

Axel Wikner

# Bitcoin Trading using Reinforcement Learning

An Analysis of Q-Learning and DQN Algorithms on a Daily Timeframe.

Bachelor's thesis in Business Administration - Business Analytics

Supervisor: Denis M. Becker

April 2023



Norwegian University of  
Science and Technology



Axel Wikner

# **Bitcoin Trading using Reinforcement Learning**

An Analysis of Q-Learning and DQN Algorithms on a Daily Timeframe.

Bachelor's thesis in Business Administration - Business Analytics  
Supervisor: Denis M. Becker  
April 2023

Norwegian University of Science and Technology  
Faculty of Economics and Management  
NTNU Business School





## Preface

This bachelor thesis signifies the culmination of three years as an economic student at NTNU, specializing in business analytics. Early on I knew that either business analytics or finance would be my specialization of choice. Fueled by my passion for emerging technologies such as AI and blockchain outside of academia, I eventually opted for a specialization in business analytics. While digital transformation has been affecting the field of economics for many decades, it is only recently the use of AI has made its presence clear in areas such as automation, risk analysis, and trading. Through this bachelor thesis I have gained a much more profound understanding of AI, and in particular reinforcement learning. I have gained insight into its applicability in the economic domain, and primarily how it has the potential to greatly alter the competitive landscape of financial markets in the near future. The thesis has further reinforced my belief that AI will transform the labor market for both skilled trades and white-collar jobs in the next decade. However, the thesis has also dispelled the notion that AI is a mysterious black box as it may be perceived in the public eye. Reinforcement learning is rather an elegant information processing approach that leverages mathematical principles and computational power.

I would also like to direct a sincere thank you to my supervisor Denis Becker for the support in helping me develop in the aforementioned areas, as well as for being a critical contributor to the technical aspect of the thesis.

The content of this thesis is the sole responsibility of the author.



---

Axel Wikner

## Abstract

This thesis investigates the use of reinforcement learning algorithms, namely Q-learning and DQN, for trading bitcoin on a daily timeframe. The experimental results demonstrates that while all models trained without transaction costs technically outperform a buy-and hold strategy, they are highly unstable. The various Q-learning models appear to be making identical decisions, and we hypothesize that more training time is needed. One of the DQN models on the other hand, appear to be exhibiting hints of some rational and consistent trading behavior. Interestingly, it also ends up just being profitable. However, we argue this model is still far away from being applicable in a real-world trading-setting. Based on these findings, we conclude that further research is needed to develop more stable and rigorous models. Furthermore, it is recommended that future research incorporates things like a continuous action space, additional features, and primarily more time and computational power. Overall, the thesis highlights both the potential applicability and the challenging aspects of reinforcement learning in the world of trading.

## Sammendrag

Denne oppgaven undersøker bruken av reinforcement learning algoritmer, nemlig Q-learning og DQN, for handel med bitcoin på en daglig tidsramme. De eksperimentelle resultatene viser at selv om alle modeller utkonkurrerer en buy-and-hold-strategi (ikke tatt transaksjonskostnader med i beregning), er de svært ustabile. De ulike Q-læringsmodellene ser ut til å ta identiske beslutninger, og vi antar at det trengs mer treningstid. På den andre siden viser en av DQN-modellene en viss rasjonell og konsistent atferd. Interessant nok klarer den akkurat å bli lønnsom. Imidlertid hevder vi at denne modellen fortsatt er langt unna å være anvendelig i en virkelig situasjon. Basert på disse funnene konkluderer vi med at ytterligere forskning er nødvendig for å utvikle mer stabile og strenge modeller. Videre anbefales det at fremtidig forskning inkluderer ting som et kontinuerlig action-space, flere features, og først og fremst mer tid og datakraft. Samlet sett fremhever oppgaven både den potensielle anvendeligheten og de utfordrende aspektene ved reinforcement learning i handel med finansielle eiendeler.

## Table of Contents

<b>1</b>	<b><i>Introduction</i></b> .....	<b>1</b>
<b>2</b>	<b><i>Literature Review</i></b> .....	<b>2</b>
2.1	<b>Supervised Learning Models for Prediction Bitcoin</b> .....	<b>2</b>
2.2	<b>Reinforcement Learning Models for Bitcoin Trading Decisions</b> .....	<b>4</b>
<b>3</b>	<b><i>Theory</i></b> .....	<b>7</b>
3.1	<b>The Efficient Market Hypothesis and Random Walk Hypothesis</b> .....	<b>7</b>
3.2	<b>The role of Bitcoin in financial markets</b> .....	<b>9</b>
<b>4</b>	<b><i>Practical Conventions</i></b> .....	<b>10</b>
<b>5</b>	<b><i>Data</i></b> .....	<b>12</b>
5.1	<b>Data Sources</b> .....	<b>12</b>
5.2	<b>Input Features</b> .....	<b>12</b>
<b>6</b>	<b><i>Method</i></b> .....	<b>14</b>
6.1	<b>Specifications of the environment</b> .....	<b>14</b>
6.1.1	Environment with two decisions “in” or “out” .....	14
6.1.2	Environment with continuous decisions .....	16
6.2	<b>Reinforcement Learning</b> .....	<b>16</b>
6.2.1	Policy.....	19
6.3	<b>Deep Reinforcement Learning</b> .....	<b>20</b>
6.3.1	Q-Networks and Deep Q-Networks (DQN).....	20
6.4	<b>Testing the Algorithms with Simple Time-Series</b> .....	<b>21</b>
<b>7</b>	<b><i>Results</i></b> .....	<b>22</b>
7.1	<b>DQN</b> .....	<b>22</b>
7.2	<b>Q-Learning</b> .....	<b>28</b>
7.3	<b>Final Remarks</b> .....	<b>33</b>
<b>8</b>	<b><i>Conclusion</i></b> .....	<b>35</b>
<b>9</b>	<b><i>References</i></b> .....	<b>36</b>
	<b><i>Appendix #1 – Experiments with Different Time Series and RL Algorithms</i></b> .....	<b>40</b>



# 1 Introduction

Since its inception in 2009, bitcoin has emerged as the most renowned digital currency that is currently undergoing widespread adoption. Its use cases are widely debated, but nonetheless it has attracted noticeable attention from both retail and institutional investors. As of today, bots and automated trading make up most of the traded volume in the financial markets, and different forms of AI is also incorporated to gain an edge. Bitcoin is often an attractive arena for such trading techniques as the market never closes, the volatility is high, and there is access to large amounts of historical data. AI is more commonly incorporated into trading using, for example, LSTM networks to predict future prices. While some research indicates interesting results in this domain, there are also findings indicating that such an approach is comparable to the Naive method, which is simply predicting tomorrow's price based on the price today. Intrigued by these findings, this thesis instead explores the use of reinforcement learning. Traditionally, investors would like to know the reasoning behind a certain recommendation on whether to buy or sell an asset. Because it is hard to decipher the reasoning behind the output of a reinforcement learning model, this approach has not been as widely researched as other AI methods in the world of finance. This renders reinforcement learning a particularly exciting area to research when it comes to trading. In more detail, this thesis explores the effectiveness of two popular reinforcement learning algorithms, namely Q-Learning and Deep Q-Learning.

## 2 Literature Review

There have been numerous efforts in applying various analytical techniques to predict the price of bitcoin and exploit its volatility. In this literature review, we will examine the findings of previous research regarding AI and machine learning as an aid in trading and highlight the key-areas important to this paper going forward. In what follows, we will differentiate two strains of literature: The first strain is aiming purely at predicting bitcoin prices or returns, and models used for this task can mainly be classified as supervised learning models. The other strain looks at bitcoin trading decisions, where models belong to the family of reinforcement learning approaches.

### 2.1 Supervised Learning Models for Prediction Bitcoin

Huang, Huang, & Ni (2019) performed a study to determine if bitcoin returns are predictable by using price based technical indicators. The researched used a classification tree-based approach, and the findings suggest that their model has strong predictive power for certain ranges of daily returns on bitcoin. The team of researchers used 124 technical indicators from the “ta” library in pandas, and thus there was a variety of different types of indicators used as predictive variables. This model was trained on daily BTC-USD data, including the open, high, low, and close price of bitcoin. They used a sub-sample of the data to estimate and calculate the starting value of the 124 indicators. In the end, we are not told if they performed any testing to determine some sort of statistical significance for each indicator, something that would have been preferable. Similar findings are highlighted by Erfanian et al. (2022) whose aim was to predict the price of bitcoin in the context of micro-and macroeconomic theories such as cost-based pricing models. The research team used a variety of comparative approaches including OLS, Ensemble learning, SVR, and MLP to investigate how well the microeconomic, macroeconomic, technical, and blockchain indicators predict the price of bitcoin. The results are somewhat similar to the results by Huang, Huang, & Ni (2019) since the team claims that “*some technical indicators are significant short-run BTC price predictors, thus confirming the validity of technical analysis*” (Erfanian et al. abstract, 2022). Furthermore, the report also suggests that macroeconomic and blockchain indicators are significant long-term predictors. In terms of the different models used, it turns out that SVR was superior to other machine learning models. Nonetheless, the authors doubt whether machine

learning currently can outperform traditional models in BTC price prediction. It is important to note that while Huang, Huang, & Ni (2019) tried to predict returns, Erfanian et al. (2022) tried predicting price. This can be fundamentally different in the context of timeseries and machine learning.

A study by Chen (2023) also provides interesting findings with regards to predicting the price of bitcoin. This study used normalized daily bitcoin data to predict next day's price using random forest regression and LSTM. 41 input variables were used, among them bitcoin variables, blockchain activity, other cryptos, commodities, foreign exchanges, market indexes, tweets, and day of the week. The findings indicated several interesting conclusions. For example, it appears the Open, High, Low, and Close price of bitcoin using only 1 lag is the best predictor for the price in the next period, which aligns with the efficient market hypothesis. This indicates that the best predictor for tomorrow's price is simply today's price, which in turn could imply the random walk hypothesis is correct. The results also indicated that while the most important explanatory variables vary within the testing periods, variables like the SP500, JP225, Oil, and some other cryptos like ETH had predictive power. The study also found that incorporating redundant variables, or too few variables, decreased the model accuracy. In addition, both the LSTM and random forest algorithm, struggled with predictions when the price exceeded that of the highest price in the training data.

A problem with predicting the actual future price of an asset using machine learning models is often related to the normalization and the random walk hypothesis. Few of the research papers above have compared their models to the naïve stock prediction forecast, which is using the price in  $t-1$  to predict the price in  $t$ . In addition, as pointed out earlier, the longevity of some of these models could be questioned, as they are using normalized prices as a variable. This can easily result in a broken model as stock prices rarely only fluctuate between their previous min and max price. Furthermore, it could be argued that price prediction in and of itself is not intuitive. Rather, what an investor is more concerned with is the return, which is what Huang, Huang, & Ni (2019) successfully predicted.

A paper dealing with the previously mentioned issues regarding normalization and price prediction in a time series, is Nevasalmi, (2020). By using a multinomial approach, they attempted to predict the return of the S&P500 on a daily timeframe using various machine learning algorithms. The

team essentially established two thresholds of positive and negative return to reduce near-0% return noise. The findings indicated that decision tree models such as random forest and gradient boost were able to generate a statistically significant return prediction. The team also tested all the models in a trading environment, which resulted in all the models outperforming a standard buy & hold strategy.

A paper dealing with the previously mentioned issues regarding normalization and price prediction in a time series, is Nevasalmi, (2020). By using a multinomial approach, they attempted to predict the return of the S&P500 on a daily timeframe using various machine learning algorithms. The team essentially established two thresholds of positive and negative return to reduce near-0% return noise. The findings indicated that decision tree models such as random forest and gradient boost were able to generate a statistically significant return prediction. The team also tested all the models in a trading environment, which resulted in all the models outperforming a standard buy & hold strategy.

## 2.2 Reinforcement Learning Models for Bitcoin Trading Decisions

Sattarov et al. (2020) built a deep reinforcement learning model using tensorflow libraries and Keras API which purpose was to maximize short term accumulated returns when trading cryptocurrency pairs. The policy function used was stochastic, which means the output was a probability resulting in, for example, 0.8 buy, 0.1 sell and 0.1 hold. The reward mechanism was simply the sell price minus the previous buy price, and the size of the reward was directly proportional to this profit. With this approach, the agent seemed to show signs of increasingly improved results. Another interesting approach was the decision to use 4 multilayer models. In short this meant that the output of the 4 models were used to assess confidence in each of the three possible actions. If the highest confidence indicator was less than a certain threshold, this would mean the agent took no action whatsoever as the signals were deemed unreliable. The study tested the models on bitcoin, Ethereum and Litecoin, and compared the model to three other common investing strategies (Double Cross MA, swing trading, and scalping). This resulted in the model outperforming the other investing strategies convincingly in all cases, as well as making a profit in all cases. The researchers also conclude that simply using “buy” and “sell” actions might be the best for improving the model further.

Similar results were obtained in a study by Liu, et al. (2021). They argue deep reinforcement learning is better than earlier reinforcement algorithms such as Q-learning since such algorithms can only be applied to limited states and actions which need to be manually designed in advance. By using deep reinforcement learning, they argue that they can produce massive states on a longer time frame. Their proposed framework looked as follows:

- Create and initialize a gym trading environment.
- Setup the framework and trading sessions.
- Decide the basis of the policy function, the award function, and the optimization method.
- Train and test an agent and visualize the trading process.

Similar to the study by Munim, Shakil, & Alon, (2019), this study processed the data using the differencing method in order to get the bitcoin data stationary. This is because bitcoin is heavily influenced by seasonality which can negatively affect predictions. Subsequently, they also normalized the price-data using the minimum-maximum value normalization method. Again, the problem with normalizing price data in this way is that stock prices technically does not have an absolute minimum and maximum. The reward mechanism in this paper is the Omega-ratio, which is unique compared to other papers examined in this literature review. When comparing the policies LSTM, MLP, SVM, and TCN, the study found LSTM to be superior. This supports similar findings in other studies. The framework in the study utilized a PPO-based Agent and the Gym anytrading environment provided by Open AI. The study compared the model to various strategies such as golden/death cross, a momentum strategy, variable moving average, buy and hold, and more. The results indicated that the model outperformed the other strategies. In the testing period, the model made a 341% profit rate, while the second-best strategy (buy-and hold), made a 302% profit rate. The study concludes that the model manages to significantly outperform high-frequency trading strategies in times where the price is experiencing extreme surges upwards. This is because the agent constructed in the paper considers the long-term market trend information which high-frequency trading strategies do not.

Regarding deep reinforcement learning, Lucarelli & Borrotti (2019) used Double and Dueling Double Deep Q-Learning Networks to trade bitcoin with profitable results. The data was sourced from Kaggle.com on a 1-minute timeframe but aggregated into hourly data thus resulting in 30.000

data points. Furthermore, they compared two different reward mechanisms, one simply being profit, and the other being Sharpe-ratio. It turns out the Double Deep Q-Network with a Sharpe ratio reward function performed the best. This model was later tested on out-of-sample data and yielded an average return of 8% with a standard deviation of 2.77%. It is worth noting that the price also saw a positive surge of 182% followed by a 68% decrease in this period, resulting in a total (negative) return of 13%. Unfortunately, the result of the model in this timespan is not compared to any other traditional strategies, making it difficult to assess whether 8% profit is a good or bad. Similar results were found in a study by Zhang, Zohren, & Roberts (2020) who tested three different RL algorithms, DQN, PG, and A2C on 50 different future contracts in various assets classes. The inputs used for the models included normalized past price series, past returns, as well as the MACD and RSI technical indicators. Comparing the models to various other traditional momentum-trading strategies as well as a long-only strategy, on average the DQN model came out on top. DQN and A2C also managed to perform profitable results even when very high transaction costs were taken into consideration.

In terms of reinforcement learning, Li et al., (2019) proposes a slightly more sophisticated approach to trading with the use of AI. The team proposes a trading agent based on deep Q-network and actor-critic algorithms (A3C). Two interesting aspects of this paper is the way the data is pre-processed, as well as the action set of the agent. Firstly, the team filtered the financial time series data by utilizing a technique called SDAE that reduces noise, deal with non-stationarity, and increases the model's generalization-capabilities. Secondly, instead of defining the action space as "buy" and "sell" (1,0) the team defined multiple discrete positions (long, sell, short, cover). As a result, the agent could for example cover (close) a short position, without the need of having to go long simultaneously. This extended the action space to  $(-n, -n+1, \dots, 0, \dots, n-1, n)$  which represents the position of stock held in the next state. For example, if the previous action was 5 and the current action -2, the agent would currently sell 5 shares, and short 2 shares. These actions more effectively represent a real complex trading environment. The input variables were a hand full of market variables such as OHLC, as well as a few technical indicators. The result of the model was quite impressive as both the A3C and DQN-learning based of the SDAE out-performed a buy & hold strategy including slippage and transactions costs. The basic DQN and A3C without SDAE also outperformed the benchmark strategy, although not to the degree of the proposed models.

## 3 Theory

In the following chapter we will go over economic theories relevant to topic at hand.

### 3.1 The Efficient Market Hypothesis and Random Walk Hypothesis

The efficient market hypothesis (EMH) is a theory coined by Nobel prize winner Eugene Fama in 1979. The theory states that current prices in the stock market reflect existing available information. In essence, this implies that outperforming such a market would be highly unlikely (Baldrige, 2022). This theory is what partly has led to the popularity of index funds that track the benchmark stock market with low fees among investors. Since EMH predicts that the vast number of buyers and sellers in the market always have access to the same information, this leaves no room for any type of analysis or strategy that will outperform said market. The assumption that all market participants have access to the same publicly shared information about a stock, is the most important assumption underlying the theory. As mentioned by John H. Cochrane (2014), the EMH can be more precisely defined as “informationally efficient” markets. This implies that the market itself might at times act inefficient, but the point of the theory in this case is to highlight the fact that if an inefficient market move is not predictable, the market is informationally efficient. The result could in theory be a stock or commodity that acts what appears to be inefficient or irrational, but as long as it is also unpredictable, that means the market is informationally efficient. In other words, no one single actor can exploit the movement of the stock to outperform other market participants (Cochrane 2014).

This theory and its underlying assumptions undeniably result in what some would describe as radical predictions. For example, EMH implies that any trading strategy based on mathematical formulas, technical analysis, or a similar approach cannot outperform the market. A substantial body of research support these predictions. Barber et al. (2014) found less than 3% of day-traders can predictably make profit, and other research suggest that number might be much lower. Arguably, the fact that some percentage of traders do outperform the market could be used as an argument *against* EMH in this case. However, the findings indicate that the predictions of EMH are generally true. If EMH holds true, this would also imply that the multi-billion-dollar industry of the fundamental analysis (usually part of the business model of investment firms) is no more than picking lottery tickets. Studies on this topics result in similar findings as the studies regarding

profitability of day traders, meaning hardly any fund manager systematically outperform the market. As suggested by Cochrane (2014) this is both a surprising and a rather radical outcome, since professionals usually always outperforms the average man in any other field. There are, however, weaker forms of the EMH which assumes that new information which is not yet publicly available is not priced into the market. This would mean that a fundamental analyst or insider trader would be able to outperform the market in the short-term (Baldrige, 2022).

The predictions of the EMH are very similar to that of the Random Walk Theory. One could argue that one theory supports the other. The Random Walk Theory (RWT) simply states that stock prices are a random walk, and do not follow a specific trend or pattern. Therefore, trying to predict future market moves by means of technical, fundamental, or any other type of analysis, is futile. Followers of this theory often subscribe to the same teachings of the EMH, namely that the best strategy is a strategy that follows the market index as closely as possible (Park, 2018). The RWT is also partly built upon the findings that professional investors rarely outperform random pickings or amateurs, in accordance with the predictions of EMH. Critics of RWT, argue that trends and patterns do exist to varying degree within stocks, both in the short and long term. However, they may be very difficult, or in some cases impossible, to decipher (Park, 2018). Nonetheless, that does not mean that predictable trends and patterns are not prevalent critiques argue. A similar critique could be made about the EMH. Even if markets are “informationally efficient”, just having access to the information is not enough. An investor would also have to be able to 1: decipher or interpret the information, and 2: act upon the information (in other words buy the stock). This is the only way for the information to properly become incorporated into the market. If investors are not capable to decipher the information, they are also not able to act upon it. Furthermore, even if they can decipher the information, they must be able to buy the stock before the information becomes available to other investors. This is especially a problem in the world of stocks where the market is closed for trading generally two days per week. In conclusion, this could imply that contrary to what RWT and EMH proposes, stock prices could be predictable. The predictability of an asset would depend on to what degree a pattern exists within the data, and an investor’s ability to decipher it and act upon it. In the world of AI, where machines can decipher large amounts of data in record time, this is an intriguing critique.



### 3.2 The role of Bitcoin in financial markets

There have been plenty of speculation about how cryptocurrencies such as bitcoin will fit into the global financial markets. Proponents of bitcoin argue one of its main advantages is the scarcity, which is ensured by the process of “mining” bitcoin using the proof of work algorithm. This scarcity, proponents argue, is what makes bitcoin a hedge to inflation and inflatable assets such as fiat currencies. This aspect of bitcoin can therefore be compared to the benefits of gold or other scarce commodities. However, critics argue that the volatile nature of bitcoin does not make it suitable as a hedge against inflation. This critique is largely supported by the fact that bitcoin saw a 75% decline in value from its all-time-high in November of 2021 to November of 2022, when inflation around the world was running rampant. Despite this, bitcoin’s ability to serve as a hedge against inflation might not be a question of if, but rather of when. Some proponents argue that bitcoin will eventually stabilize, and by then its purpose as a hedge against inflation will be more pronounced. Whether bitcoin is a hedge to inflation or not, is up for debate. At the current time, it appears that bitcoin is behaving more like a risk-on asset with positive correlation to tech stocks, and negative correlation to traditional safe-haven assets such as gold (Wood, 2022). There is also the case to be made that the value and price of bitcoin is more so driven by its fundamentals and practical use-cases. The decentralized nature of bitcoin allows for it be used peer-to-peer without a middleman such as a bank. In 2021, El Salvador became the first country in the world to adopt bitcoin as official currency, and it appears that adoption of bitcoin globally in terms of practical use is growing steadily, irrespective of the volatile price (Browne, 2022). Furthermore, the practical adoption of bitcoin appears to be dominated by emerging markets in continents like Asia, Africa, and south America. This can be considered a testification of bitcoins fundamental value as a decentralized commodity and tender in places where stable financial institutions and infrastructure is scarce (“The Global Crypto Adoption Index”, 2022).

## 4 Practical Conventions

In order to ensure the validity of the results of the models, we need to make sure that the agents' decisions in training actually translates well into a real-world scenario. When trades are made on an exchange, there are several factors and properties affecting the trade which are not usually considered during the training of a model because of practical reasons. Some crucial factors requiring mentioning is market makers, orders, and transaction costs. In most larger tradeable assets on exchanges, there are market makers. These are actors which help facilitate trades on an exchange by providing liquidity to the asset. Essentially, market makers provide liquidity at the highest bid-price, and the lowest sell-price, thus enabling trades to be made immediately at these prices. The spread is the difference between these two prices, and it is how market makers make money. Orders placed in the market can either be market orders or limit orders. Market orders are immediately executed at the best available buy or sell price, while limit order specify a particular price to buy or sell at. Furthermore, each trade is affected by transaction costs. These are fees that the trader must pay to the exchange when placing the trade. Such fees can be constant, percentage-based, or both, and vary depending on the exchange, type of asset, regulatory domain, and other factors.

In the training of the model, we are making some basic assumptions about how trades are made on an exchange.

- 1. There is always liquidity.**

This means that we assume that whenever our agent wants to place a trade (market-order), there is always liquidity available at that certain price, and the trade is immediately facilitated. This might not always be true in a real-life scenario.

- 2. Our orders do not affect the market price.**

This means we assume that the order the agent place does not affect the asset price in any direction.

- 3. Transaction costs do not change.**

We assume that any transaction cost we include in the model, has been the same throughout history and will remain the same in a real-world scenario.

It is worth mentioning that the importance of these assumptions varies depending on the timeframe the model is trading at.

## 5 Data

The following chapter briefly goes over the set of data used, as well the input variables.

### 5.1 Data Sources

The bitcoin historical price on a daily timeframe is collected from investing.com. The reason being the availability of data goes all the way back to 2009. In order to reduce the number of NULL values, we only pick data from 1st Jan 2013. Generally, it is difficult to come across bitcoin data with a lot of history, however investing.com seems to be a very good candidate for a daily timeframe and higher. The daily data contains 3 726 rows. It would have been preferable to compare the models in different timeframes, but this is an area of exploration for future research.

### 5.2 Input Features

The number of input variables chosen as well as their type is largely influenced by computational power, and the findings of previous literature. The dataset has been equipped with 11 input variables that are displayed in table 1.

Table 1: Input variables for the models

<i>Abbreviation</i>	<i>Name</i>	<i>Type</i>
<i>Returns</i>	-	The variable the model tries to predict. Returns of the bitcoin close price.
<i>RSI</i>	Relative Strength Index	Technical momentum oscillator
<i>EMA(T=14)</i>	Exponential Moving Average	Technical trend indicator (Returns as input)
<i>TSI</i>	True Strength Index	Technical momentum oscillator
<i>RVI</i>	Relative Volatility Index	Technical volatility oscillator
<i>MFI</i>	Money Flow Index	Technical momentum oscillator
<i>STDEV</i>	Standard Deviation	Statistical indicator – Standard deviation of returns (Length=30)
<i>Open % Change</i>	-	Returns of the bitcoin open price from previous period.
<i>High % Change</i>	-	Returns of the bitcoin high price from previous period.
<i>Low % Change</i>	-	Returns of the bitcoin low price from previous period.
<i>Volume % Change</i>	-	Percentage change of volume from previous period.

Using oscillating and percentage-based input variables has mainly two benefits. 1) We can better ensure the longevity of the model since the risk of there being new all-time-highs or all-time-lows is severally reduced, particularly over a longer time frame. 2) We reduce some the inherent issues that non-normalized price data has. For example, by using percentage change instead of prices, we get rid of most of the trends within the data. There will still be volatility-clustering as well as some seasonality in the data, but addressing such issue will be a recommendation for further research.

## 6 Method

The following chapter goes over the technical aspects of reinforcement learning and explains the inner workings of the Q-Learning and DQN algorithm.

### 6.1 Specifications of the environment

#### 6.1.1 Environment with two decisions “in” or “out”

The very first environment we wish to create will only allow for two discrete actions, namely “in” or “out” of the market (1,0). “in” simply means to enter the market, or to stay in the market if we are already invested. “out” means to exit the market if we are invested, or to stay out if we are no longer invested. OpenAI has preprogrammed trading environment for these types of decision called gym-anytrading. After preliminary testing of this environment, we decided to develop our own. In this iteration of the environment, we do not allow for short selling. Technically, a “hold” or “do nothing” position is also not allowed as we are either in the market or out of the market. Below, the pseudo-code of the trading environment is shown. The term “stock” can easily be substituted for other assets such as commodities or cryptocurrencies. We apply the following notation:

$S_{\text{Out}}$	Value of stocks in the end of the iteration after the decision, and after the reward/return from the current decision.
$S_{\text{In}}$	Value of stocks in the beginning of the iteration before decision, but after the reward/return from previous decision.
$\tau$	Transaction costs when selling or buying stocks. There are no additional transaction costs for buying/selling bonds.
$r_{S,t}$	Return on stock investment
$r_B$	Interest rate on bonds/bank (this is constant throughout time). This can also be considered the risk-free return.

**Initialization:**

$$B_{\text{Out}} = B_0$$

$$S_{\text{Out}} = 0$$

This are the values from some fictive previous point in time.

**Iterate:**

The outgoing values from the previous iteration become the ingoing values:

$$B_{\text{In}} = B_{\text{Out}}$$

$$S_{\text{In}} = S_{\text{Out}}$$

Reading the return from the data that will be achieved by stocks ( $r_{S,t}$ ):

If  $a = 1$  (we go or stay long in stocks):

If  $S_{\text{In}} = 0$  (out of stocks, and long in bonds, i.e.,  $B_{\text{In}} \geq 0$ ):

We need to buy stocks: \*

$$S_{\text{Out}} = \frac{B_{\text{In}}}{(1 + \tau)} \cdot (1 + r_{S,t})$$

$$B_{\text{Out}} = 0$$

Else (i.e.  $S_{\text{In}} > 0$  and  $B_{\text{In}} = 0$  which means long in stocks, out of bonds):

We will stay in stocks:

$$S_{\text{Out}} = S_{\text{In}} \cdot (1 + r_{S,t})$$

$$B_{\text{Out}} = 0$$

If  $a = 0$  (we sell stocks or stay out):

If  $S_{\text{In}} = 0$  (out of stocks, and long in bonds, i.e.,  $B_{\text{In}} \geq 0$ ):

$$S_{\text{Out}} = 0$$

$$B_{\text{Out}} = B_{\text{In}} \cdot (1 + r_B)$$

Else (i.e.,  $S_{\text{In}} > 0$  and  $B_{\text{In}} = 0$  which means long in stocks, out of bonds):

We need to sell stocks: \*

$$B_{\text{Out}} = S_{\text{In}} \cdot (1 - \tau) \cdot (1 + r_B)$$

$$S_{\text{Out}} = 0$$

### ***Explanation of \*:***

We buy stocks and pay transaction costs on the value invested into stocks:

$$S_{\text{Bought}} + S_{\text{Bought}} \cdot \tau = B_{\text{In}}$$

Solving for  $S_{\text{Bought}}$ :

$$S_{\text{Bought}} = \frac{B_{\text{In}}}{1 + \tau}$$

On the stocks acquired we earn the return:

$$S_{\text{Out}} = S_{\text{Bought}} \cdot (1 + r_{S,t}) = \frac{B_{\text{In}}}{1 + \tau} \cdot (1 + r_{S,t})$$

### **6.1.2 Environment with continuous decisions**

There is also the possibility of an environment with continuous decision, similar to Li et al., (2019). This means that the agent will have a larger action space available, where it can choose to invest only parts of the capital. Equivalently, the agent can choose to only sell parts of the active positions as well. We hypothesize that similarly to the findings by Li et al., (2019), the agent should stay more heavily invested in long-positions, during in bull-markets/trends, and less invested in bear-markets/trends. It is reasonable to assume that such an action space and environment is more suitable for a financial timeseries where unforeseeable and volatile changes frequently occur. Li et al., (2019) found that tripling the action space (amount of stock held) from  $(-1,0,1)$  to  $(-3,-2,-1,0,1,2,3)$  resulted in a more than triple the reward. However, in this case, the agent had multiple stocks to choose from, so it is unclear in our case how this will affect an agent that only trades one asset. We can hypothesize that perhaps the agent will increase its position during times of less volatility, as well as adjust the size of its position according to the confluence of our input-variables. This type of environment is interesting, but beyond the scope of this thesis, and is therefore recommended for future research.

## **6.2 Reinforcement Learning**

Reinforcement learning uses the framework of Markov Decision Processes (MDPs). The ingredients of an MDP are states, decisions (actions) and rewards. A MDP is a mathematical framework to model the decision-making by our agent (Vijay Kanade, 2022). This is essentially



the mechanism of how the agents' actions in a particular state, leads to another state. MDPs can be illustrated in different ways.

Figure 1: Interaction between environment and agent in reinforcement learning

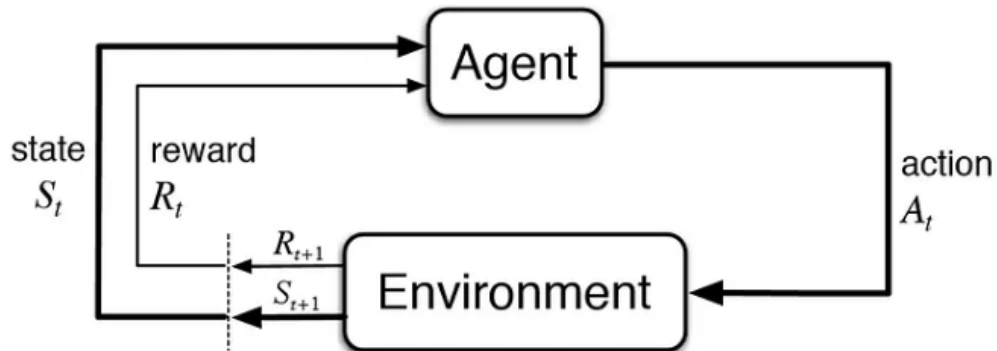
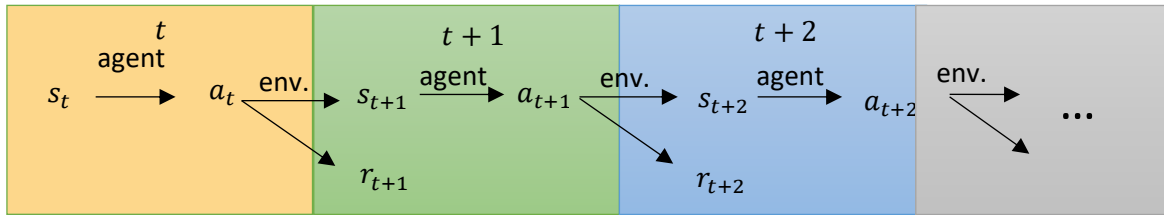


Figure 1 illustrates the interaction between the agent and the environment. The agent will observe its state  $s_t$ , and then pick an action which through the environment will land the agent in a new state,  $S_{t+1}$ . By performing an action, the agent will also receive a reward  $R_{t+1}$  which it will learn from. In our case the reward will be the profit, and the states will be comprised of various economic and technical indicators. In a real-world trading scenario, we cannot possibly observe all the variables which goes into explaining to rise or fall in the future price of an asset. We can merely observe some of these variables, and therefore this is arguably a so called partially observable Markov decision process (Williams & Young, 2007).

Another way to illustrate this process is shown below in figure 2.

Figure 2: Alternative illustration of Markov-Decision-Chain



At first glance, it may seem like the flow of time is essential in this process, and in terms of how much we train the model, we are using the term “timesteps”. However, the point in time is not crucial in such a decision-making process. Rather it is the state  $S$  that is important. An agent will, based on the state it finds itself in, find the optimal action. Therefore, a better, but simplified and canonical example is to represent the problem in the form of a table as in figure 3.

Figure 3: The relationship between states and actions in the Markov Decision process

	Action 1	Action 2	Action 3	Action n...
State 1 ( $s_t$ )	Estimated reward	Estimated reward	....	....
State 2 ( $s_{t+1}$ )	Estimated reward	Estimated reward	....	....
State 3 ( $s_{t+2}$ )	Estimated reward	Estimated reward	....	....
State 4 ( $s_{t+3}$ )	....	....	....	....
State n... ( $s_t$ )	....	....	....	....

In figure 3, we note that it is not time which is the essential ingredient, but it is the state and the action taken in this state which is important in reinforcement learning. Intuitively, the agent can exist in the same state at different points in time, and the goal is not necessarily reached at a given point in time. Should we illustrate this on a timeline, an example would look at follows in figure 4.

Figure 4: The relationship between states and actions illustrated on a timeline



Figure 4 more accurately illustrates the notion that the state and action is the important relationship in reinforcement learning rather than action-time.

### 6.2.1 Policy

With respect to the relationship between a state and an action, it is convenient to introduce the notion of policy. The policy is the agent's approach to picking a certain action in a particular state, and thus landing the agent in a new state. Simply put, this is the agent's strategy of picking actions which will achieve its goals. In more technical terms, many different policies make up a probability distribution of all the action-state pairs the agent currently must choose from. The agent then evaluates these policies by computing the utility function  $U$  over said policies. By doing such, the agent obtains the reward for each policy, and subsequently chooses the policy with the highest

reward (Gabriele De Luca, 2020). This is also known as a stochastic policy. At the very start, all of these policies are randomly generated, and through iterations, the agent will eventually (in theory) learn the optimal policy which will map a probability distribution of actions to each action space (MLK, 2021).

## 6.3 Deep Reinforcement Learning

### 6.3.1 Q-Networks and Deep Q-Networks (DQN)

A deep Q-Network is a reinforcement learning algorithm that combines the regular Q-learning algorithm with deep neural networks. One of the main differences between a Q algorithm the DQN, is the agents brain. In Q-learning, the Q-table, which is composed of states and actions, is the agent's brain. In a DQN the brain is the neural network, and the neural network takes the state of the environment as an input and calculates the expected reward for each possible action in the Q-table (Moghadam, 2019). One main advantage of the DQN is that it can handle high-dimensional state spaces, which a Q-Network cannot. This means we can work with many input variables in our data. This opens up the possibility of including variables that may be more far-fetched in the context of having an effect on price-percentage change. Another advantage of the DQN is its ability to learn "online", meaning it can learn from experiences as it receives it. The value-iteration method has the purpose of solving the Markov Decision Process which we are already familiar with. In the case of the DQN, this method initializes the Q-values to random values, and then iteratively updates these values as they eventually converge to their optimal value or state. The Q-value can simply be defined as the immediate reward for a particular state-action pair (see figure 3), plus the discounted value of the very next state (Ken, n.d.). The idea of discounting the value of the very next state is the same novel concept used in finance, namely that we value rewards today more than rewards in the future. The value-iterations to calculate all possible state-actions, is performed using a deep neural network. A common disadvantage of DQN is that the learning process can be very slow due to the use of deep neural networks. This type of approach is also prone to overfitting, and the learning process can also be unstable which is an inherent characteristic in neural networks.

## 6.4 Testing the Algorithms with Simple Time-Series

To get a sense of how the aforementioned algorithms apply to real-life data, they were first tested on deterministic data or very simplistic time-series. Firstly, we allow for the algorithms to trade on deterministic timeseries where the optimal decision should be very easily obtained. These timeseries have been divided into 4 types.

- Predetermined prices.
- Sine-Curve
- Sine-Curve with linear trend.
- Time series with predefined ups, downs

Some of the results are shown in the appendix. After having evaluated the algorithms on all the deterministic timeseries, we can conclude that Q-learning behaves as expected and that DQN generally behaves as expected. However, in timeseries with predefined ups and down, the DQN algorithm often fails. The reason for this is unknown, but could be due to the data set, hyperparameters, or number of timesteps. Regardless, the results generally warrants that we move forward with both algorithms.

## 7 Results

In the following we will dissect and compare the results between the DQN and Q-learning algorithm. The models have been trained on a dedicated PC. Each model is given 100 units of currency to trade with, and the result is compared to a simple buy-and hold strategy as that is what proponents of the random walk theory and efficient market hypothesis generally would suggest.

### 7.1 DQN

The hyper parameters used in training the DQN models are shown in table 2. Other useful settings and parameters are shown in table 3.

Table 2: Explanation of DQN hyper parameters

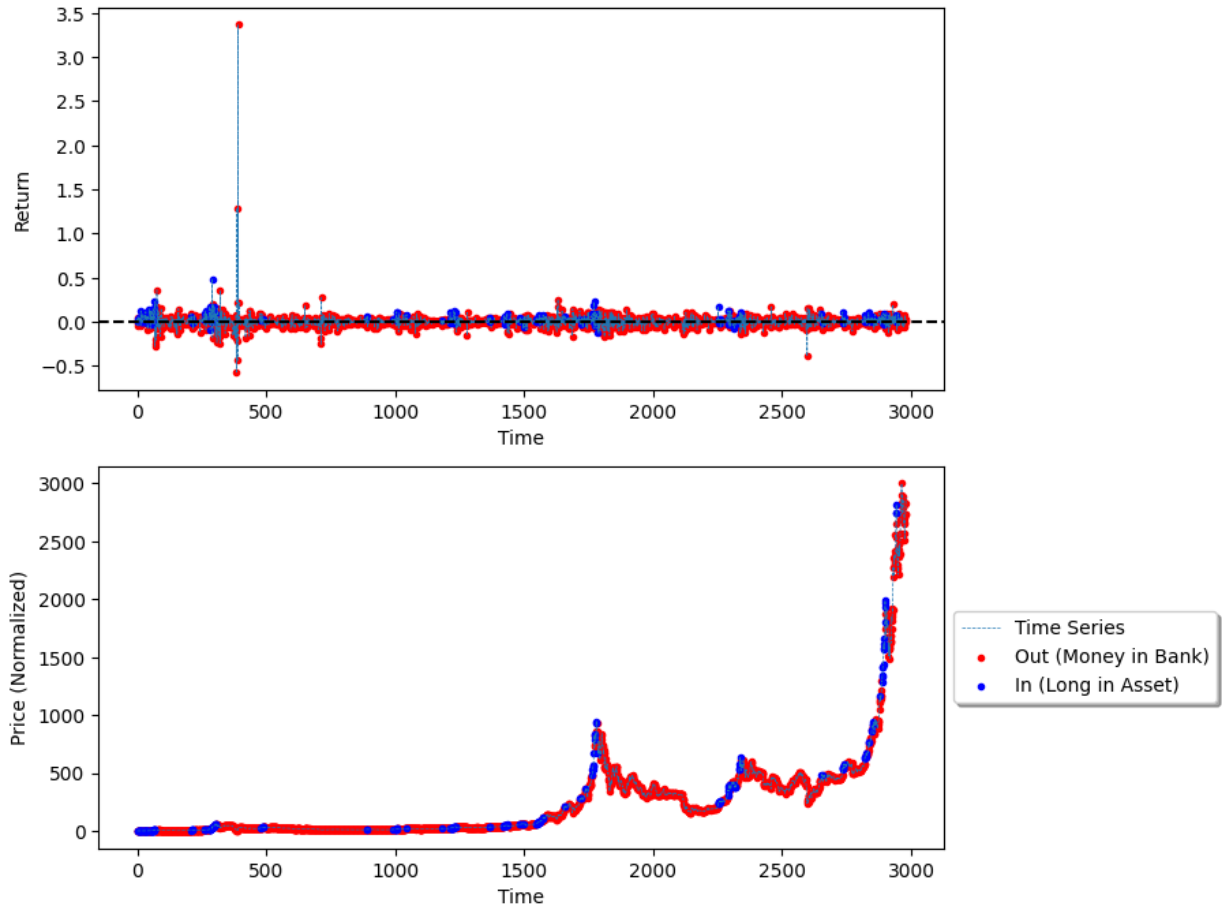
Hyper Parameter	Value	Explanation
<i>gamma</i>	1	Determines the present value of future rewards
<i>learning_rate</i>	0.05	Learning rate for adam optimizer
<i>exploration_fraction</i>	0.8	The probability of the agent taking random actions in the beginning
<i>exploration_initail_eps</i>	0.8	Length of the exploration period
<i>exploration_final_eps</i>	0.00	Probability of random actions in the end
<i>batch_size</i>	120	Size of a batched sampled from replay buffer for training

Table 3: Other DQN settings and parameters

Other settings	Training/Test Split	Epochs	Timesteps	Average Run time	Early Stopping Callback
	80/20	5,000	14,905,000	5hrs	600

## Training #1

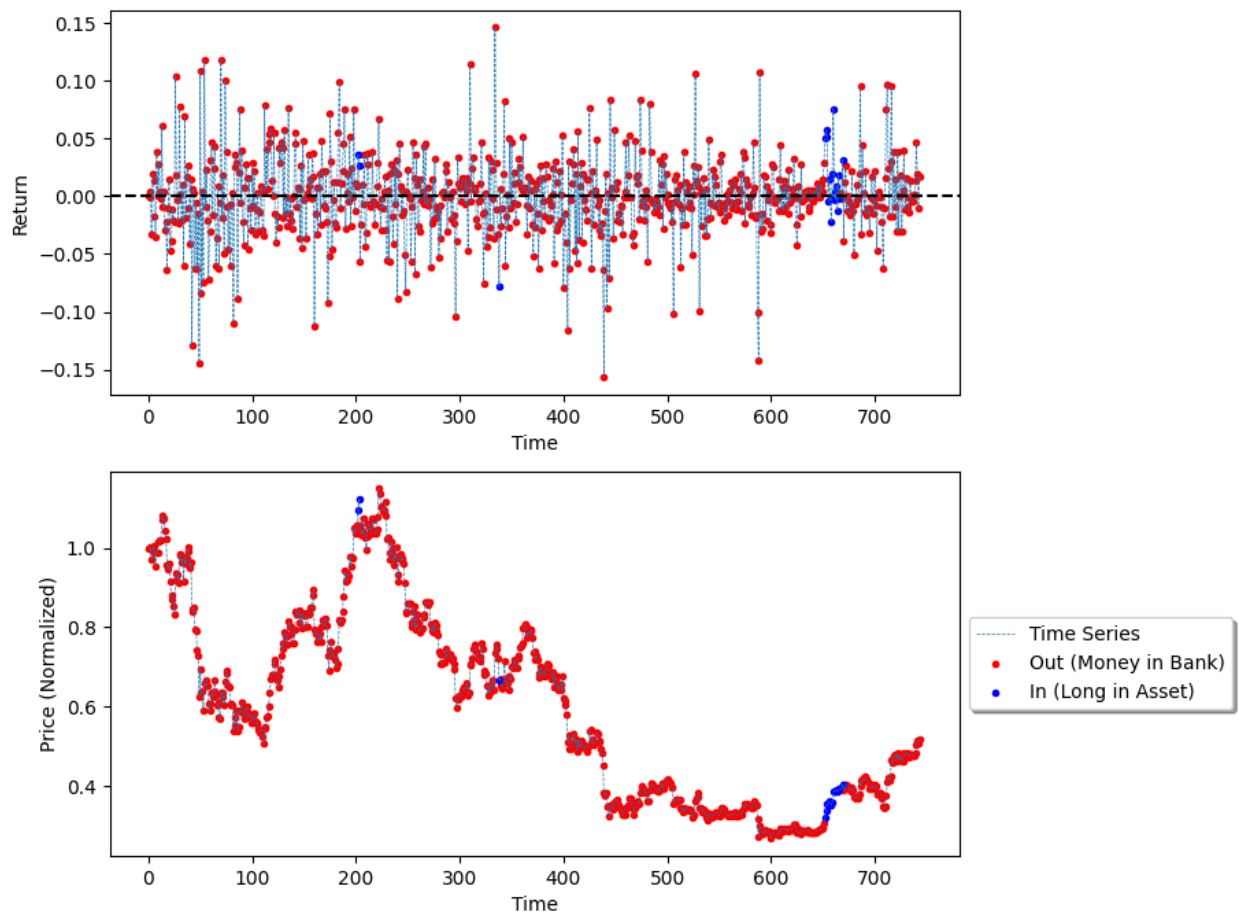
Figure 5: Graphical illustration of trading decisions in training DQN model #1



- The final profit of the DQN strategy is: 2153499865.072....
- The final profit of the buy-and hold strategy is: 288094.60....

## Test #1

Figure 6: Graphical illustration of trading decisions in testing DQN model #1

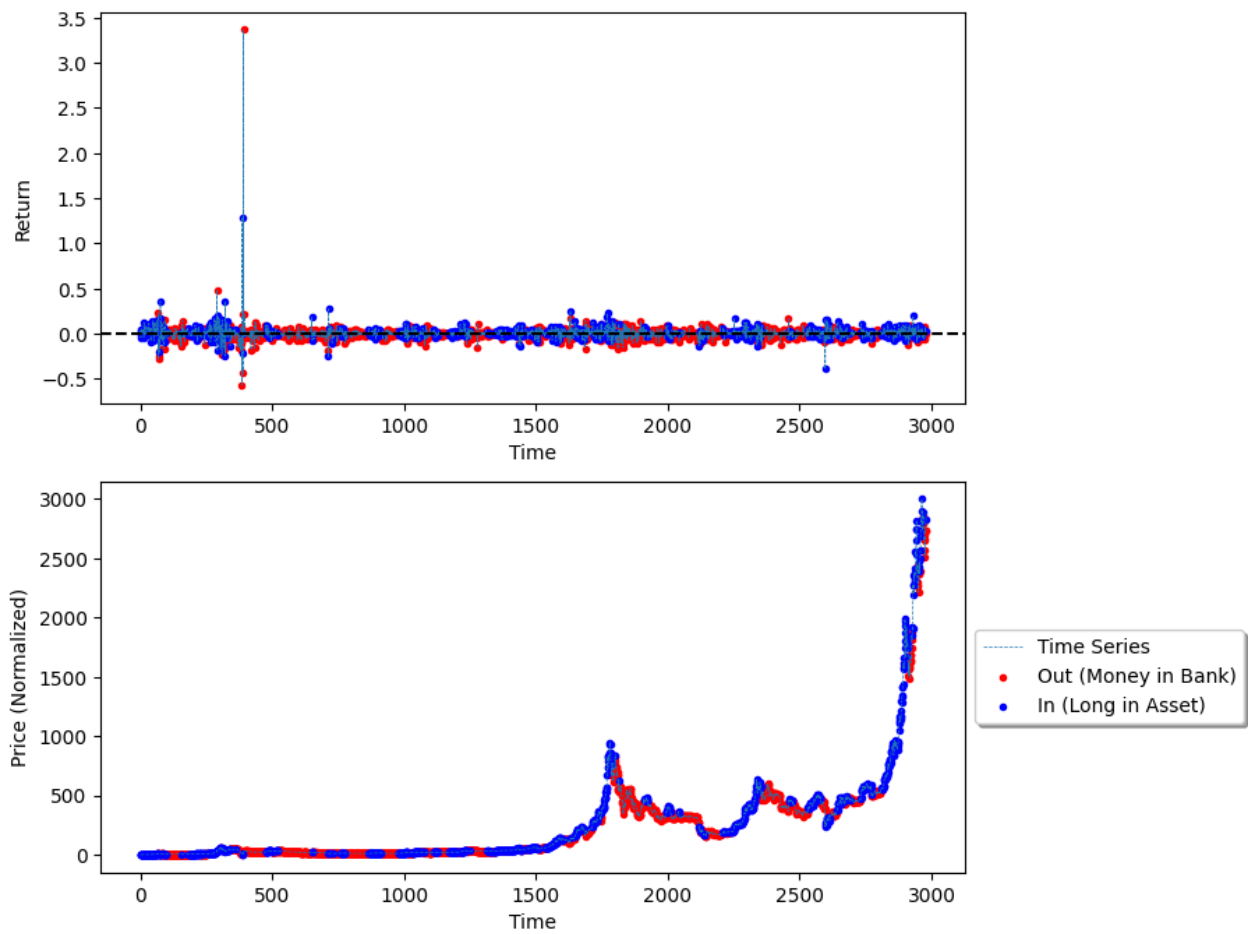


- The final profit of the DQN strategy is: 114.61.... (+14,61%)
- The final profit of the buy-and hold strategy is: 52.41....(-47,59%)



## Training #2

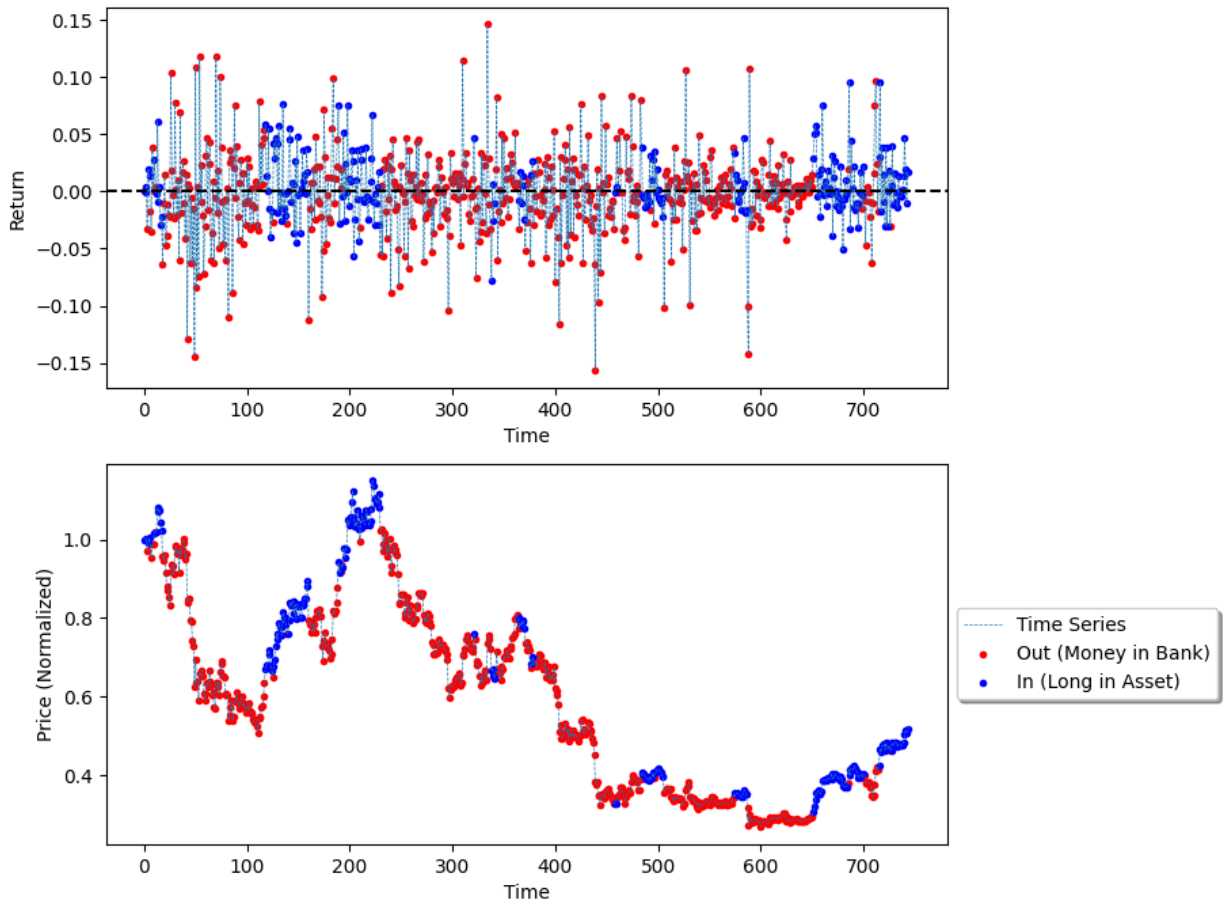
Figure 7: Graphical illustration of trading decisions in training DQN model #2



- The final profit of the DQN strategy is: 27541984.072....
- The final profit of the buy-and hold strategy is: 288094.60....

## Test #2

Figure 8: Graphical illustration of trading decisions in testing DQN model #2



The final profit of the DQN strategy is: 103.60.... (+3,6%)

The final profit of the buy-and hold strategy is: 52.41.... (-47,59%)

## Training #3

The final profit of the DQN strategy is: 21533944.072....

The final profit of the buy-and hold strategy is: 288094.60....

## Test #3

The final profit of the DQN strategy is: 100.65.... (+0.65%)

The final profit of the buy-and hold strategy is: 52.41.... (-47,59%)

While at first glance, it might appear that the DQN models have outperformed the buy-and hold strategy in every single case, the more evident takeaway should be the instability of the models. As derived from the results above, model #1 has behaved seemingly irrationally by staying out of the market almost exclusively. At some points in time, it enters and then exits market profitably. While it is impossible to say why the agent suddenly enters the market towards the end of the time period, this appears to be more of a random artefact rather than the result of any meaningful pattern-recognition. Despite this being the most profitable DQN model, we argue these results are not fully reliable. While it is interesting to note that all the models have outperformed the buy-and hold strategy, the validity and reliability of the models is up for discussion. The most interesting behavior of the DQN models is exhibited by model #2. This model appears to be making some consistent decisions and utilizing some sort of pattern. There is a trading pattern emerging from the model, where it enters the market after some type of bullish reversal, and then chooses to stay in the uptrend until the trend is broken. However, it is also important to note that these models were trained without transaction costs. Introducing transaction costs would likely result in all models trading at a loss.

## 7.2 Q-Learning

The hyper parameters used in training the Q-Learning model are shown in table 3. Other useful settings and parameters are shown in table 4.

Table 3: Explanation of Q-Learning hyper parameters

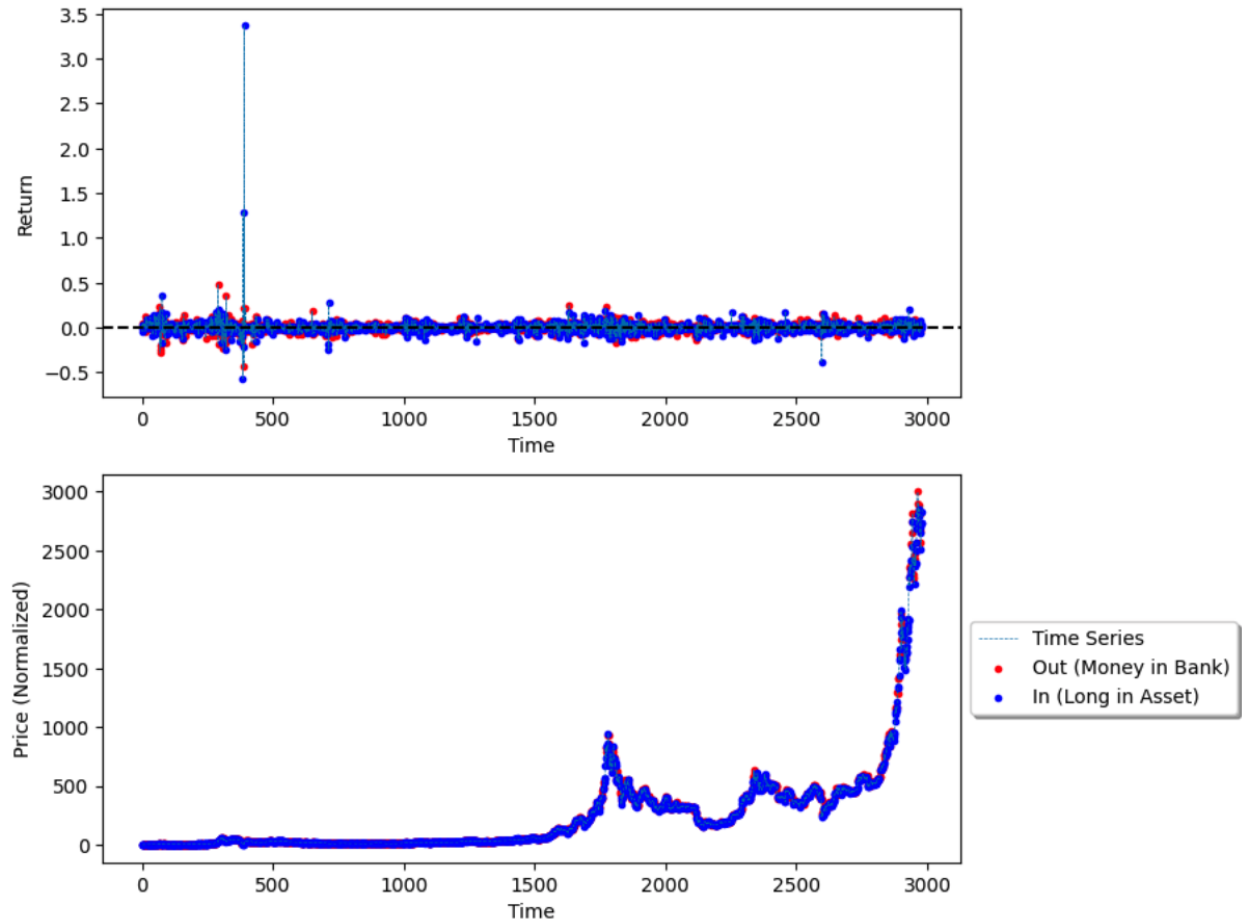
<i>Hyper Parameter</i>	<i>Value</i>	<i>Explanation</i>
<i>Learning_rate</i>	0.03	The extent to which the Q-values are updated during the learning process.
<i>Discount_Rate</i>	1	Determines the present value of future rewards
<i>Episodes</i>	150,000	
<i>Epsilon</i>	0.5	The exploration-exploitation trade-off
<i>End_Epsilon</i>	0.00	The lowest value of epsilon the agent will reach.
<i>Start_Epsilon_Decay</i>	0.00	Zero means that it starts in episode 0 (immediately)
<i>End_Epsilon_Decay</i>	Episodes // 2	After this many episodes we will not decrease the epsilon anymore.

Table 4: Other Q-Learning settings and parameters

<i>Other settings</i>	<i>Training/Test Split</i>	<i>Epochs</i>	<i>Timesteps</i>	<i>Average Run time</i>	<i>Early Stopping Callback</i>
	80/20	-	-	5hrs	-

## Training #1

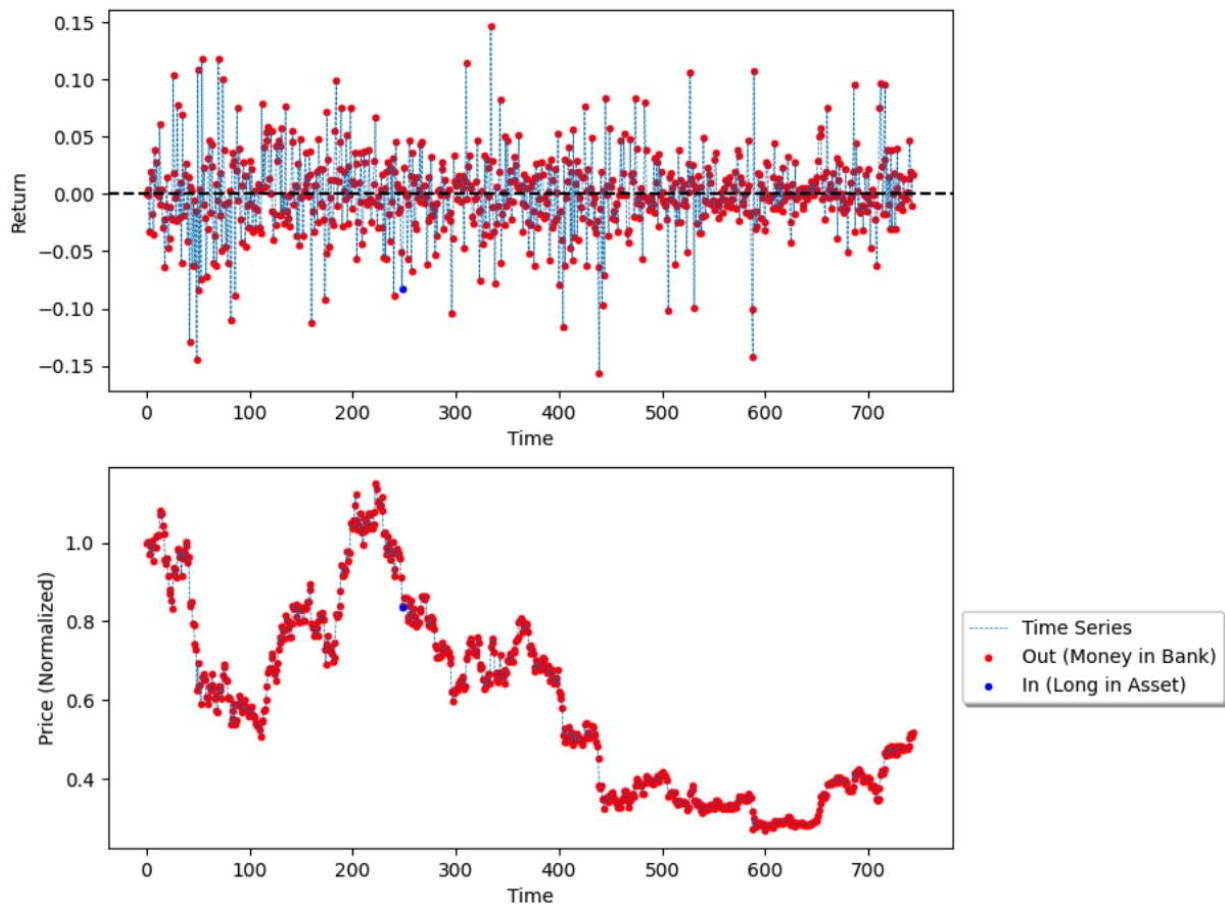
Figure 9: Graphical illustration of trading decisions in training Q-Learning model #1



- The final profit of the Q-Learned strategy is: 873576347527.28....
- The final profit of the buy-and hold strategy is: 288094.60....

## Test #1

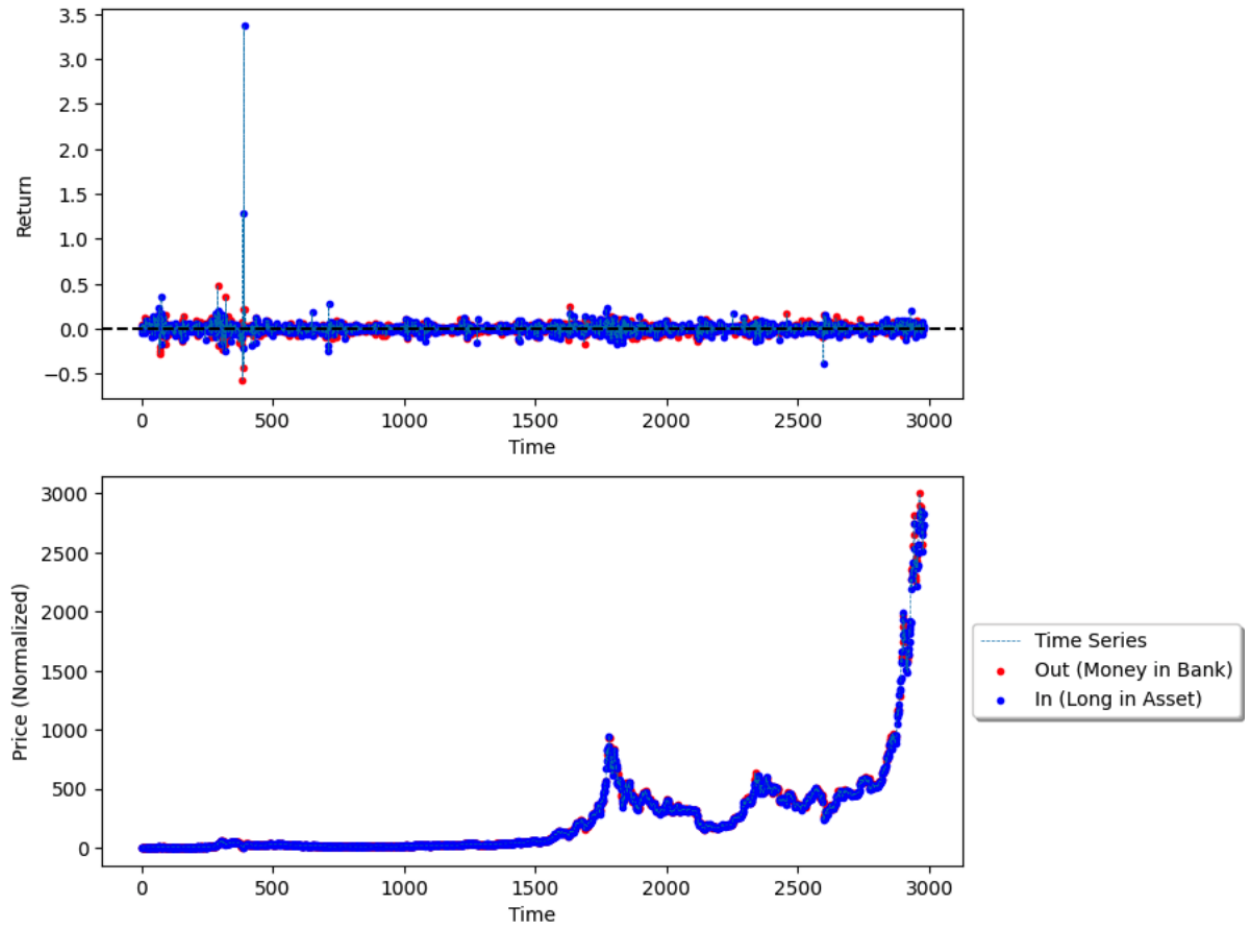
Figure 10: Graphical illustration of trading decisions in testing Q-Learning model #1



- The final profit of the Q-Learned strategy is: 100.43... (+0,43%)
- The final profit of the buy-and hold strategy is: 52.41.... (-47,59%)

## Training #2

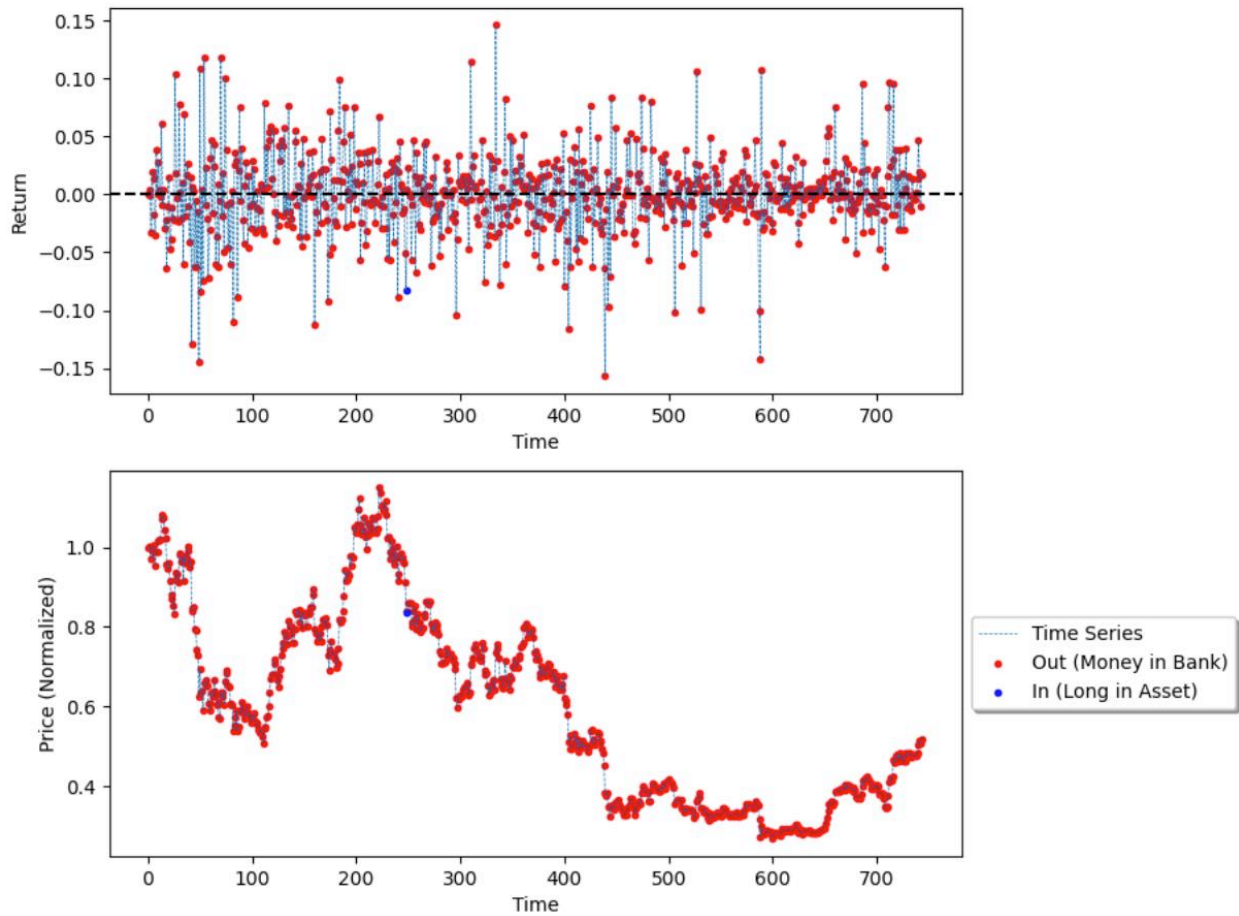
Figure 11: Graphical illustration of trading decisions in training Q-Learning model #2



- The final profit of the Q-Learned strategy is: 461299986811.80....
- The final profit of the buy-and hold strategy is: 288094.60....

## Test #2

Figure 12: Graphical illustration of trading decisions in testing Q-Learning model #2



- The final profit of the Q-Learned strategy is: 100.43... (+0,43%)
- The final profit of the buy-and hold strategy is: 52.41.... (-47,59%)

## Training #3

The final profit of the Q-Learned strategy is: 7669037549.67....

The final profit of the buy-and hold strategy is: 288094.60....

## Test#3

The final profit of the Q-Learned strategy is: 100.43... (+0,43%)

The final profit of the buy-and hold strategy is: 52.41.... (-47,59%)



As can be seen from the results, the Q-learning models has very similar results to the DQN-models. Despite the Q-Learning models also technically outperforming the buy-and hold strategy, it appears this is more due to chance rather than the model making meaningful decisions. We can also note that all Q-learning models converge to the same decisions in the test set, but not in the training set. The test data is largely characterized by a down trend, and it is hard to say what would happen if the test data instead was a strong up trend or a neutral trend. Summarized results for the DQN and Q-Learning models can be seen in table 5.

Table 5: Summary of the test results of the DQN and Q-Learning models

Summary	DQN			Q-Learning			Buy-and Hold
Model No	Model #1	Model #2	Model #3	Model #1	Model #2	Model #3	-
Profit	+14,61%	+3,6%	+0,65%	+0,43%	+0,43%	+0,43%	-47,59%

### 7.3 Final Remarks

The major takeaway from results of the models is that more time and computational power is needed. Furthermore, due to time constraints we we're unable to test the algorithms in ideal circumstances. For future research, these findings suggest several recommendations. For starters, one should experiment with more sophisticated environments where the models are allowed to trade with continuous decisions. Furthermore, the combination of computational power and time appears to be crucial. This allows for the models to be fed more input variables, which can result in better decision making. We also recommend experimenting with different reward functions, such as the sharpe or sortino ratio. It also important to remember that factors such as short positions, transaction costs, and slippage can largely affect the results. In these experiments, it may appear that the models are strictly outperforming the buy-and hold strategy, which they technically are. However, the way the models behave suggests that this is likely not due to some sort of rigorous trading strategy, but rather by chance. DQN Model 2 is the one model which appears to

demonstrate hints at rational trading behavior. The model manages to end up in a 3,6% profit where the market otherwise saw negative returns of 47,59%. Finally, it is also wise to experiment with the various hyperparameters, although this should be far less prioritized compared to constructing a solid reward function and environment.

## 8 Conclusion

This thesis has covered the topic of using reinforcement learning to trade bitcoin on a daily timeframe. We have explored and investigated the effectiveness of two popular algorithms, namely Q-learning and DQN. Our experiments demonstrate both the difficulty and the potential applicability of reinforcement learning algorithms in financial trading. We can conclude that data preprocessing, the construction of the reward function and the environment, as well as the combination of computational power and time are crucial. We have shown that while the algorithms have potential, they can also be prone to irrational decision making. The findings primarily suggest that there is more to learn about the algorithms and their use case in trading. While all models of both algorithms technically performed better than a simple buy-and hold strategy, we hypothesize that this is mostly due to chance. Nonetheless, some findings indicate hints at rational trading behavior, but to draw any meaningful conclusions we require more time, computational power, and resources. Overall, the thesis has contributed to the growing body of research on using machine learning for financial trading, and hopefully it will inspire further research and experimentation in the domain.

## 9 References

- Baldrige, R. (2022, August 5). What Is the Efficient Market Hypothesis? *Forbes*. Retrieved from <https://www.forbes.com/advisor/investing/efficient-market-hypothesis/>
- Barber, B. M., Lee, Y.-T., Liu, Y.-J., & Odean, T. (2014). Do Day Traders Rationally Learn About Their Ability? *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.2535636>
- Browne, R. (2022, April 28). *Central African Republic becomes second country to adopt bitcoin as legal tender*. CNBC; CNBC. <https://www.cnbc.com/2022/04/28/central-african-republic-adopts-bitcoin-as-legal-tender.html>
- Chainalysis Team. (2022, September 14). *2022 Global Cryptocurrency Adoption Index - Chainalysis*. Chainalysis. <https://blog.chainalysis.com/reports/2022-global-crypto-adoption-index/>
- Chen, J. (2023). Analysis of Bitcoin Price Prediction Using Machine Learning. *Journal of Risk and Financial Management*, 16(1), 51. <https://doi.org/10.3390/jrfm16010051>
- Erfanian, S., Zhou, Y., Razzaq, A., Abbas, A., Safeer, A. A., & Li, T. (2022). Predicting Bitcoin (BTC) Price in the Context of Economic Theories: A Machine Learning Approach. *Entropy*, 24(10), 1487. <https://doi.org/10.3390/e24101487>
- Gabriele De Luca. (2020, August 9). *What is a Policy in Reinforcement Learning?* | *Baeldung on Computer Science*. Baeldung on Computer Science. <https://www.baeldung.com/cs/ml-policy-reinforcement-learning>

Huang, J.-Z., Huang, W., & Ni, J. (2019). Predicting bitcoin returns using high-dimensional technical indicators. *The Journal of Finance and Data Science*, 5(3), 140–155.

<https://doi.org/10.1016/j.jfds.2018.10.001>

Jiang, X. (2020). Bitcoin Price Prediction Based on Deep Learning Methods. *Journal of Mathematical Finance*, 10(01), 132–139. <https://doi.org/10.4236/jmf.2020.101009>

John H. Cochrane. Eugene F. Fama, Efficient Markets, and the Nobel Prize. (2014). Retrieved March 10, 2023, from The University of Chicago Booth School of Business website:

<https://www.chicagobooth.edu/review/eugene-fama-efficient-markets-and-the-nobel-prize#:~:text=Fama%20defined%20a%20market%20to,and%20low%20costs%20of%20information.>

Ken. (n.d.). *Recitation 13: Reinforcement Learning*.

[https://deeplearning.cs.cmu.edu/S20/document/recitation/Recitation%2013\\_%20Reinforcement%20Learning.pdf](https://deeplearning.cs.cmu.edu/S20/document/recitation/Recitation%2013_%20Reinforcement%20Learning.pdf)

Liu, F., Li, Y., Li, B., Li, J., & Xie, H. (2021). Bitcoin transaction strategy construction based on deep reinforcement learning. *Applied Soft Computing*, 113, 107952.

<https://doi.org/10.1016/j.asoc.2021.107952>

Li, Y., Zheng, W., & Zheng, Z. (2019). Deep Robust Reinforcement Learning for Practical Algorithmic Trading. *IEEE Access*, 7, 108014–108022.

<https://doi.org/10.1109/access.2019.2932789>

Lucarelli, G., & Matteo Borrotti. (2019, May). A Deep Reinforcement Learning Approach for Automated Cryptocurrency Trading. Retrieved March 8, 2023, from ResearchGate website:[https://www.researchgate.net/publication/333107392\\_A\\_Deep\\_Reinforcement\\_Learning\\_Approach\\_for\\_Automated\\_Cryptocurrency\\_Trading](https://www.researchgate.net/publication/333107392_A_Deep_Reinforcement_Learning_Approach_for_Automated_Cryptocurrency_Trading)

MLK. (2021, March 31). Machinelearningknowledge.ai.  
<https://machinelearningknowledge.ai/beginners-guide-to-what-is-policy-in-reinforcement-learning/#:~:text=What%20is%20Policy%20in%20Reinforcement%20Learning%20In%20a,take%20at%20a%20particular%20state%20in%20the%20environment.>

Munim, Z. H., Shakil, M. H., & Alon, I. (2019). Next-Day Bitcoin Price Forecast. *Journal of Risk and Financial Management*, 12(2), 103. <https://doi.org/10.3390/jrfm12020103>

Nevasalmi, L. (2020). Forecasting multinomial stock returns using machine learning methods. *The Journal of Finance and Data Science*, 6, 86–106.  
<https://doi.org/10.1016/j.jfds.2020.09.001>

Park, M. (2018, February 2). Random Walk Theory. Retrieved March 12, 2023, from Corporate Finance Institute website: <https://corporatefinanceinstitute.com/resources/capital-markets/what-is-the-random-walk-theory/>

Parsa Heidary Moghadam. (2019, July 21). *Deep Reinforcement learning: DQN, Double DQN, Dueling DQN, Noisy DQN and DQN with Prioritized Experience Replay*. Medium; Medium. [https://medium.com/@parsa\\_h\\_m/deep-reinforcement-learning-dqn-double-dqn-dueling-dqn-noisy-dqn-and-dqn-with-prioritized-551f621a9823#:~:text=The%20only%20difference%20between%20Q,is%20a%20deep%20neural%20network.](https://medium.com/@parsa_h_m/deep-reinforcement-learning-dqn-double-dqn-dueling-dqn-noisy-dqn-and-dqn-with-prioritized-551f621a9823#:~:text=The%20only%20difference%20between%20Q,is%20a%20deep%20neural%20network.)

Sattarov, O., Muminov, A., Lee, C. W., Kang, H. K., Oh, R., Ahn, J., ... Jeon, H. S. (2020). Recommending Cryptocurrency Trading Points with Deep Reinforcement Learning Approach. *Applied Sciences*, 10(4), 1506. <https://doi.org/10.3390/app10041506>

Vijay Kanade. (2022, December 20). *Markov Decision Process Definition, Working, and Examples*. Spiceworks; Spiceworks. <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-markov-decision-process/>

Williams, J. D., & Young, S. (2007). Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2), 393–422. <https://doi.org/10.1016/j.csl.2006.06.008>

Wood, J. (2022, November 10). *What Bitcoin's Inflation Hedge Narrative Needs: More Time*. @Coindesk; CoinDesk. <https://www.coindesk.com/business/2022/11/10/what-bitcoins-inflation-hedge-narrative-needs-more-time/>

Zhang, Z., Zohren, S., & Roberts, S. (2020). Deep Reinforcement Learning for Trading. *The Journal of Financial Data Science*, 2(2), 25–40. <https://doi.org/10.3905/jfds.2020.1.030>

## Appendix #1 – Experiments with Different Time Series and RL Algorithms

### Direct Optimization of Optimal Trading with Deterministic Series

When time series are deterministic and comparably small, we can find the optimal solution by means of linear optimization:

Let  $x_t^S$  and  $y_t^S$  represent the number of stocks purchased or sold in point of time  $t$ . The variable  $x_t^B$  is the amount invested in bonds (bank), respectively.  $P_t^S$  is the stock price in point of time  $t$ . Let  $r$  be the constant interest rate, and  $\tau$  are transactions costs in % when selling or buying stocks.  $B_0$  denotes the initial budget available at point in time  $t = 0$

for  $t = 0$ :

$$\text{Budget constraint:} \quad P_0^S \cdot x_0^S + P_0^S \cdot x_0^S \cdot \tau + x_0^B = B_0$$

$$\text{Holding of asset:} \quad z_0^S = x_0^S$$

$$\text{Non-negativity:} \quad x_0^S \geq 0, x_0^B \geq 0$$

$$z_0^S \geq 0 \text{ (redundant)}$$

We need to follow with the volume of stocks:

for all  $0 < t < T$ :

$$\text{Budget constraint:} \quad P_t^S \cdot y_t^S \cdot (1 - \tau) - P_t^S \cdot x_t^S \cdot (1 + \tau) + (1 + r) \cdot x_{t-1}^B - x_t^B = 0$$

$$\text{Cannot sell more than holding:} \quad y_t^S \leq z_{t-1}^S$$

$$\text{Holding of asset:} \quad z_t^S = z_{t-1}^S - y_t^S + x_t^S$$

$$\text{Non-negativity:} \quad x_t^S \geq 0, x_t^B \geq 0, y_t^S \geq 0$$

$$z_t^S \geq 0 \text{ (redundant)}$$

In point of time  $t = T$ , we assume that we hold the position (no more trading) from the previous point in time, and we calculate its value. The wealth in the end of the trading-horizon is:

$$w_T = P_T^S \cdot z_{T-1}^S + (1 + r) \cdot x_{T-1}^B$$

This wealth is supposed to be maximized.



The solution to this problem can serve as a benchmark against the solutions of RL.

It is also possible to formulate a corresponding optimization problem for a stochastic environment. When applying the framework of stochastic optimization with deterministic equivalent in form of linear programs, this would require the construction of scenario trees. We will omit this approach here.

### RL-Experiments with Deterministic Series

We have run several experiments with artificial time series, to have some kind of guarantee that RL Methods will learn to make good trading decisions when there exists a possibility of learning. For this reason, we have tested deterministic and simple stochastic time series. In what follows, we will present the results.

#### Prespecified Return List

In this experiment, we have used the following list of prices, which was repeated three times:

[0.10, 0.05, 0.08, -0.03, -0.04, 0.00, 0.00, 0.07, -0.02, ...]

Except for the return of 0.00 this time series does not really contain any ambiguous states.

In this list we have indicated, when to stay in (green) and when to stay out (red).

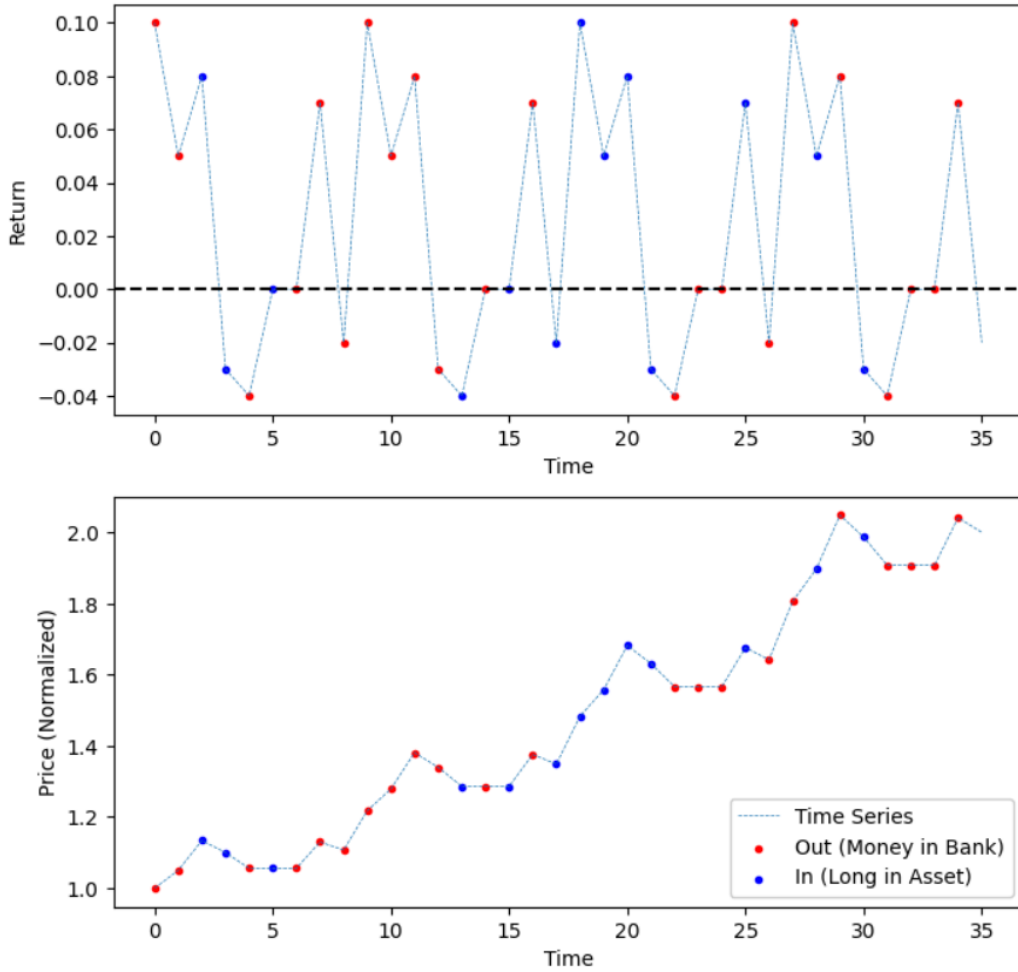
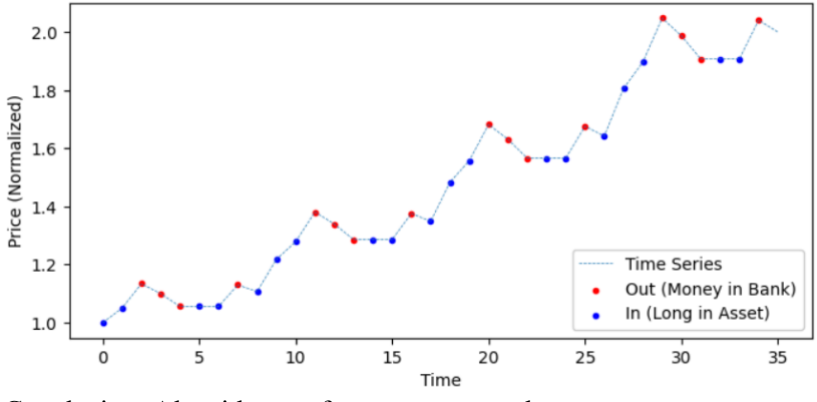
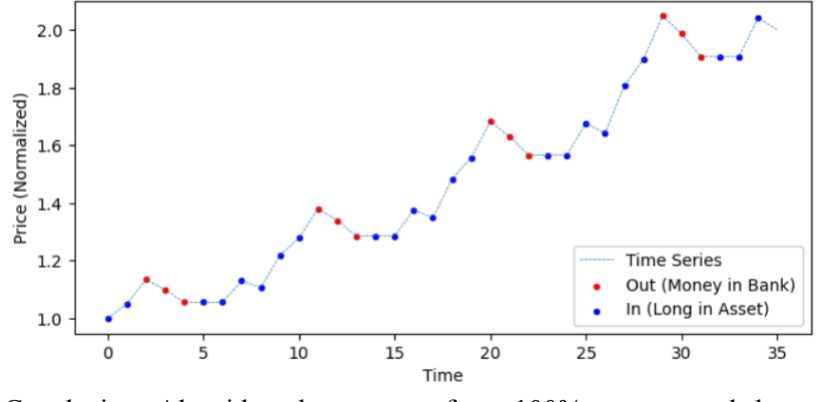


Figure 13: Plot of Returns and Prices for Deterministic Return List with Random In and Out Decisions

If we use the price  $r_t$  as only information, RL algorithms should not unambiguously learn the right decisions:

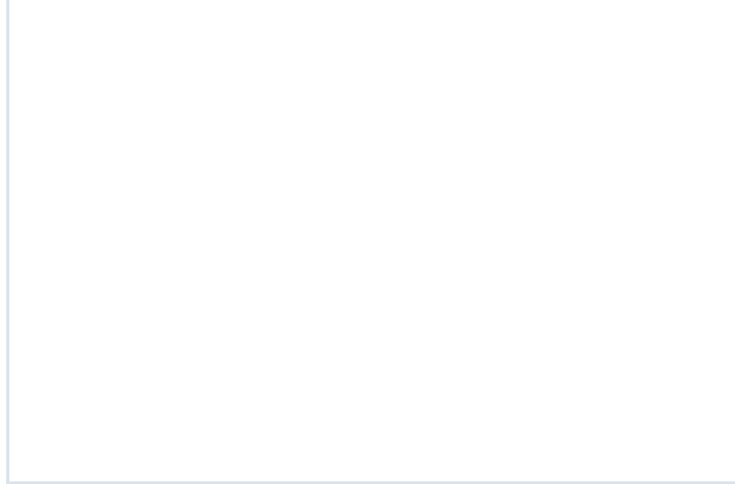
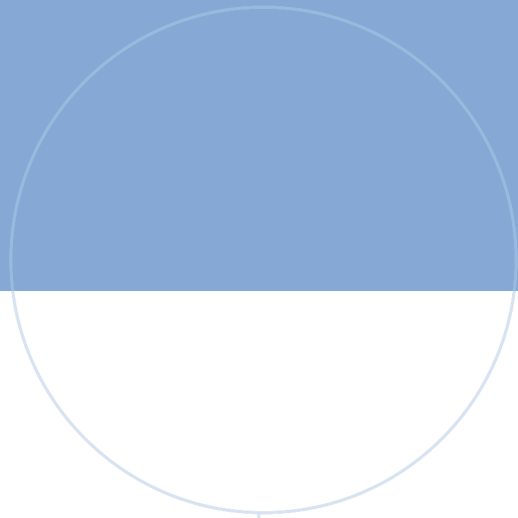
Return	Next Return	Reward (for one repetition)	Expected Decision
0.10	0.05	1 x positive reward	Stay in
0.05	0.08	2 x positive reward	Stay in
0.08	-0.03	1 x pos., 1 x neg. (but pos. reward is larger)	Stay Out
-0.03	-0.04	1 x pos., 1 x neg. of same absolute size	Stay Out
-0.04	0.00	2 x positive reward	Indifferent
0.00	0.00	1 x negative	Indifferent
0.00	0.07		Stay in
0.07	0.02		Stay in
0.02	0.10	1 x negative	Stay in

At price 12: 2 x positive reward when it comes to downward trends or negative rewards.

<p><b>Q-Learning:</b></p> <ul style="list-style-type: none"> <li>• Bin-size = 9</li> <li>• Learning_Rate = 0.03</li> <li>• Discount_Rate = 1</li> <li>• Episodes = 5,000</li> <li>• Epsilon = 0.5</li> <li>• End_Epsilon = 0.00</li> </ul>	 <p>Conclusion: Algorithm performs as expected.</p>
<p><b>DQN:</b></p> <ul style="list-style-type: none"> <li>• gamma = 1,</li> <li>• batch_size = 18</li> <li>• total_timesteps = 1,440,000</li> <li>• Time: 1,554 s</li> </ul>	 <p>Conclusion: Algorithm does not perform 100% as expected, but quite acceptable.</p>
<p><b>PPO:</b></p>	<p><b>No training results achieved at all.</b></p>

If we use both the price  $p_t$  and the lagged price  $p_{t-1}$  as information, the RL algorithms should be perfectly able to conclude the correct decisions:





 **NTNU**

Norwegian University of  
Science and Technology