**CS402 Reinforcement Learning**



School of Undergraduate Studies, DGIST

201611161 Jung wonkyun

201611182 Choi donghyeon

Final-report

2022. 05. 31

**Introduction**

Since 2017, interest in cryptocurrencies, including bitcoin, has increased, and the price of bitcoin, which was around 8 million won in early 2017, has risen at least 10 times to a high of 80 million won in 2021. As a result, many people started investing in Bitcoin. In many cases, the price of Bitcoin could not be easily predicted in the next state based on price volatility of up to 50% or more. Explosive price volatility from the end of 2017 to January 2021 shows that bitcoin is a high-risk asset.

Many promising results have been reported from the supervised learning community on the possibility of building a profitable trading system. More recently, several studies have shown that even the problem of integrating cryptocurrency price prediction results with trading strategies can be successfully addressed by applying reinforcement learning algorithms. Motivated by this, we present a Bitcoin trading auto-machine that attempts to further enhance the performance of reinforcement learning-based systems. The proposed approach incorporates multiple Q-learning and Deep Q learning agents, allowing them to effectively divide and conquer the cryptocurrency trading problem by defining necessary roles for cooperatively carrying out cryptocurrency pricing and selection decisions.

In this project, we present an automatic Bitcoin trading machine to further improve the performance of reinforcement learning-based systems. Q-Learning was learned using yahoo finance data. And it was implemented using short-term stock trading data. The model uses n-day windows of closing prices to determine if the best action to take at a given time is to buy, sell or sit. As a result of the short-term state representation, the model is not very good at making decisions over long-term trends but is quite good at predicting peaks and troughs.

The simple prediction of future prices with RNNs or CNNs is not enough to make mostly correct decisions in the crypto-trading world given the complexity and volatility of such environment. One possible solution could be to use reinforcement learning in combination with some clever deep neural network optimized policies.

This Final-report is an attempt to see if reinforcement and Q-Learning can be used to superhumanly predict and act on cryptocurrency prices and position. This model uses a Model-free Reinforcement Learning technique called Deep Q-Learning (neural variant of Q-Learning). At any given time (episode), an agent observes its current state (n-day window stock price representation), selects, and performs an action (buy/sell/hold), observes a subsequent state, receives some reward signal (difference in portfolio position) and lastly adjusts its parameters based on the gradient of the loss computed.

**Literature Reviews**

Cryptocurrency trading is an optimization problem that involves a sequential decision-making process. Recently, much effort has been made to solve this stochastic optimization problem using a learning-based approach, the RL approach. To formulate this stochastic optimization problem, it is necessary to determine how to measure the features of the stochastic components corresponding to changes in the Cryptocurrency trading is an optimization problem that involves a sequential decision-making process. A recent research direction is optimizing a trading strategy using RL such that a learning agent develops a policy while interacting with the financial environment. Using RL, a learning-based method, the learning agent can search for an optimal trading strategy flexibly in a high-dimensional environment. Unlike supervised learning, RL allows learning from experience, meaning that the agent can be trained with unlabeled data obtained from interactions with the environment.

Contribution of Park et al’s study is applying the DQN algorithm to derive a portfolio trading strategy in the practical action space. However, applying DQN to portfolio trading has some challenges. To overcome these challenges, Park et al devise a DQL model for trading and several techniques. First, Park et al introduce a mapping function to handle infeasible actions and derive a reasonable trading strategy. Trading strategies derived from RL agents may be unreasonable for real-world applications. Thus, Park et al apply a domain knowledge rule to develop a trading strategy with an infeasible action mapping constraint. As a result, this function works well, and Park et al can derive a reasonable trading strategy. Second, Park et al design a DQL agent and a Q-network to consider the features of multiple assets, and we derive a multi-asset trading strategy in the practical action space that determines the assets’ trading directions by overcoming the dimensionality problem. (Park et al, 2020)

Paper from Lee et al has explored the issues of designing a multiagent system that aims to provide effective decision support for the daily stock trading problem. The proposed approach, which was named MQ-Trader, defines multiple Q-learning agents to effectively divide and conquer the stock trading problem in an integrated environment. Lee et al presented the learning framework along with the state representations for the cooperative agents of MQ-Trader and described the detailed algorithms for training the agents. (Lee et al, 2007).

The generation of the optimal dynamic trading strategy is a crucial task for stock traders. Chakole et al tried to find an optimal dynamic trading strategy using the Q-learning algorithm of Reinforcement Learning. The performance of a trading agent using the Q- learning method is firmly based on the representation of the states of the environment. Chakole et al proposed two models based on the representation of the states of the environment. (Chakole, 2021).

With this paper, Schnaubelt demonstrate that state-of-the-art deep reinforcement learning can successfully learn superior order placement strategies to optimize execution. Execution optimization is highly relevant for both professional asset managers and private investors as execution quality affects portfolio performance at economically significant levels and is under regulatory supervision. This paper contributes to the existing literature in the following way. describe in-depth how deep reinforcement learning can be applied to the problem of optimal limit order placement. (Schnaubelt, 2020)

**Design**

* Q-learning

In Q learning, we define the accumulated reward as and use a Q function Q(St, At) to approximate the maximum accumulated reward, which can be updated by the Bellmen equation:

The Q learning algorithm (i.e., value iteration) is therefor as follow:

텍스트이(가) 표시된 사진

자동 생성된 설명

Figure 1. Q-learning algorithm

The base algorithm for DQN, Q-learning, is value based RL, which is a method that approximates an action value (i.e., a Q-value) in each state. Further, Q-learning is a model-free method such that even if the agent does not have knowledge of the environment, the agent can develop a policy using repeated experience by exploring. In addition, Q-learning is an off-policy algorithm, that is, the action policy for selecting the agent’s action is not the same as the update policy for selecting an action on the target value.

* Deep Q-Learning

This project uses a Model-free Reinforcement Learning technique called Deep Q-Learning (neural variant of Q-Learning). At any given time (episode), an agent absorbs its current state (n-day window stock price representation), selects and performs an action (buy/sell/hold), observes a subsequent state, receives some reward signal (difference in portfolio) position) and lastly adjusts it's parameters based on the gradient of the loss computed.

In traditional Q learning, to keep a Q table, we can only have finitely many states. But if we approximate the Q function by a neural network instead of a table, we can have infinitely many states.

Follow algorithm to implement the deep Q learning trading agent:

텍스트이(가) 표시된 사진

자동 생성된 설명

Figure 2. Deep Q-learning algorithm

**Data**

* Finding Price History Datasets



Figure3. Historical Bitcoin

In this project, we are planning to consider the most popular cryptocurrencies in the portfolio, Bitcoin (BTC), which is in large trading volumes so that the price movement will not be controlled by someone easily. The daily price movements of BTC have been collected in Crypto Data in CSV format and the data time range is from Nov 2014 to Mar 2022. The data includes the daily open price, high price, low price, close price, and the volume. For simplicity, we will consider the Open price and the volume\_(BTC) in this project.

텍스트이(가) 표시된 사진

자동 생성된 설명Figure4. Data Sample of BTC price from Crypto Data, from 2014/11/28 to 2022/03/01

Timestamp is start time of time window (60s window), in Unix time. All prices are in USD currency. Volume\_(BTC) transacted in this window. Volume\_(Currency) transacted in this window.

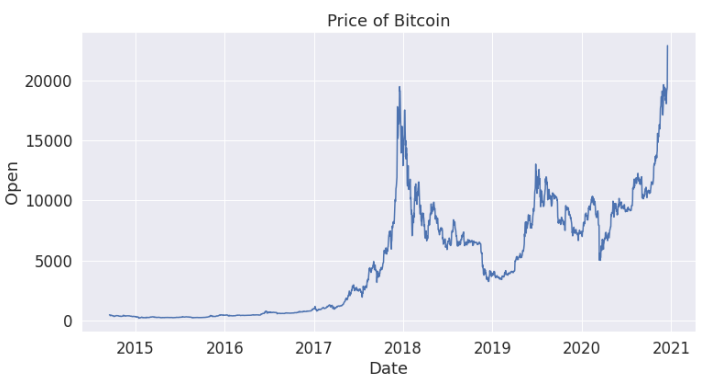


Figure5. Price of Bitcoin, from 2014/11/28 to 2022/03/01

The following is a graph plotting the BTC Data Sample mentioned above. We will use it to learn the price of Bitcoin from late 2014 to late 2022, and we will see if the model through reinforcement learning is properly profitable.

* API Dataset

텍스트이(가) 표시된 사진

자동 생성된 설명

Figure 6. Results the latest 200 hours of Bitcoin data with the pyupbit library

Through training, we will not only create an automatic trading machine, but also try to see if the trained model can predict the future Bitcoin price. The API of the most recent date and closing price of Bitcoin is imported into the library and the procedure is performed based on this data.

* Time Interval of Dataset

In the initial plan, as the cryptocurrency price is more volatile than stock market, it is aimed to get price movement with shorter time frame. However, with the consideration of the limited access of history data and extreme high training time, the price dataset is in daily interval which simplifies the training process.

**Methodology**

Although the result could tell humans whether the price will go up or down in the future, if we would like to implement a fully automated robot which authorize the right to execute actions on” buy”,” sell” or” hold”, machine learning approaches might not be the best solution as it still needs to involve human’s interruption. Therefore, this project is going to implement the automation with reinforcement learning.

* Advantages of Reinforcement Learning

Reinforcement learning is a kind of simulation of human beings. When facing with unfamiliar environment, it will first try the possible actions randomly. After getting more experience, it will adjust the policy to correct the error made before and execute a more optimal action in the next state. The algorithm will then keep improving with more training loops. After training, we can also know how the agent behaves in each state. Comparing with other prediction methods, RL can solve complex problems which traditional one would not be able to solve.

Cryptocurrency trading is a good example to implement RL as it could teach humans when the best time is to” buy” or” sell” the coin. Also, as pricing pattern could be changed in the future, RL can explore more possibilities which do not encounter in the previous data by learning from the mistake.

* Environment

In this project, to simplify the formation of investment portfolio, therefore in the environment setting, we assume that a user usually trades with only one type of cryptocurrencies at the same time, which is Bitcoin (BTC). However, commission fee will be also considered in the simulated environment.

* Action

 m: remaining cash

 r: rate of commission fee

 pi: the price of each cryptocurrency

 ni: the number of shares of each cryptocurrency (minimum amount to execute the trade: 1)

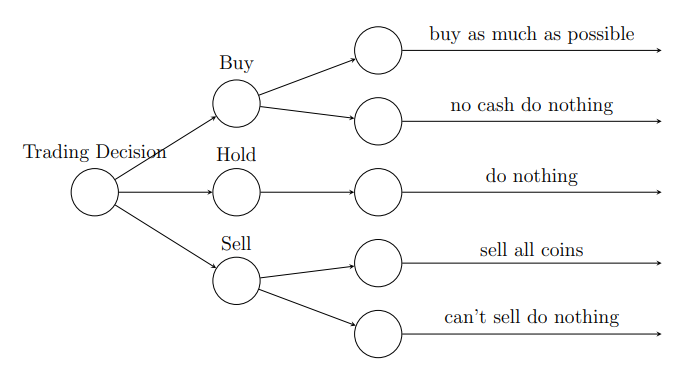


Figure7. schematic diagram of the Illustration of Trading Decision.

* Rewards

The reward is the change in portfolio value between the current state and the previous state:

텍스트, 손목시계이(가) 표시된 사진

자동 생성된 설명

* Agent

The agent will follow the Epsilon-Greedy Algorithm. In this algorithm, we can decide which action to take in terms of exploration and exploitation.

텍스트이(가) 표시된 사진

자동 생성된 설명

The neural network is formed by a multi-layer perception, which consists of 8 inputs, corresponding to the 8 states on a single-cryptocurrency trading environment. A hidden layer with ReLU activation is created between the input layer and the output layer. The number of hidden neurons has configured to 64. Finally, output layer is formed which is corresponded to the three different actions defined previously. MSE will be used on calculating the loss and Adam will be used for the optimizer.

**Experiment Setting**

* Experiment Environment

All the process, including data prepossessing and reinforcement learning are operated in Jupyter Notbook environment, configured with a Nvidia Geforce GTX 1050 GPU, Intel(R) Core(TM) i7-7700HQ CPU, 8.00GB memory capacity.

* Training and Testing Data

The history price data has been split into training data and testing data.

|  |  |
| --- | --- |
| Data Type | Time Range |
| Training | 2014-11-28 to 2022-03-01 |
| Testing | 2022-03-02 to 2022-05-30 |

* Hyper-parameter Setting

|  |  |  |
| --- | --- | --- |
| Hyper-parameter | Value | Description |
| Money | 500000 | Initial capital |
| Epoch | 10 | Number of passing through the entire training dataset |
| Batch size | 32 | Number of samples propagating through the network |
| Discount rate γ | 0.95 | Importance of considering rewards in the distant future |
| Exploration rate | 1 | Probability of choosing actions in random |
| decay rate | 0.995 | Decrease the probability of choosing actions in random |

**Results**

* Training Result (10 epochs)

Figure 8 below shows the results of the trained model.

텍스트이(가) 표시된 사진

자동 생성된 설명

Figure8. Training Result for 10 Episode

In Figure 8, the total revenue of the model trained with a total of 10 epochs is as follows. It is seen that a fairly volatile return is obtained when running 10 epochs. As the number of episodes increases, the volatility of revenue will converge.

* Training Result (100 epochs)

The following is a graph showing the results after learning at 100 epochs.



Figure9. Trend of Training Data Rewards within 100 epochs

The following is a graph showing the revenue for each Episode when learning with 100 epochs is carried out. It is seen that while the number of episodes is small and the volatility is high, the total profit gradually increases as the episode progresses, and the value also converges to a constant value.

* Buy-Sell propensity in training model

The following graph expresses the buying and selling trends of learning data by epoch. The orange dot means buy BTC and the blue dot means sell. Here, if we want to make the most profit, we can predict that the profit rate will increase if the sales point tends to be located higher than the purchase point.

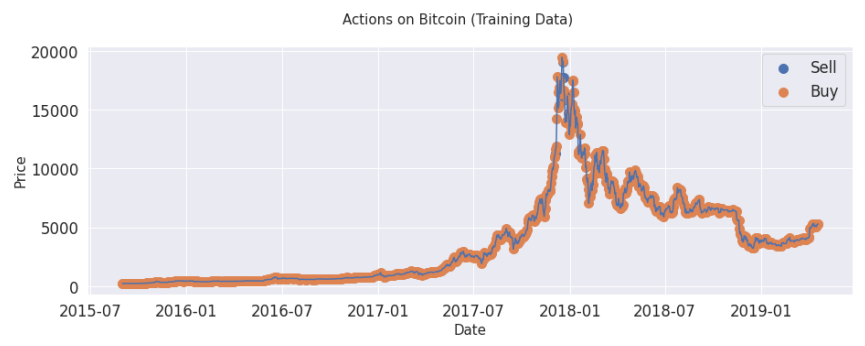


Figure10. Trend of Training Data Rewards within 10th epochs

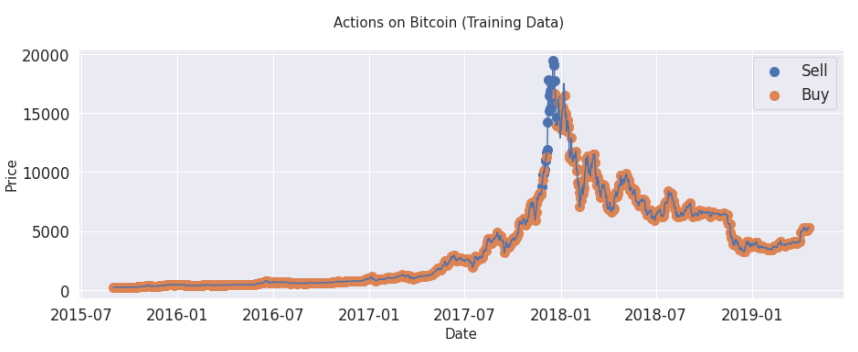


Figure11. Trend of Training Data Rewards within 55th epochs

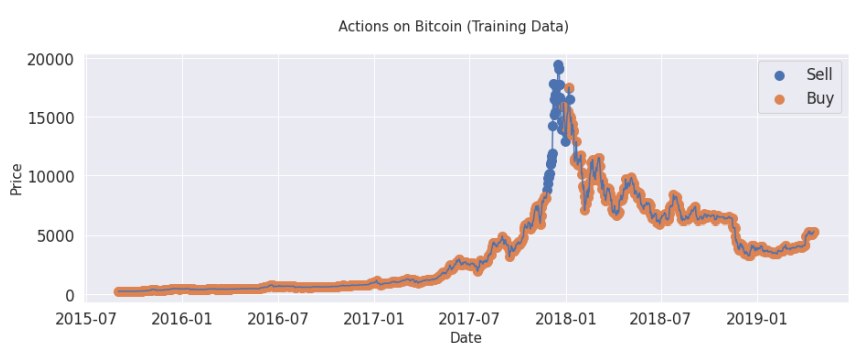


Figure12. Trend of Training Data Rewards within 80th epochs

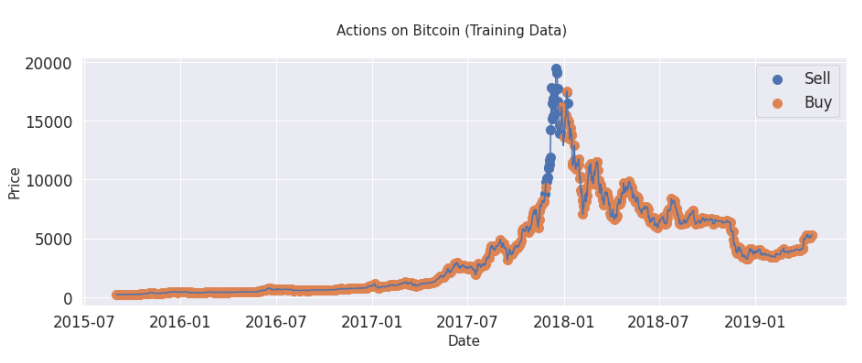


Figure13. Trend of Training Data Rewards within 100th epochs

Examining the buy sell tendency from epochs 10 to 100, it is seen that when the number of epochs is small, there is little difference between buying and selling, so there is no meaningful profit. However, as we go to the 55, 80, and 100 epochs, we can see the result that the automatic trading machine buys coins at a low price and sells them at a high price to make a profit. If training is performed with more than 100 epochs, it will converge to a certain maximum return, and we can expect more than the confirmed return.

* Testing Result

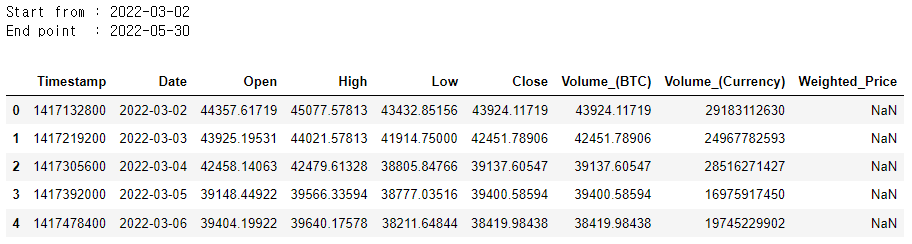


Figure 14. Test Data Sample of BTC price from Crypto Data, from 2022/03/02 to 2022/05/30

Among the training models previously conducted, the model with the highest yield was selected and evaluated for Bitcoin data from March 2, 2022, to May 30, 2022.

테이블이(가) 표시된 사진

자동 생성된 설명

Figure 15. Learning Process for Test dataset

['-$190104.89', '+$255050.92', '+$959016.59', '-$108321.19', '-$2026325.06', '+$0.00', '+$0.00', '+$0.00', '+$0.00', '+$0.00’… ]

As a result of the evaluation, it is seen that each model produces various profit values, and in the case of the model with the largest profit among them, it is seen that the profit of $959016 was obtained with a capital of $500000, and almost doubled the profit.

텍스트이(가) 표시된 사진

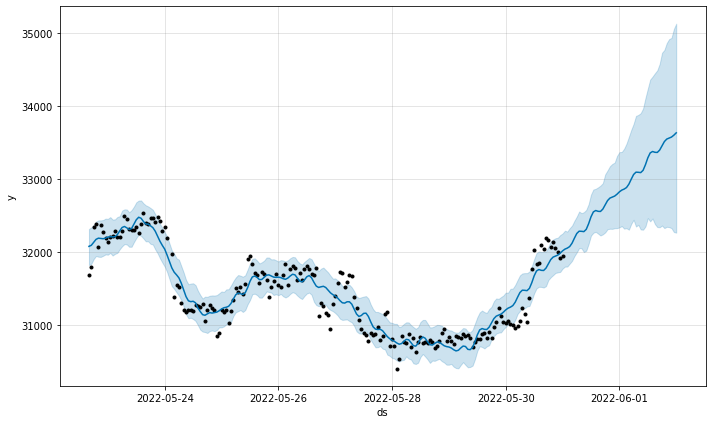
자동 생성된 설명

Figure 16. The result of considering the exchange rate in the starting price and extracting it by time

We decided to do an experiment on whether the model we learned can predict the future Bitcoin price, not just the process of finding the model that yields the maximum profit. Also, recently 200 data sets (every hour) were called into the API and applied to the highest profit model found earlier. And when the forecast was conducted, interesting results were derived.

The graph below is the result of predicting how the price of Bitcoin will change on May 31st and June 1st. As our model repeats buying and selling from May 22, we predicted future price changes. In Figure 17, it is predicted that Bitcoin will show an uptrend on May 31st and June 1st, and Figure 18 predicts that the price increase will be the highest from about 10:00 am to 1:00 pm.

Afterwards, I applied a module that predicts the future price by inputting model data and actual price as input among the open-source prophet library provided by meta. The future prediction was made based on our model to which the reinforcement learning algorithm was applied with the forecast () and predict () functions.

Figure 17. With May 30, Bitcoin Price Prediction for May 31st and June 1st

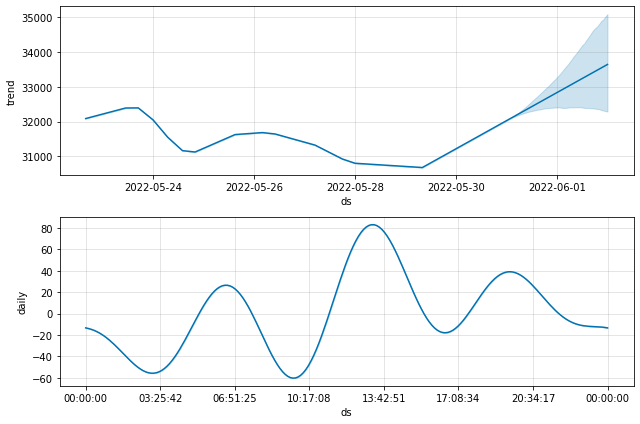


Figure 18. Bitcoin Price Prediction and Bitcoin Price Plot Over Time

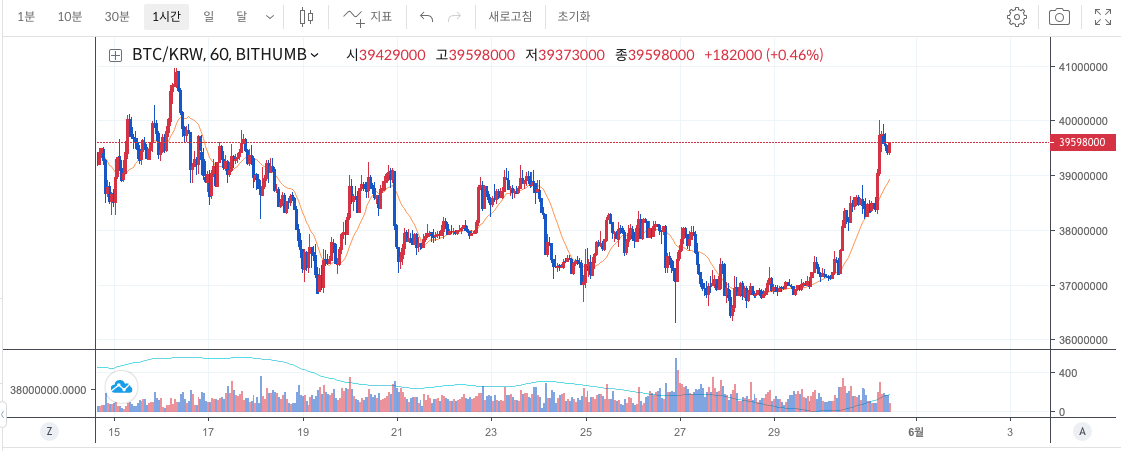
****

Figure 19. Real Bitcoin Price for each day

Figure 19 shows the actual Bitcoin chart from May 15th to May 31st. Looking at the chart, we can actually see that the price continued to decline from May 15th to May 30th, and from May 31st it shows an uptrend again. This shows the same results as the future prediction results shown in Figure 18.

****

Figure 20. Real Bitcoin Price for each hour on May 31

Figure 20 shows the results from the early morning of May 31, when the bull market starts to 12:30. Similar to the prediction shown in Figure 18, it shows an upward trend, reached the lowest price around 10 o'clock, and rose, showing the highest price until about 12:15. Therefore, it shows that we can actually make a profit when we buy and sell Bitcoin at a similar point in time as predicted. However, there is a slight error in time, so it is necessary to pay attention to this.

**Conclusion**

* Conclusion and Similar Research of Auto trading bot

Based on our results alone, it is seen that when Q-learning and DQN are used, good returns can be obtained by testing the latest data. It did not stop here, but we predicted the future price of Bitcoin based on the model we made. As a result of comparing the prediction of the reinforcement learning model with the actual Bitcoin fluctuation, it is seen that the results are almost similar except for some errors. Looking at this, it is seen that Q-learning and DQN, which were used as reinforcement learning methodologies, can be applied to reality.

Here, in another report that conducted a study similar to ours, there is another example where a different reinforcement learning algorithm is applied, and I will introduce it. According to Cano (2021) He use Double Deep-Q-Learning, Deep Actor-Critic, Proximal Policy Optimization, Deep Deterministic Policy Gradient, Twin Delayed DDPG, Soft Actor-Critic Algorithm to make Auto trading model.

* Analysis of Global Results

• Agents.

The use of frameworks allows you to implement and see the different behaviors of the agents quickly but with a big penalty as it is the understanding of how to create these models internally at the programming level and the loss of customization and experimentation. Stable-Baselines is an excellent framework for beginners who start experimenting with the world of reinforcement learning. Still, for models where production is required, such as the different tests we have created, it is necessary to create custom architectures that allow us to address the problem in the most optimal way possible. Furthermore, in the case of a bug because we don’t know what behavior the model is having, such as the result of our Actor-Critic, it is quite complex to analyze and try to solve it only by changing hyper-parameters. One of the serious problems of using this type of frameworks is the depth of the networks, as they tend to use generic networks of few layers for simple problems and that everyone can run the library. So, with this type of problems, it is a volatile market that makes agents not able to learn, agents conclude that it is best not to bet, but keep your money in your wallet so they hardly make movements of buying and selling cryptocurrencies, not to even mention that some models do not even try.

• Environment.

The environment created is quite simple and has great importance, almost as important as the design of the agents because it is the room in which the agents are going to interact, using the programmed rules of the game. However, it has several problems that I have been able to conclude after experimenting with the different models. First, it has a bottleneck with the transmission of information to the GPU, which means that the GPU practically works at only 30% of its capacity. Second, it has another bottleneck in the access search of the rows to obtain the information. Finally, it has not implemented parallelism, so it is impossible to train with multi-agents which would improve trying different architectures or strategies in a much faster way.

Another serious problem it has is the feedback of the reward to the agents. For example, that in a bull market like Bitcoin, since its creation, a programmed reward such as valuing the current total equity where is the sum between the value of the cryptocurrencies in FIAT and the own money retained in the portfolio make it too simplistic. This means that in case of interacting an agent with the market what he will do is buy and hold because he knows that the market will continue to rise. That said, it seems pretty smart to bet on the long. Still, we want to make even bigger profits and the possibility of being able to buy even more cryptocurrencies, and with such simplistic formulas, the trader becomes conservative.

**References**

Chakole, J. B., Kolhe, M. S., Mahapurush, G. D., Yadav, A., & Kurhekar, M. P. (2021). A Q-learning agent for automated trading in equity stock markets. *Expert Systems with Applications*, *163*, 113761. <https://doi.org/10.1016/j.eswa.2020.113761>

Lee, J. W., Park, J., O, J., Lee, J., & Hong, E. (2007). A Multiagent approach to Q-Learning. for daily stock trading. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *37*(6), 864-877. <https://doi.org/10.1109/tsmca.2007.904825>

Park, H., Sim, M. K., & Choi, D. G. (2020). An intelligent financial portfolio trading strategy using deep Q-learning. *Expert Systems with Applications*, *158*, 113573.

https://doi.org/10.1016/j.eswa.2020.113573

Schnaubelt, M. (2020). Deep reinforcement learning for the optimal placement of. cryptocurrency limit orders. European Journal of Operational Research, 296(3), 993-1006. <https://doi.org/10.1016/j.ejor.2021.04.050>

Cano, J. (2021). AutoTrading with Reinforcement Learning. *Departament de Matemàtiques i. Informàtica* (Attached to our zip file)