Bitcoin Price Prediction and Automated Trading via LSTM Networks and Reinforcement Learning

Ioannis Ntourmas[1]
Dionisios N. Sotiropoulos, PhD[1]

July 18, 2022

[1]University of Piraeus, Department of Informatics

Table of contents

Introduction

Bitcoin

Bitcoin (BTC) constitutes a decentralized digital currency which is mediated by the **peer-to-peer** bitcoin network.

Bitcoin **transactions** are verified within the aforementioned network by means of cryptography and are recorded in the publicly distributed **ledger** of **blockchain**.

BTC was the first **cryptocurrency** invented, back in **2008**, by an unknown person or group of individuals using the alias **Satoshi Nakamoto**.

The worldwide utilization of the BTC cryptocurrency did not begin before **2009** when its implementation as an **open-source** software platform was first released.

The price of Bitcoin

The price of Bitcoin has been proven to be **extremely volatile**.

Professional traders claim that investing in this cryptocurrency is quite **similar** to **gambling**.

It is worth mentioning, however, the existence of exceptional cases were individual investors were able to earn **millions of dollars** by trading in Bitcoin.

It is extremely impressive to notice that bitcoin was worth **1 dollar** back in **2011** reaching a value of **60,000 dollars** by November **2021**.

Problem Definition

Problem Definition

Predicting the future price of BTC is an extremely challenging task since it is influenced by a wide range of parameters such as:

1. **political issues**
2. **current affairs**
3. **natural disasters**
4. **fake news**
5. a **singe tweet**

In this work, we address the following problems:

- predict the **average price** of BTC **per minute** through the utilization of **Long Short-Term Memory** (**LSTM**) networks.
- develop an **automated trading mechanism** for BTC based on training a **Recurrent Reinforcement Learning (RRL)** model.
- incorporate the **LSTM-based** predictions in the **RRL-based** trading mechanism in order to enhance its profitability.

Dataset Description

Dataset Description I

Our dataset was collected from the **CryptoDataDownload** page forming a total amount of **14400 records**.

This web source provides information concerning the **price** and **volume** of bitcoins trades on a **minute basis** from **021-10-20 03:01 a.m.** to **2021-10-20 03:00 a.m**

Data Description II

Figure 1 presents the **general structure** of our dataset.

In this work we utilize the closing value of bitcoin at the end of each minute interval as the target prediction variable.

	time	low	high	open	close	volume
0	2021-10-30 03:00:00	61868.81	61920.00	61888.18	61919.98	2.946079
1	2021-10-30 02:59:00	61882.74	61912.67	61903.16	61882.74	2.942357
2	2021-10-30 02:58:00	61854.82	61907.96	61854.82	61903.15	3.047848
3	2021-10-30 02:57:00	61848.04	61883.02	61883.01	61851.39	3.381070
4	2021-10-30 02:56:00	61879.33	61925.42	61907.47	61882.28	5.663128

Figure 1: Sample data.

Data Normalization:

- All price values were normalized in the $[0, 1]$ interval.

Data Preparation

Training - Testing Data Segmentation:

- The first **90%** of the available price observations were used for training.
- The last **10%** of the available price observations were used for testing (i.e. **1440 minutes** or **24 hours**).

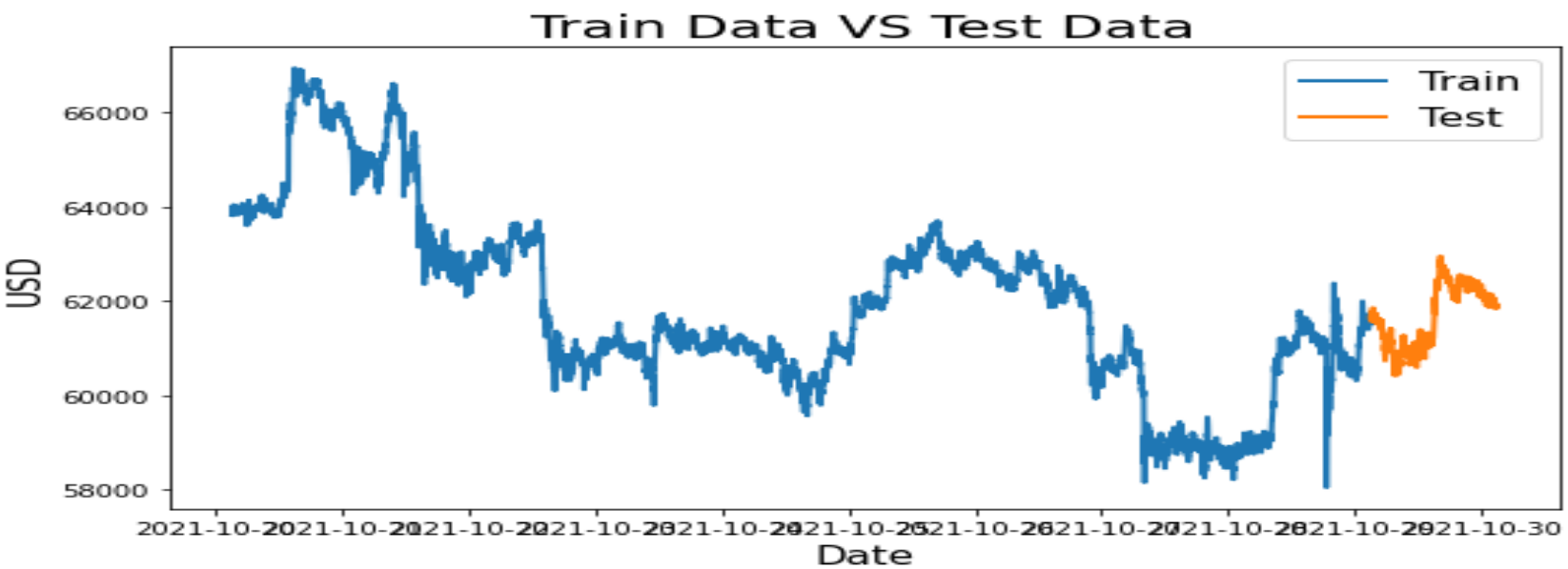

Figure 2: Train data VS Test data.

Long Short-Term Memory Networks

Recurrent Neural Networks I

Recurrent Neural Networks (**RNNs**) possess the ability to model the **inherent time structure** of an input signal which is presented to the system as **sequence** of **multi-dimensional vectors** according to:

$$\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\} \tag{1}$$

where $\boldsymbol{x}_t$ is the **input vector** at time t.

RNNs are able to capture the **time dependency** of a given input sequence to its **previous instances** through the incorporation of a time-dependent **state vector** $\boldsymbol{h}_t$ whose computation depends upon the current **input vector** $\boldsymbol{x}_t$ as:

$$\boldsymbol{h}_t = F(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta}) \tag{2}$$

where $\boldsymbol{\theta}$ encapsulates the shared **weight-related** parameters of the model.

Recurrent Neural Networks II

The unfolded version of the recursive Eq. 2 after t time steps can be formulated through the utilization of a new function $G^{(t)}(\cdot)$ as:

$$\boldsymbol{h}_t = G^{(t)}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \cdots, \boldsymbol{x}_2, \boldsymbol{x}_1) \tag{3}$$

which is depicted in Fig. 3.

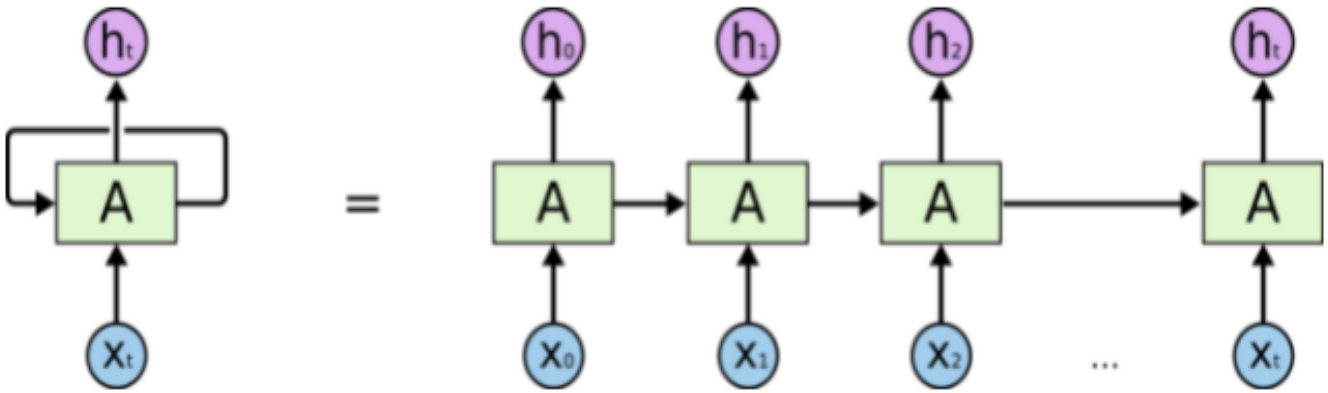

Figure 3: Recurrent neural network.

Long Short - Term Memory Networks I

Traditional **RNNs** suffer from the problem of **vanishing gradients** and therefore, fail to capture **long term dependencies** of the input signal with respect to its previous values.

LSTM networks surpass this predicament by introducing a new version of the fundamental recurrent unit, namely, the **gated LSTM cell** that is designed to control which information will eventually pass through.

Neural architectures containing repeating modules of **interacting layers** of **LSTM cells** exhibit the ability to **track information** throughout **many time steps**.

Long Short - Term Memory Networks II

Each LSTM component maintains a cell state $\boldsymbol{c}_t$ facilitating the flow of information within the unit.

The hidden state $\boldsymbol{h}_t$ of the LSTM cell is utilized to devise the final output $\boldsymbol{o}_t$ of the unit.

Information is **added** to or **removed** from the current cell state through structures such as the input gate ($\boldsymbol{i}_i$), the output gate ($\boldsymbol{o}_t$) or the forget gate ($\boldsymbol{f}_t$).

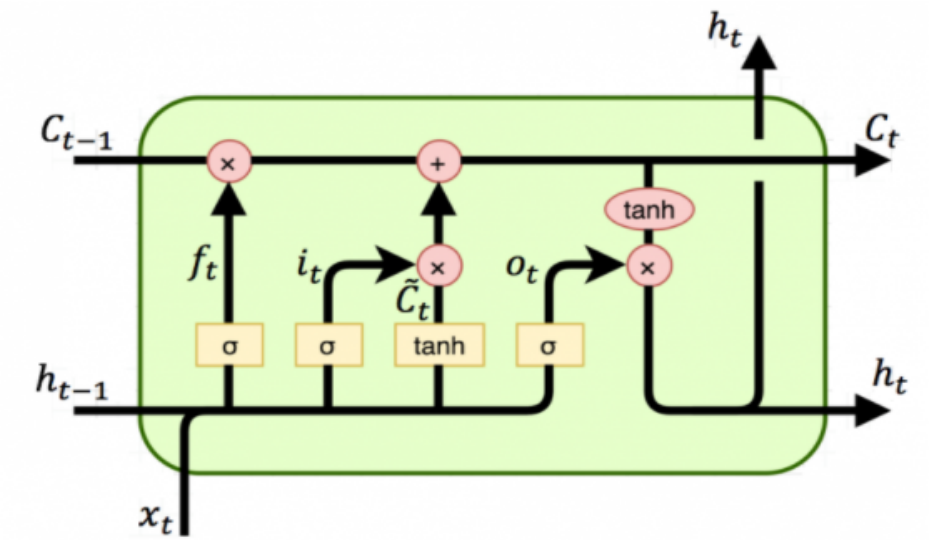

Figure 4: LSTM unit.

Long Short - Term Memory Networks III

- $\boldsymbol{b} = \{\boldsymbol{b}_i, \boldsymbol{b}_o, \boldsymbol{b}_f, \boldsymbol{b}_c\}$:
  is the set of bias vectors associated with the input gate, output gate, forget gate, and cell state of the LSTM unit.
- $\boldsymbol{W} = \{\boldsymbol{W}_i, \boldsymbol{W}_o, \boldsymbol{W}_f, \boldsymbol{W}_c\}$:
  is the set of input-related weight matrices associated with the input gate, output gate, forget gate, and cell state of the LSTM unit.
- $U = \{\boldsymbol{U}_i, \boldsymbol{U}_o, \boldsymbol{U}_f, \boldsymbol{U}_c\}$:
  is the set of hidden state-related recurrent weight matrices associated with the input gate, output gate, forget gate, and cell state of the LSTM unit.

The hidden state $\boldsymbol{h}_t$ of the LSTM unit is updated by the following equation:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \cdot \tanh(\boldsymbol{c}_t) \tag{4}$$

where $\boldsymbol{o}_t$ is the outcome of the output gate and $\boldsymbol{c}_t$ is the current cell state. Given that $\sigma(\cdot)$ is the sigmoid function, $\boldsymbol{o}_t$ is computed by:

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \cdot \boldsymbol{x}_t + \boldsymbol{U}_o \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{5}$$

Long Short - Term Memory Networks IV

The cell state is updated as follows:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \tag{6}$$

where $\tilde{\boldsymbol{c}}_t$ is the new information to be incorporated in the cell state of the LSTM unit, computed as:

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_c \cdot \boldsymbol{x}_t + \boldsymbol{U}_c \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{7}$$

The outcome of the forget and input gates is given by the following equations:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \cdot \boldsymbol{x}_t + \boldsymbol{U}_f \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{8}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \cdot \boldsymbol{x}_t + \boldsymbol{U}_i \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{9}$$

Recurrent Reinforcement Learning

Recurrent Reinforcement Learning I

RRL formulates the problem of determining the sequence of **optimal trading positions**:

$$F = \{F_1, \cdots, F_T\} \tag{10}$$

as the following maximization problem:

$$\max_{F} \quad S_T \tag{11}$$

The objective function S_T corresponds the **Sharpe's Ratio** which is a commonly used indicator that quantifies the **risk adjusted performance** of an investment strategy over a trading period T, computed by:

$$S_T = \frac{\frac{1}{T}\sum_{t=1}^{T} R_t}{\sqrt{\frac{1}{T}\sum_{t=1}^{T} R_t^2 - (\frac{1}{T}\sum_{t=1}^{T} R_t)^2}} \tag{12}$$

given that R_t represents the **net revenue** at time t according to:

$$R_t = \mu \cdot (F_{t-1} \cdot r_t - \delta \cdot |F_t - F_{t-1}|) \tag{13}$$

where $r_t = p_t - p_{t-1}$ is the price difference between two consecutive time instances and δ the associated **transaction fee**.

Recurrent Reinforcement Learning III

Each trading position F_t at time $t \in [T]$, is encoded as a real value in the $[0, 1]$ interval representing the percentage with respect to a maximum amount of bitcoins μ that should be traded at any given time.

Long Position ($F_t > 0$): the trader **buys** an amount $n_t = \mu \cdot F_t$ of bitcoins at price p_t and hopes that it appreciates by time $t+1$. Thus, the momentary revenue when **opening** the **position** is given by:

$$\delta R_t = -n_t \cdot p_t \tag{14}$$

By **selling** the same amount of bitcoins at time $t+1$, the momentary revenue when **closing** the **position** will be:

$$\delta R_{t+1} = n_t \cdot p_{t+1} \tag{15}$$

Therefore, the revenue at time $t+1$ will be given as:

$$R_{t+1} = \delta R_t + \delta R_{t+1} = \mu \cdot F_t \cdot r_{t+1} \tag{16}$$

It is obvious that $p_{t+1} > p_t \implies r_{t+1} > 0$ which, in turn yields $R_{t+1} > 0$.

Recurrent Reinforcement Learning III

Short Position ($F_t < 0$): the trader **borrows** and **sells** an amount $n_t = |\mu \cdot F_t|$ of bitcoins at price p_t with the expectation that the price will decrease at time $t+1$. Thus, the momentary revenue when **opening** the **position** is given by:

$$\delta R_t = n_t \cdot p_t \tag{17}$$

By **buying** the same amount of bitcoins at time $t+1$ in order to return the initial loan, the momentary revenue when **closing** the **position** will be:

$$\delta R_{t+1} = -n_t \cdot p_{t+1} \tag{18}$$

Therefore, the revenue at time $t+1$ will be given as:

$$R_{t+1} = \delta R_t + \delta R_{t+1} = -\mu \cdot F_t \cdot r_{t+1} \tag{19}$$

It is obvious that $p_{t+1} < p_t \Longrightarrow r_{t+1} < 0$ which, in turn yields $R_{t+1} > 0$.

Recurrent Reinforcement Learning IV

Each trading position is determined through the utilization of a single layer neural network whose output node provides the following response:

$$F_t = \tanh(\boldsymbol{w}^T \cdot \boldsymbol{x}_t) \tag{20}$$

where $\boldsymbol{x}$ and $\boldsymbol{w}$ are the **input** and **weight vectors** respectively.

The input vector is formed by aggregating information concerning the past price values of bitcoin within a time frame of size M and the trading position taken the exact previous moment as:

$$\boldsymbol{x}_t = [1, r_{t-M}, \cdots, r_t, F_{t-1}] \tag{21}$$

Thus, the optimal weight vector $\boldsymbol{w}$ can be acquired by employing the following iterative procedure:

$$\boldsymbol{w} \longleftarrow \boldsymbol{w} + \rho \cdot \frac{\partial S_T}{\partial \boldsymbol{w}} \tag{22}$$

Experimental Results

Long Short-Term Memory Model Results I

Training Parameters:

1. Loss Function: Mean Squared Error
2. Activation Function: tanh
3. Validation Percentage: 20% of the training data.
4. Early Stopping Criterion: 10 epochs to avoid overfitting.
5. Maximum Training Epochs: 100

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 32)                4352

 dropout (Dropout)           (None, 32)                0

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: Model Summary.

Long Short-Term Memory Model Results II

Figures below present the prediction accuracy of the LSTM-based model on the testing dataset with past time frame size $M = 10$.

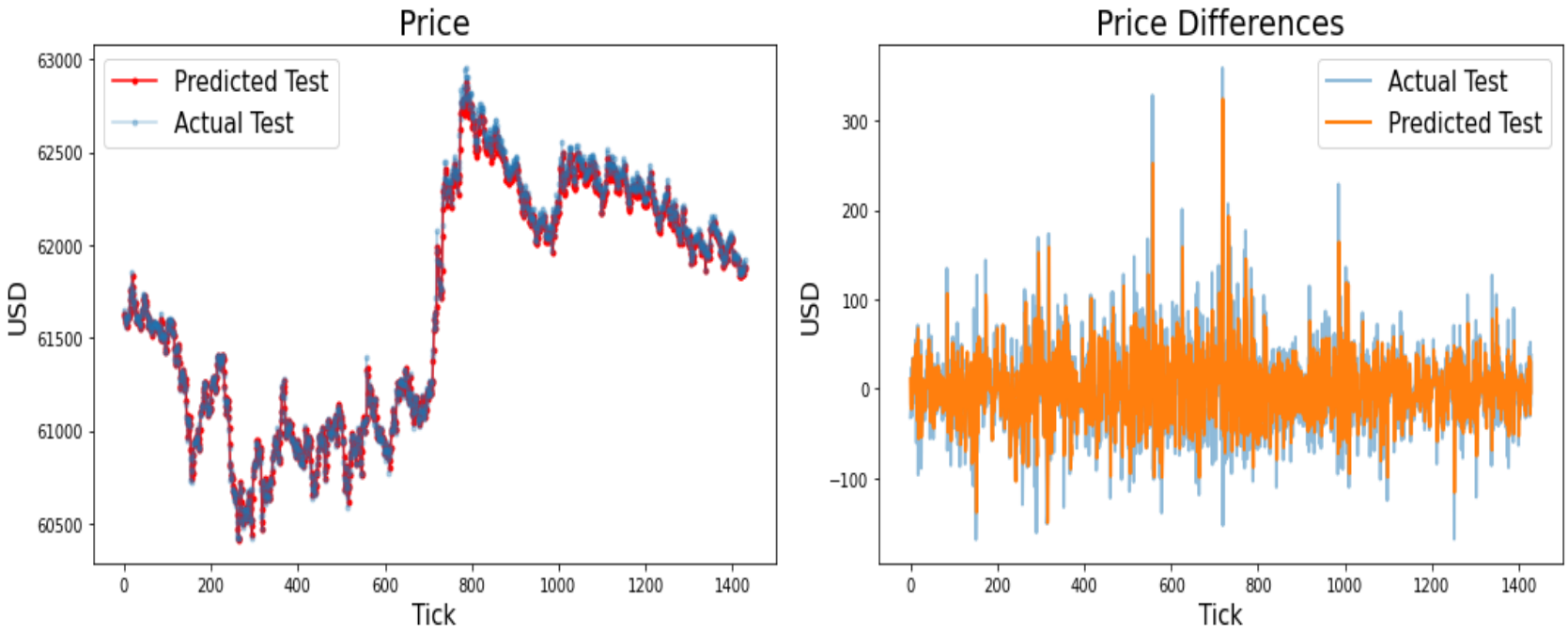

Figure 6: Test data VS Predicted data.

Long Short-Term Memory Model Results III

Table 1 summarizes the prediction accuracy of the LSTM-based model in terms of **MAE**, **MSE**, **RMSE** and $\boldsymbol{R}^2$ when varying the past time window size M.

Table 1: LSTM MODEL PERFORMANCE ON TEST DATA.

INPUT SIZE	MAE	MSE	RMSE	R^2
5	41.48	3126.09	55.91	0.99
10	44.66	3453.36	58.76	0.99
15	48.03	3932.06	62.70	0.99
20	45.13	3627.90	60.23	0.99
25	45.60	3646.29	60.38	0.99
30	43.61	3342.37	57.81	0.99
35	43.45	3328.33	57.69	0.99
40	44.91	3468.22	58.89	0.99
45	44.57	3456.22	58.78	0.99
50	46.13	3653.25	60.44	0.99

Recurrent Reinforcement Learning Model Results I

The Recurrent Reinforcement Learning (RRL) model was trained for **1000 epochs**. The learning rate was ρ was set equal to 0.5 and we considered that the transaction cost δ was equal to 0.

Figure 7 presents the profit gained by the **RRL-based** trading model (**blue curve**) for each time instance of the training set against the **buy and hold strategy** (**green curve**) when the time frame parameter was set to $M = 10$.

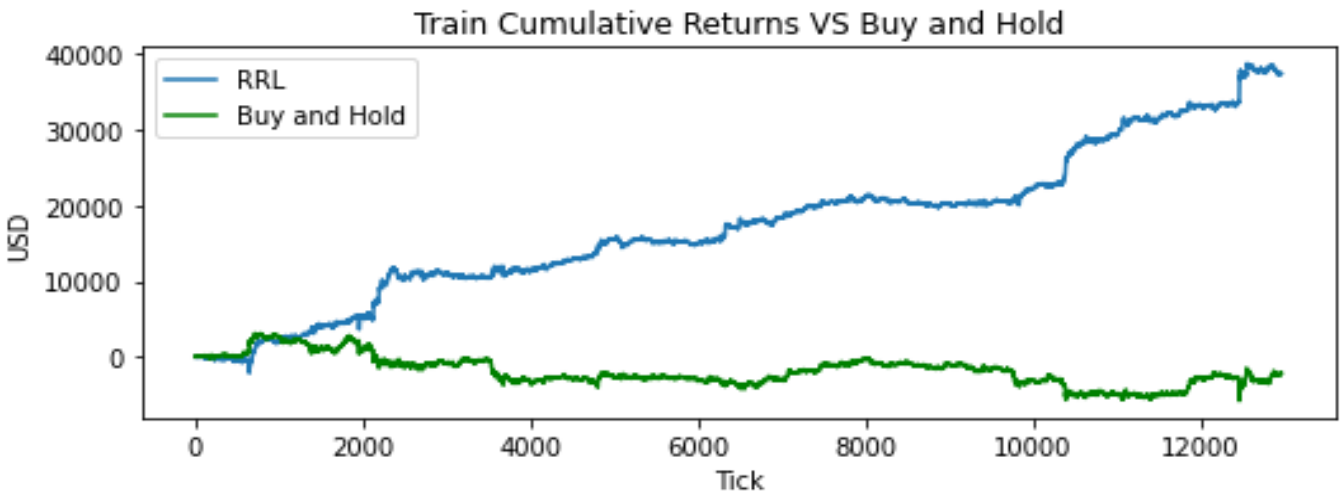

Figure 7: Profits on train data.

Recurrent Reinforcement Learning Model Results II

Figure 8 presents the profit gained by the two **RRL-based** trading models for each time instance of the testing set against the **buy and hold strategy** (**green curve**) when the time frame parameter was set to $M = 10$.

The **blue curve** identifies the **RRL-based** trading model on the standard feature set, whereas the **orange curve** depicts the trading performance of the **RRL-based** model on the extended set of features which incorporates the **LSTM-based** prediction for the **future price** of BTC.

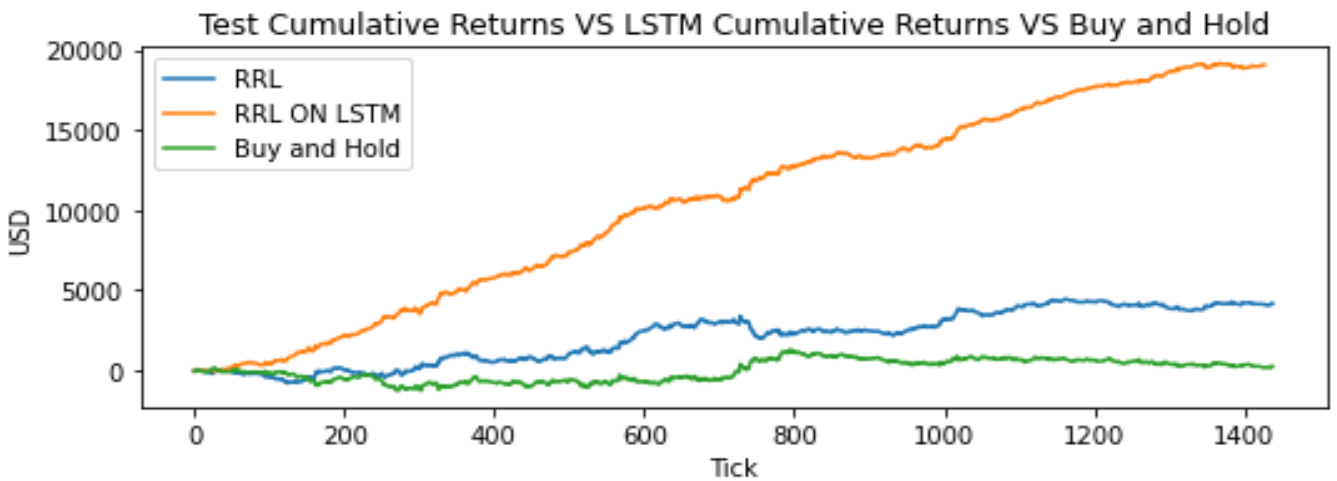

Figure 8: Profits on test data.

Recurrent Reinforcement Learning Model Results III

Table 2 presents a comparative evaluation of the two proposed trading models on the testing data set in terms of the Sharpe's Ration and the total amount of USD that were ultimately gained or lost for various values of the time frame parameter.

Table 2: SHARPE RATIO AND PROFITS ON TEST DATA (USD)

M	SHARPE RATIO	RRL PROFIT	RRL-LSTM PROFIT
5	0.034	-1321.06	13984.91
10	0.049	4184.85	19073.93
15	0.050	4319.34	7795.68
20	0.031	-1471.86	-7753.83
25	0.041	656.54	18160.14
30	0.047	1738.16	8397.31
35	0.048	2301.87	9158.30
40	0.052	-2480.82	-5651.78
45	0.054	2403.26	259.49
50	0.054	-1423	6422.99

It is evident that passing the LSTM-based predictions in the RRL-based trading model is more profitable for the majority of the experiments.

Conclusions & Future Research

Conclusions & Future Research

- The experimental results presented in this paper provide strong evidence concerning the trading superiority of the LSTM-enhanced RRL-based trading algorithm.
- LSTM networks were proved extremely efficient in predicting the future price of BTC both in training and testing environments.
- Future research will focus on experimenting with larger volumes of high frequency data where the price of BTC will change on the time scale of several milliseconds.
- Ongoing research is conducting on incorporating information from the Limit Order Book which is provided by several digital cryptocurrency trading platforms.
- An interesting research avenue may be explored towards incorporating Sharpe's Ratio as the objective function to be optimize by the LSTM mechanism.