempt of getting as much information as possible. This is also a great example of a really early stage of cryptocurrency and it maybe should be researched more to its core to understand, what can affect its behaviour or some time should pass for it to stabilize, because it is not yet fully accepted by the global society, having only 3.9% of estimated global ownership in the year 2021.

## 5.3 Future Work

The author of the thesis suggests these things for the future work:

- The main thing is to properly approach the issue of data collection. Firstly, a great advantage would be to collect the data from all possible sources. An enormous gain would be a collection of tick-by-tick data. By combining tick-by-tick data of limit order books and executed transactions would give a complete picture of the state of the market. It would be possible to extract the information of the orders, that were canceled, by comparing the state of the limit order books and executed transactions at every tick, if the order was indeed executed or in fact canceled. That is a completely new behaviour, which is not researched in this work.

- Different time intervals could be researched. Having tick-by-tick data it may be possible to use time periods smaller than 7 days for training an LSTM-RNN model and then updating it every day. Bigger time intervals can also be observed, but in the case of cryptocurrency small intervals are a better option, because there should be far less volatility reflected in them.
- For limit order books can be made another apply, which is an exact opposite of one of the used one in this work. Instead of prioritizing the top level of limit order book, bottom ones can be prioritized. It can open a great spectre of possibilities, because some big players place their order in the bottom of the limit order books intentionally, so that they are not seen for most of the people.
- It is strongly advised to make the price aggregation, given by the equation 3.7, which perfectly depicts the price movement and at least the problem of price prediction can be converted from regression to multi-classification class, which would work perfectly for an average user.
- Prediction on the data from the market can be combined with the social side of cryptocurrencies. The most common approach is to use the Twitter sentiment and it is proven to give good results by using LSTM network in the work [14].

Chapter 6

# Abbreviations

| | |
|---|---|
| $\mathbb{N}$ | Natural numbers set |
| $\mathbb{N}_0$ | Natural numbers set $\cup \{0\}$ |
| $\mathbb{Z}$ | Integer numbers set |
| $\mathbb{Z}_m$ | Least nonzero residue number set with a module of $m$ |
| $\mathbb{S}_m$ | Symmetric residue number set with a module of $m$ |
| $\mathbb{Q}$ | Rational numbers set |
| $\mathbb{F}_t$ | Floating point numbers set with a precision of $t$ |
| $\mathbb{R}$ | Real numbers set |

| | |
|---|---|
| **LSTM** | Long short-term memory |
| **GRU** | Gated recurrent unit |
| **DNN** | Deep neural network |
| **RNN** | Recurrent neural network |
| **LOB** | Limit Order Book |

Chapter 7

# Supplement Structure

- `src` ..... the directory of source codes
  - `thesis` ..... the directory of LaTeX source codes of the thesis
  - `thesis.pdf` ..... the thesis text in PDF format

# Bibliography


[1] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, pp. 367–415, 174, 420, http://www.deeplearningbook.org.

[2] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm", *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[4] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.", *Journal of machine learning research*, vol. 12, no. 7, 2011.

[5] 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*, 2012.

[6] Y.-W. Cheung and K. S. Lai, "Lag order and critical values of the augmented dickey–fuller test", *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 277–280, 1995.

[7] C.-H. Wu, C.-C. Lu, Y.-F. Ma, and R.-S. Lu, "A new forecasting framework for bitcoin price with lstm", in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2018, pp. 168–175. DOI: 10.1109/ICDMW.2018.00032.

[8] C. Cao, O. Hansch, and X. Wang, "The information content of an open limit-order book", *Journal of Futures Markets*, vol. 29, no. 1, pp. 16–41, DOI: https://doi.org/10.1002/fut.20334. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fut.20334. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/fut.20334.

[9] Z. Zhang, S. Zohren, and S. Roberts, "Deeplob: Deep convolutional neural networks for limit order books", *IEEE Transactions on Signal Processing*, vol. 67, no. 11, pp. 3001–3012, 2019. DOI: 10.1109/TSP.2019.2907260.

[10] P. Cowpertwait and A. Metcalfe, "Introductory time series with r", in Jan. 2009, p. 122, ISBN: 0387886974, 9780387886978. DOI: 10.1007/978-0-387-88698-5.

[11] R. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*, 2nd ed. OTexts, 2018, p. 215.

[12] C. Bishop, *Neural networks for pattern recognition*. Oxford University Press, USA, 1995, p. 296.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[14] X. Huang, W. Zhang, X. Tang, *et al.*, "Lstm based sentiment analysis for cryptocurrency prediction", in *Database Systems for Advanced Applications*, C. S. Jensen, E.-P. Lim, D.-N. Yang, *et al.*, Eds., Cham: Springer International Publishing, 2021, pp. 617–621, ISBN: 978-3-030-73200-4.

[15] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Using deep learning for price prediction by exploiting stationary limit order book features", *Applied Soft Computing*, vol. 93, p. 106 401, 2020, ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2020.106401. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494620303410.

[16] A. Aggarwal, I. Gupta, N. Garg, and A. Goel, "Deep learning approach to determine the impact of socio economic factors on bitcoin price prediction", in *2019 Twelfth International Conference on Contemporary Computing (IC3)*, 2019, pp. 1–5. DOI: 10.1109/IC3.2019.8844928.