Automatically trading small market capitalization cryptocurrencies using reinforcement learning

Bachelor Thesis – 7,5 ECTS
15/04/2021

Under supervision of:
Natasha Alechina &
Dominik Klein

Stephan Akkerman
Student number: 6397514

Bachelor Artificial Intelligence
Faculty of Humanities
Utrecht University
Netherlands

Table of contents

1. Introduction

Since the beginning of trading, people have sought out ways to "beat the market"; a phrase meaning that you have earned more with investing than the performance of the S&P 500, the most popular benchmark for trading US-based stocks. Traders have searched for all kinds of ways and techniques to do so, yet only a few succeed. However, the rise of artificial intelligence has brought change to this. With the amount of data available nowadays and the increasing computing power, it has become a lot easier to train a reinforcement learning agent for automatic trading. Since trading stocks automatically is a topic that has been well established over the years, there are not many inefficiencies left in the market for newcomers to explore and exploit. The exploits that exist have all been claimed by big corporations. With cryptocurrency this is a different case, as it is decentralized, which gives big corporations considerably less power [1]. Besides that, automatic trading focused on cryptocurrencies is an area that is not well studied yet. Combining that with reinforcement learning, which is also a research area that is upcoming, makes it so that there is not much research on this topic that has been published. The research that has been published, focuses solely on the cryptocurrencies with large market capitalization, such as Bitcoin and Ethereum [2]. The type of cryptocurrencies researched in this paper is something else, as it focuses on cryptocurrencies with a small market capitalization. In section 1.2 it is explained why market capitalization is an essential aspect to take into consideration. This research tries to give a clear comparison of the performance of a reinforcement learned strategy applied to small market capitalization cryptocurrencies and the performance of this strategy on large market capitalization cryptocurrencies.

1.1. Reinforcement Learning

There are numerous ways available for applying artificial intelligence [3]. This paper focuses on reinforcement learning, a machine learning technique. The way reinforcement learning works can be compared to the way humans learn. In the beginning it will attempt different actions, these actions will be applied on the environment which returns a state and reward. It consists of five components: the agent, the environment, the action of the agent on the environment, the resulting reward, and state of that action [4]. The agent is managed by an algorithm, it will receive a state from the environment and based on that, the agent will take the action it considers the best. After doing that, the agent will receive feedback from the environment, the reward. If the action is something that we would like to see, then it returns a positive reward. If the action results in a state that is not desirable, the agent receives a negative reward. Gradually the agent will learn what actions are best to take, depending on the rewards it receives. The downside of this is that it takes time before resulting in the best actions, since the agent learns by trial-and-error. Finally, the sequence of actions that result in the best long-term reward are known, all through the interaction with the environment and the results of that.
The reason for choosing reinforcement learning for this study, is that it is an area that gained a lot of success and popularity over the past few years, for instance with AlphaGo Zero, that utilized this technique [5]. Reinforcement learning performs well for games, because the reward of a certain action is known, and it is possible to calculate the states that follow. Unfortunately, this is not conceivable when trading cryptocurrency, since no one knows how the market will look like

next, which makes this a trickier problem than any environment where games are involved in. Another reason for choosing reinforcement learning, is that it provides a lot of freedom. There are multiple different algorithms available for the reinforcement learning agent, each having their own strengths and weaknesses. Besides that, the environment, rewards, and actions are all configurable. In later sections this is explained in depth. This paper shows different reinforcement learning algorithms, so that a comparison can be made. The outcomes of these algorithms are compared to each other. The best performing algorithm is compared to common passive trading strategies, such as a straightforward buy and hold strategy.

1.2. Market Capitalization

A statement that was made in the beginning of this section, is that currently there are no published studies on automatically trading small market capitalization cryptocurrency. Market capitalization is an important aspect to consider when trading. In cryptocurrency it measures the relative size of a cryptocurrency coin, for instance Bitcoin. To calculate it, the current price and total number of coins in circulation needs to be multiplied. You can view market capitalization as how much a coin is worth, determined by the market. There are three classes of capitalization. Large capitalization cryptocurrencies have a market capitalization of more than ten billion dollars. The large market capitalization coins are well established and have been known for a long time. These assets do not fluctuate as much, which makes them a safer investment, especially as a long-term investment. The term mid capitalization is used for a market capitalization between two and ten billion dollars. These coins are more volatile, but that also means that more profit can be realized compared to large-capitalization assets. The last class is small market capitalization, these are the riskiest investments of all, since this type of asset is the most volatile.

1.3. Research Question

This paper researches the profitability of a reinforcement learned strategy applied to small market capitalization cryptocurrencies, to lay the foundation for potentially more elaborate further research. To do so, the following research question is relevant: "**To what extent is a reinforcement learned strategy more profitable than common passive strategies applied to cryptocurrencies with a small market capitalization?**". It is important to get a clear distinction between the feasibility of using reinforcement learning for trading small market capitalization and large market capitalization cryptocurrencies. As stated before, this is something that has not been researched before. Therefore, it is not clear what to expect prior to the results. Since large market capitalization coins have been in the industry for a longer period, there is more historical data of it available. That means that the reinforcement learning agent can be trained better. Large market capitalization coins are less volatile compared to small market capitalization coins, which means that the agent can make safer investments when trading it. However, trading small capitalization coins could result in more profit, because the price changes faster during the same time. If there is enough data on small capitalization cryptocurrency coins for the reinforcement learning agent to properly learn from, then it should be able to make more profit trading that instead of the large market capitalization coins.

To answer the research question in the clearest way possible, this research consists of explaining how the reinforcement learning is set up and used. The next section explains the design of the research. That consist of explaining the foundation that is needed for properly conducting this research. After that the method of research is explained, stating how the research was set up. Section four shows the results of the experiment. It shows the comparison of the different reinforcement learning algorithms, and the comparison of small market capitalization and large market capitalization cryptocurrencies. The fifth section consists of the discussion, explaining what could be improved in further research on this topic. At last is the conclusion, expressing the concluding thoughts on the presented research and its results.

2. Design

This section describes the design of reinforcement learning and everything else necessary for it to operate properly. There are a lot of options for frameworks available, depending on the programming language one would like to use. For practical reinforcement learning, the TensorTrade library can be used [6]. TensorTrade is a python framework that focusses on trading cryptocurrency, using TensorFlow-like methods. This supports the creation of environments. These trading environments are a lot different than the default environments found in the OpenAI Gym [7]. The TensorTrade environments are specifically made for trading cryptocurrency and are modular as well. This makes it an interesting library for researching reinforcement learning, as there are many parameters and options to consider.

2.1. Trading Environment

The trading environment simulates the way a human trader would perceive the market. This is a single agent environment, since there is only one trader involved. A trading environment consists of multiple components, consisting of the action scheme, reward scheme, observer, stopper, informer, and renderer. The action scheme defines the action space of the agent in the environment and applies it to the environment. In the case of trading, there are only a few actions to take, which means that these are discrete actions. The reward scheme calculates the corresponding reward for a certain action in our trading environment. If the action makes profit, the reward will be positive. If the action results in a loss, the reward will be negative. The observer specifies what the agent can perceive, this should at least be the price of the asset it is trading. This component takes care of lookahead bias, since that is a big issue for training and evaluating trading strategies. The components that need to be specified in the observer are what data it should be receiving and how far it should consider previous observations. Using this information, it will determine the current best action to take. The stopper component checks if the environment can stop after each step in the environment. This is an important hyperparameter for creating a good trading strategy, because if the value is too high, it will learn to do only the safest of trades. However, if the value is too low, the agent will execute risky trades that might result in a loss. The informer provides contextual information, such as the current worth of the portfolio, commission of the broker and current net worth. The renderer is merely used for visualizing the results of training and testing.

2.2. Agents

After setting up the trading environment, there is another significant aspect to consider. That is the reinforcement learning agent in the environment. There are enough libraries that provide these kinds of agents. For this paper, a library called Ray is utilized [8]. Other libraries have been tested, such as TensorFlow Agents, but no other library provides as much freedom in customization as Ray does. Another reason for choosing Ray, is that it offers multiple additions. For instance, search algorithms, schedulers, and custom metrics. These additions make it easier for the user to train and optimize the reinforcement learning agent.

2.3. Data

The historical data used for training and evaluating, was provided using the exchange Binance. For getting enough data, and the correct timeframe, a custom script had to be prepared. At the moment of writing, Binance is the most popular cryptocurrency exchange available [9]. Besides the exchange itself, the data that it provides is also crucial. The reason being, if the data does not consist of the date, open price, highest price, lowest price, close price, and volume during that timeframe, then the environment will not accept it. That way, the data can be used in more sophisticated methods, since it is important for decision-making and rendering. One of these sophisticated methods it can be used for, is for converting the data in technical analysis output. Technical analysis helps the agent deciding which decision it should make. There are numerous different types of technical analysis indicators. To make sure the agent does not get excessive amounts of unnecessary data, the overlapping indicators are filtered. This is done by comparing the indicators of the same type, and only using those that have a mean correlation of less than 0.3. To visualize this process a heatmap is generated, so it is straightforward to see which indicators correlate the most. In the heatmap below the absolute correlation is plotted, the closer it gets to 1.0, shown as yellow, the less important it is to include in the dataset. It is noticeable that there are some bright yellow squares. The explanation for this is that there are groups of indicators that represent the same concept. For instance, the Ichimoku indicator consists of 6 parts, it is not strange these indicators have high correlation between them.

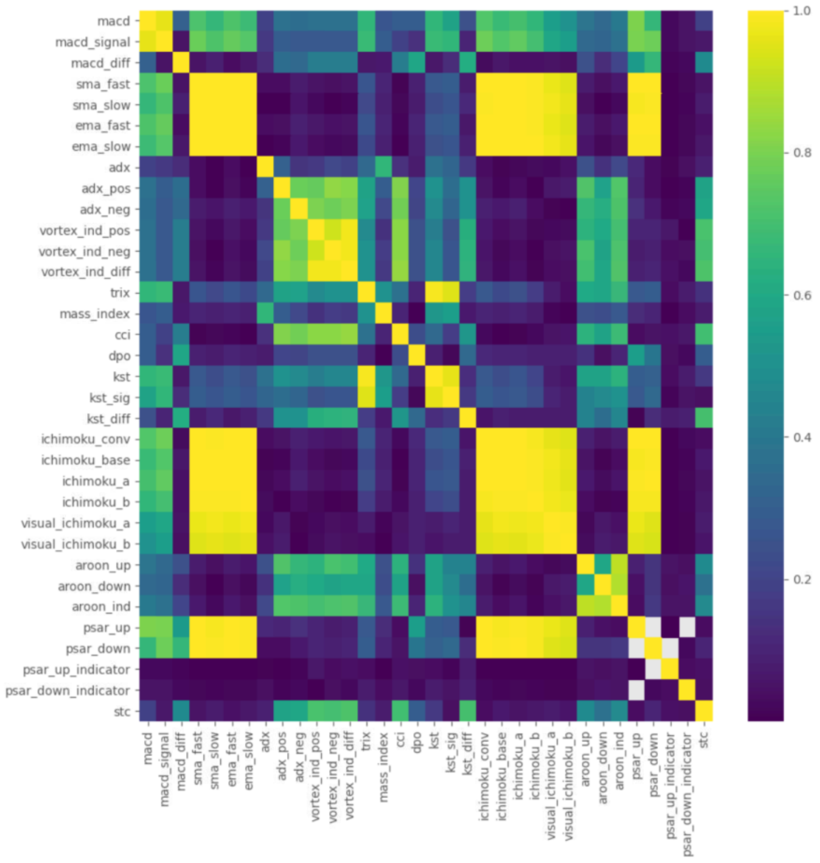

Figure 1: A heatmap made of all the trend indicators available in the TA library. A value close to 1.0, depicted as bright yellow, means maximal absolute correlation. A value close to 0, depicted as dark purple, means minimal absolute correlation.

Besides the heatmap, a table of indicators is generated. This table shows the indicators that have a mean correlation of less than 0.3. The result is depicted in the tables below, each table representing their class of technical analysis indicators. The labels of the indicators are their abbreviations, used by the library for the technical analysis indicators [16]. Initially, there are ten volume indicators, after selection this amount decreased by 40%. The amount of momentum indicators decreased by 53%, the volatility indicators by 62%, and the trend indicators by 65%. So around 50% of indicators were dropped. The result is that there is less unnecessary data that is being used, compared to using all the indicators. This makes training the reinforcement learning go quicker.

Volume Indicator	Mean Correlation	Momentum Indicator	Mean Correlation	Volatily Indicator	Mean Correlation	Trend Indicator	Mean Correlation
adi	0.214288	**stoch_rsi**	0.268178	**bbw**	0.282020	**macd_diff**	0.293040
cmf	0.292900	**stoch_rsi_k**	0.276770	**bbhi**	0.198300	**adx**	0.131943
fi	0.263218	**stoch_rsi_d**	0.267476	**bbli**	0.146744	**mass_index**	0.182142
em	0.135506	**uo**	0.286961	**kcw**	0.273634	**dpo**	0.175894
vpt	0.162266	**ppo**	0.275453	**kchi**	0.225085	**kst_sig**	0.283181
nvi	0.264872	**ppo_signal**	0.260765	**kcli**	0.186678	**kst_diff**	0.273635
		ppo_hist	0.177761	**dcw**	0.283415	**aroon_up**	0.284469
				ui	0.217389	**aroon_down**	0.234316
						aroon_ind	0.299218
						psar_up	0.059687
						psar_down	0.053389
						stc	0.299926

Table 1: A compilation of tables. Each table shows the indicators in their class and the corresponding mean correlation. Only the indicators that have a mean correlation lower than 0.3 are shown.

2.4. Feature Scaling

One last step needs to be made, before using the data. Everything needs to be scaled accordingly. The reason for this is that otherwise the agent will have a hard time during the testing phase. Since the prices of cryptocurrencies change a lot, the agent will often encounter situations which it has not been trained on. For instance, if the agent is trained using the historical closing prices, it will not know what to do if it is tested on new all-time highs. If the data is scaled, the described problem will not occur. The method of scaling that is used for this paper is called the logarithmic returns, which is a technique often used for algorithmic trading [22]. The way it operates is straightforward, the current value is divided by the previous value, then the logarithmic value of this outcome is calculated. This method can be used on a lot of the data, for instance on the opening price, highest price, lowest price, closing price, and volume, mentioned in the previous section. However, some indicators have values of zero or below, unfortunately log return scaling could not be applied to those indicators. Instead of that, percentage change has been used. This method simply calculates the change in percentages, using the current and previous value. Several technical analysis features are automatically scaled, for instance the bbhi volatility indicator. This indicator returns 1 if the closing price is greater than upper Bollinger band, and otherwise it will return 0 [18]. The indicators named vpt, em, and psar_down have not been used. As there were problems using these indicators as features.

2.5. Algorithms

In the case of a trading environment, there are some matters to consider when picking an algorithm. The algorithm needs to support discrete actions and single-agent environments. Ray offers multiple of these types of algorithms [19]. One of the algorithms utilized in the first experiment, is called PPO. It stands for Proximal Policy Optimization and was developed in 2017, by OpenAI [10]. A policy is the strategy used by the agent. In PPO this is updated in small batches, based on the experience of the agent in the training environment. Important is that the algorithm is not allowed to change too much, compared to the old policy. This results in lower variance but leads to a slightly higher bias. Overall, this trade-off is worth it, as it lowers the chance the agent will make a big mistake somewhere, which would result in a bad policy. Besides PPO, there are other algorithms that have been studied for this paper. One of those algorithms is named Importance Weighted Actor Learner Architecture, also known as IMPALA [14]. The workings of IMPALA resemble PPO in some ways, except that it is an off-policy algorithm. Another valuable algorithm is A2C, which stands for Advantage Actor Critic. A2C, just like PPO, has been developed by OpenAI [23]. This algorithm utilizes an actor-critic method. The actor is the policy that is being optimized and the critic represents the value function that is being learned. Besides these three algorithms, other algorithms were used as well, such as APPO, A3C, and DQN. DQN is the most noteworthy, since APPO and A3C are simply asynchronous variants of PPO and A2C, respectively. DQN stands for Deep Q-Network, it uses Q-learning in combination with neural networks [15]. All the mentioned algorithms were researched for this paper.

2.6. Benchmarking

As was mentioned in the introduction, cryptocurrency does not have a well-known benchmark like stocks do. Therefore, something else is needed to test the performance of the reinforcement learned strategy. There are some strategies that resemble the concept of comparing the performance of the S&P 500 with your own investment. Making a comparison with these common cryptocurrency trading strategies is necessary, because otherwise it is unclear how well performing the reinforcement learned strategy is. The data used for the strategies consists of the latest 900 four-hourly candlesticks of Bitcoin data. A candlestick shows how much the price has changed during a specific timeframe. The color of the candlestick depends on the change in price and the closing price of the candlestick before it. Here four-hour candlesticks are used, so a green candlestick means a price increase in those four hours. Whereas a red candlestick means a decrease in the price. For this section Bitcoin data is used, later in the results section a comparison is made using different assets. There are three strategies that are used for benchmarking. The first one is called "buy and hold". This is the simplest strategy; it consists of buying as much as possible at the earliest timeframe and not selling until the end. It is not a complex strategy, but it has turned out to be profitable in the past.

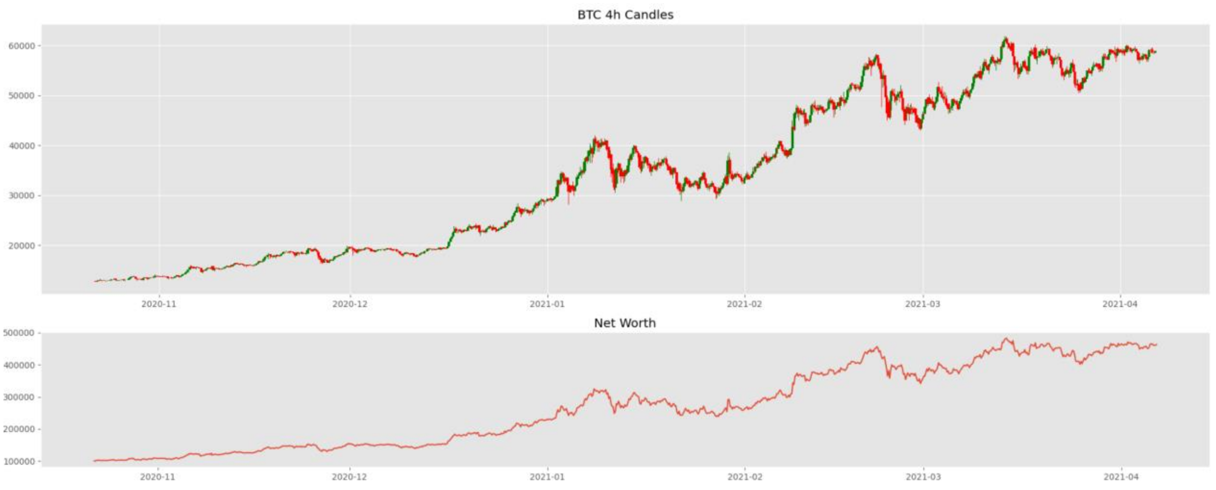

Figure 2: This graph shows the 900 four-hourly Bitcoin data between November 2020 and April 2021. Below is the net worth resulting in applying the buy and hold strategy on this data.

The second strategy is RSI divergence. RSI is an abbreviation for relative strength index. This is a popular indicator used in trading assets and is used for measuring the magnitude of recent price changes. The RSI divergence will sell if the RSI drops, whereas the price of the underlying asset rises and vice versa. In this benchmark, the RSI window is set to 14. This means that it takes the last 14 candlesticks into consideration when calculating the RSI. This is the default value for calculation.

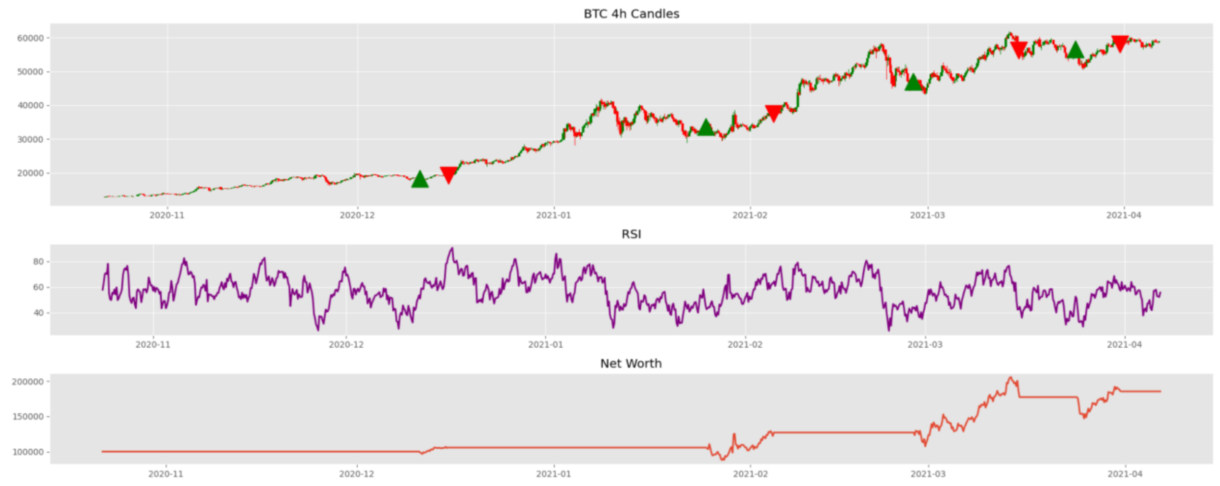

Figure 3: The first graph shows the Bitcoin data between November and April and the actions taken. The green arrow represents the buy action and the red one selling of the asset. Below that is the corresponding RSI. The third plot shows the corresponding net worth when performing these actions.

The last benchmarking strategy that will be used, is called the Simple Moving Average (SMA) crossover. The SMA calculates the average of prices in a certain range. The short-term SMA is calculated by using a smaller window than the window being used for the long-term SMA. For this benchmark, the long-term SMA is set to 100, meaning that it calculated the average of the last 100 price points. The short-term SMA is set to 40. The strategy sells when the short-term SMA crosses above the long-term SMA. The strategy will buy when the short-term SMA crosses below the long-term SMA.

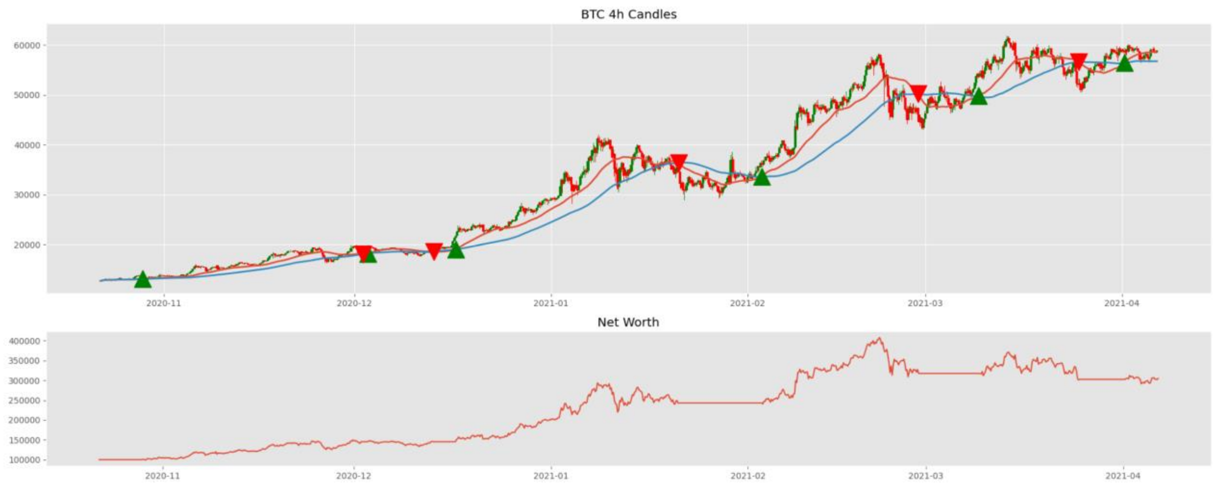

Figure 4: In the first graph the Bitcoin daily data from November until the beginning of April is plotted. The corresponding short moving average and long moving average are plotted in the same graph. The orange line represents the short moving average, and the blue line represents the long moving average. At the cross over the action is plotted. This payoff of this strategy is plotted in the net worth graph below.

3. Method

In this section the process of the actual experiment is described. There are a lot of parameters and settings to configure. The parameters that are not mentioned here are on default values. The settings described in this section function for the researched cryptocurrencies. This does not necessarily imply it would function well for other cryptocurrencies or different types of assets, for instance stocks.

3.1. Training

The data is split into two groups, the training set, and the test set. The training set consists of 80% of the available data, whereas the test set is the other 20%. To give the agent more information, the data is used for technical analysis. That way the agent has more information about the trend, volume, volatility, and momentum of the asset it is about to trade. The agent will receive 100.000 US dollars to trade with. An exchange object needs to be established. This exchange will simulate a real exchange, depending on the parameters that are passed to it. For this research, the Binance exchange is used, because the historical data is based on this exchange. The exchange options that need to be declared to replicate the actual exchange are the commission and minimum trade price. Currently these values on Binance are 0.075% for commission and 10 dollars as the minimum trade price [11]. The next component setting to declare is the type of reward scheme. The reward scheme used for the results is called "SimpleProfit". As the name suggests, this reward scheme is quite simple for its reward calculation. It returns a reward of -1 for not holding a trade, 1 for holding a trade. This ensures that the agent prefers holding a trade. If the agent buys an asset the reward is 2, and if the agent sells an asset, it receives a reward corresponding to the profit earned. There are more profound reward schemes available, unfortunately these did not result in better performance. Another component as important as the reward scheme, is the action scheme. An action scheme called "ManagedRiskOrders" was used for deriving the results. This is the most refined action scheme that is available, as it implements stop losses and take profits. This ensures that the agent cannot lose a lot of money, because of the stop loss. Take profits guarantee that the agent does not stay in the trade for too long, which could lead to losses. Using reinforcement learning, the agent decides which percentage it should use for the stop loss and take profit. The last few things that need to be declared are the lookback window and the maximum allowed loss. The window size is set at 10, meaning that the agent will take the last 10 datapoints into consideration. The window size should not be set to a large value, as this would cause overfitting. The maximum allowed loss is set to 0.95. The maximum allowed loss specifies how much the agent is allowed to lose. If maximum allowed loss is set to one, that means it cannot lose any money during trading. If it is set to zero, that means that it can lose everything. The name of this variable led to confusion for some TensorTrade users. It would make more sense if 1 would mean the agent is allowed to lose 100% of its portfolio, however it is the opposite. With these settings the training environment is created. At last, the value of gamma is set to 0. Lower gamma values will put more weight on short-term gains, whereas higher gamma values will put more weight towards long-term gains. The preferred result of every trader is as much profit, in the shortest amount of time. Setting the gamma to 0 ensured this. All the settings not mentioned are left to their default values. The agent will train for 100 iterations, after this number of iterations the rewards do not change anymore. Throughout the training phase, the agent tries to maximize the reward, which is calculated by the reward scheme. During every training step, the agent makes a quick save, called a checkpoint. At the end of training, the checkpoint that produces the highest reward is chosen. This checkpoint will be used to restore the agent, which will then be used in the next phase.

3.2. Testing

The next phase is testing the agent. To carry out the testing on unseen data, a new environment has to be created. This new environment uses the unseen data; however, it needs to be using the same parameters as the training environment. Then the restored agent, that is based on the training data, will be used in this new testing environment. To calculate how well the testing agent behaves, a few components are initialized. These components are the episode reward, a check to see if the agent is done in the environment, and the agent's observation of the environment. Using the agent's observation, its action can be computed. Next, this action will be performed in the testing environment, which results in a few outputs. The outputs received from the action are the new observation, the rewards, a check to see if all the steps are taken in the testing environment, and extra information. Then the rewards will be added to the current count of rewards. If the check results in the agent being done, these calculations will stop. At last, the testing environment will be passed to the renderer, which will display the results.

4. Results

In this section the results, including graphs, are presented. To make a worthy comparison between small and large market capitalization cryptocurrencies, multiple coins have been used for this section. To represent the large market capitalization cryptocurrencies, Bitcoin has been used. For small market capitalization Basic Attention Token (BAT) and Nano (NANO) have been used. The reason is that at the time of writing, BAT has a market capitalization of 1.5 billion US dollars [16]. NANO has a market capitalization of 788 million dollars [20]. As defined in section 1.2, small market capitalization assets have a capitalization of less than two billion dollars. Therefore, BAT and NANO can be considered as small market capitalization assets.

4.1. Algorithm Comparison

To decide what type of algorithm to use for the next section, multiple algorithms had to be tested. These algorithms are made available by Ray's RLlib library [19]. The requirements for applying these algorithms to TensorTrade, are that they must support discrete actions. An additional requirement set by the equipment used for this experiment, is that the algorithm needs no more than 16 CPU cores. Unfortunately, this meant that the APEX-DQN algorithm could not be used, since it needs at least 32 cores. Testing the algorithms was done using the settings mentioned in the method section. To conclude which algorithm was best to use, a comparison needed to be made. This is done by comparing the total acquired net worth in the testing environment, when trading BAT. Below, a bar graph is shown comparing each algorithm and the acquired net worth. A2C has the best performance, followed by A3C, which is based on A2C. The results between the algorithms do not differ a lot. It is not remarkable that the A2C algorithm performs well, as this is also the case when trading other type of assets [21].

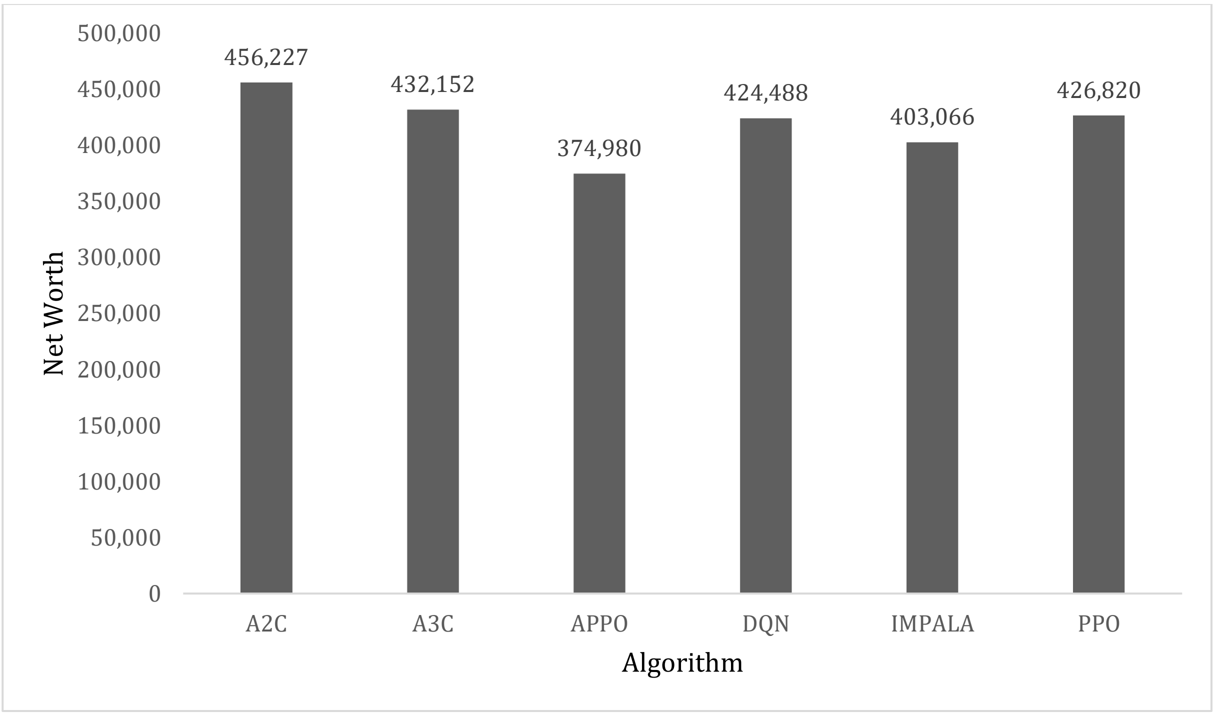

Figure 5: The acquired total net worth of each algorithm, trading BAT.

4.2. Performance of Reinforcement Learning Agent

The previous section concluded that the A2C algorithm is the best algorithm to use, when focusing on improving net worth. Therefore, this algorithm is utilized for presenting the results in this section. For these results, the same settings have been used as mentioned before, using the same timeframes mentioned in section 2.6. In the graphs below, the net worth of the reinforcement learning agent is compared to the net worth of the benchmark strategies.

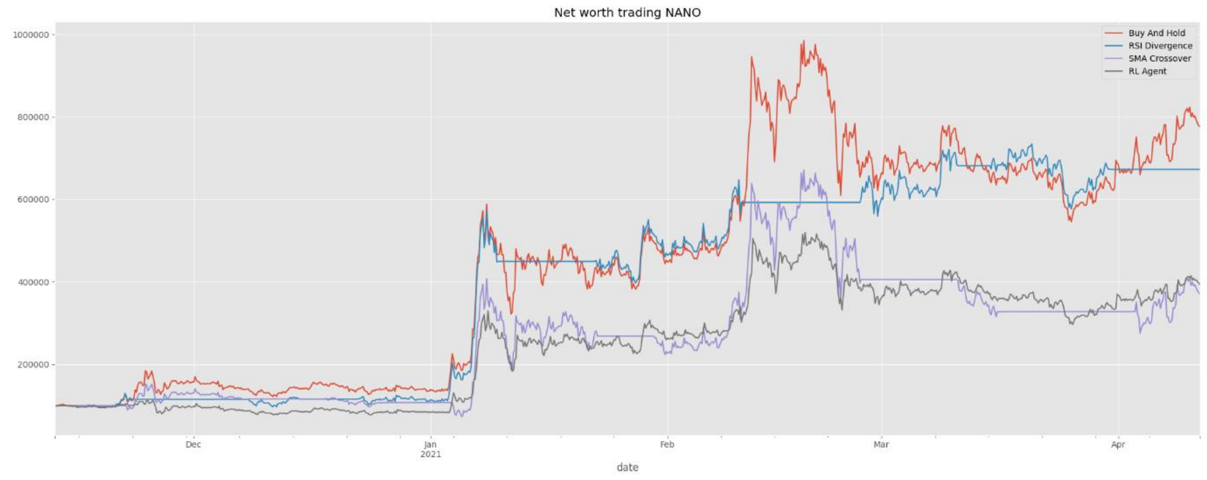

Figure 6: The resulting total net worth trading NANO, using different strategies.

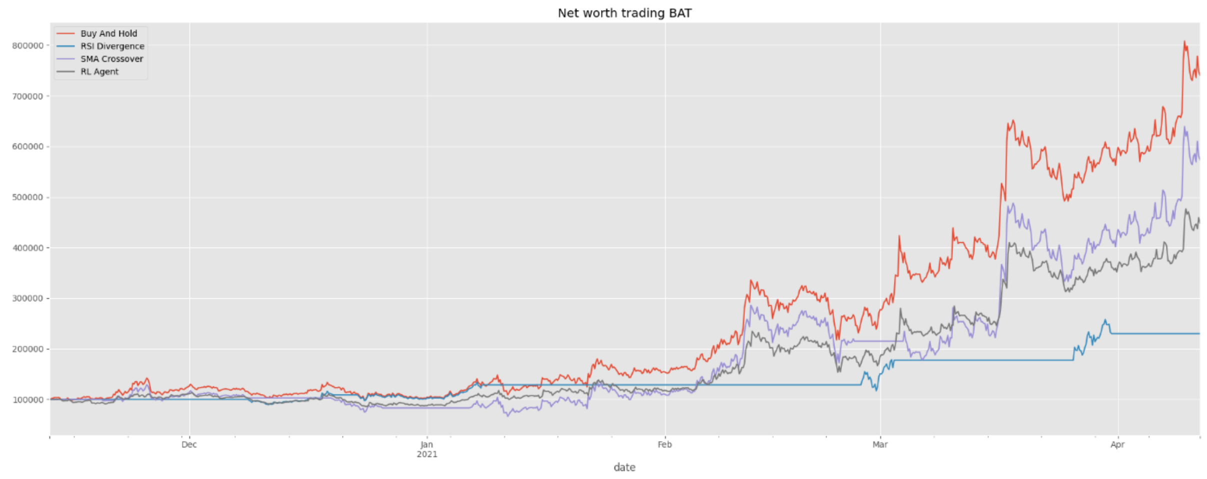

Figure 7: The resulting total net worth trading BAT, using different strategies.

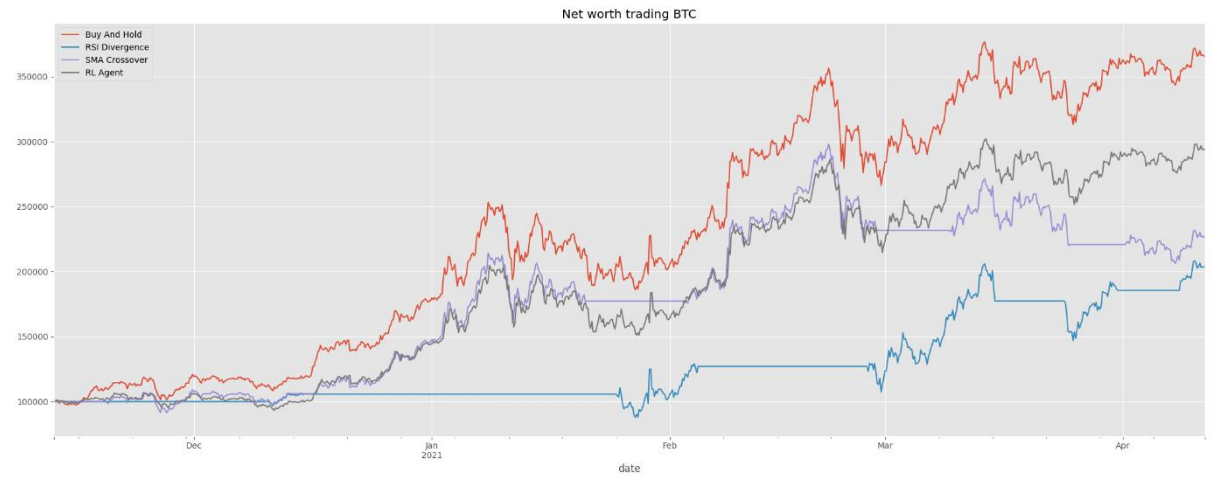

Figure 8: The resulting total net worth trading BTC, using different strategies.

The buy and hold strategy is the overall winner. It performed the best on NANO, where it increased its net worth by eightfold, and at some point was at tenfold. It is evidently that the low market capitalization assets grew the most, during this timeframe. Unfortunately, the reinforcement learning agent was not able to make much use of this growth. While trading NANO, the agent almost got outperformed by all the other strategies, it was only able to barely outperform the SMA crossover strategy. When trading BAT, the reinforcement learning was able to significantly outperform the RSI divergence strategy. This was the only strategy that the agent outperformed, whereas the buy and hold strategy was almost outperforming it by two times. When trading BTC, the reinforcement learning performed best, only being outperformed by the buy and hold strategy. However, looking solely at net worth, the reinforcement learning performed better when trading the small market capitalization assets, which fits the hypothesis that more money can be made trading small market capitalization assets, compared to trading large market capitalization assets.

5. Discussion

In this section the shortcomings and possible improvements are discussed. There are a lot of factors to consider when making a trading bot, especially when using reinforcement learning. The hardest part of getting started is finding a good example. In the world of trading, no one will share their working code, especially if they are making profit from it. That is why coding a profitable trading bot is something complicated to get into.

5.1. Limitations of TensorTrade

TensorTrade is not as recognized as the other libraries. For comparison, TensorFlow has more than 100 times the number of commits [12]. The consequence is that there is not much documentation available and that finding working examples for TensorTrade is hard. This makes using TensorTrade a difficult library to get started with. Another consequence of the unfamiliarity of TensorTrade is that fixing bugs and the addition of new features happens slowly. During the development of the code used for this research, a lot of bugs were encountered, most of them already existing for a long time.

5.2. Limitations of Ray

Ray is a more widespread library than TensorTrade, but it is not as big as TensorFlow. It currently has eight times the amount of commits on GitHub, compared to TensorTrade [13]. Unfortunately, there is also a downside of using Ray. That is that the dashboard is only available when running Linux as the operating system. Since this research was done using Windows 10, the provided dashboard could not be used.

5.3. Limitations of the Data Available

As stated before, the data was gathered using Binance as exchange. The data gathered from this exchange poses no problems. The only downside is that aggregated data is not obtainable. There is not a platform available that offers this for free. The only way this could be done, is if you make an account for the cryptocurrency exchanges that offer an API and combine all their information together. Something that is far out of the scope of this research. This limitation is the least important, but if you want your research to be generalizable it is something to consider.

5.4. Limitations of Features used

As is the case with any machine learning experiment, the features are essential. The features used in this paper are far from perfect, as they only focus on the technical side of things. On the other hand, it also shows that even when solely focusing on the technical aspect of trading, decent results can be obtained. Implementing features that consider the different aspects of trading and price action, would increase performance. Features to consider for further research are: Twitter sentiment, using multiple timeframes (with emphasis on the higher timeframes), and candlestick patterns.

5.5. Parameters

The parameters used in this research provided decent results. This does not mean that these settings will continuously provide the same type of results. There are several reasons to consider. First, the outcome of the results highly depends on the market and how it is behaving at that time. Second, the price action of the asset that is being researched influences the output as well. At last, since a lot of algorithms use probability, getting the same results using the same parameters is highly unlikely.

6. Conclusion

In this section the final thoughts are presented. This paper has shown how reinforcement learning can be used for effectively trading cryptocurrency assets. Moreover, it has shown that the market capitalization of an asset is an important concept to consider, especially when trading cryptocurrencies. This is a topic that can be researched much further than done by this paper alone. This paper has also shown how well certain algorithms perform when trading cryptocurrencies. It was concluded that A2C algorithm is the overall winner, but the other contending algorithms performed just marginally worse. By employing different settings and parameters, this outcome could be influenced. There are numerous reinforcement learning algorithms. This paper did only cover some of the algorithms that were available and ready to use. Future research should focus on implementing newer algorithms and examining how well they perform on trading environments. The graphs in the results section show that buy and hold, despite its simplicity, is a good performing strategy, especially when using this strategy on low market capitalization assets. The reinforcement learning agent was not able to consecutively outperform the benchmark strategies, notably when trading the low market capitalization assets. However, the reinforcement learning agent was able to acquire the highest total net worth when trading low market capitalization cryptocurrencies. Remarkably, the RSI divergence strategy performed very well on the lowest market capitalization asset. This could also be a point of interest for further research.

References

[1] Lee, J. Y. (2019). *A decentralized token economy: How blockchain and cryptocurrency can revolutionize business*. Business Horizons, 62(6), 773–784. https://doi.org/10.1016/j.bushor.2019.08.003

[2] Sabry, F., Labda, W., Erbad, A., & Malluhi, Q. (2020). *Cryptocurrencies and Artificial Intelligence: Challenges and Opportunities*. IEEE Access, 8, 175840–175858. https://doi.org/10.1109/access.2020.3025211

[3] Chen, S. H., Jakeman, A. J., & Norton, J. P. (2008). Artificial Intelligence techniques: *An introduction to their use for modelling environmental systems.* Mathematics and Computers in Simulation, 78(2–3), 379–400. https://doi.org/10.1016/j.matcom.2008.01.028

[4] Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). *Reinforcement Learning: A Survey*. Journal of Artificial Intelligence Research, 4, 237–285. https://doi.org/10.1613/jair.301

[5] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*. Science, 362(6419), 1140–1144. https://doi.org/10.1126/science.aar6404

[6] Adam King. (2021, February 13). *TensorTrade: Trade Efficiently with Reinforcement Learning*. https://github.com/TensorTrade-org/TensorTrade

[7] OpenAI. (2021, February 16). *Gym: A toolkit for developing and comparing reinforcement learning algorithms*. https://gym.openai.com/

[8] Ray. (2021, March 19). *Ray – Fast and Simple Distributed Computing*. https://ray.io/

[9] CoinMarketCap. (2021, March 25). *Top Cryptocurrency Exchanges Ranked By Volume.* https://coinmarketcap.com/rankings/exchanges/

[10] Schulman, J. (2020, September 2). Proximal Policy Optimization. OpenAI. https://openai.com/blog/openai-baselines-ppo/

[11] Fee Schedule. (2021, March 28). *Binance*. https://www.Binance.com/en/fee/schedule

[12] TensorFlow. (2021, March 29). *TensorFlow GitHub*. https://github.com/tensorflow/tensorflow

[13] Ray. (2021, March 29). Ray GitHub. https://github.com/ray-project/ray

[14] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S. & Kavukcuoglu, K.. (2018). *IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures*. Proceedings of the 35th International Conference on Machine Learning, in Proceedings of Machine Learning Research 80:1407-1416. http://proceedings.mlr.press/v80/espeholt18a.html.

[15] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M.A. (2013). *Playing Atari with Deep Reinforcement Learning*. ArXiv, abs/1312.5602.

[16] Padial, D. L. (2021, 9 april). *Technical Analysis Library in Python*. https://github.com/bukosabino/ta

[17] CoinGecko. (2021, April 3). *Basic Attention Token (BAT) - price, market capitalization, chart, and information.* https://www.coingecko.com/nl/coins/basic-attention-token

[18] Padial, D. L. (2021, 9 april). *Bollinger High Band Indicator*. Technical Analysis Library Documentation. https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html?highlight=hband#ta.volatility.bollinger_hband_indicator

[19] Ray. (2021, 11 april). *RLlib Algorithms — Ray v2.0.0.dev0*. https://docs.ray.io/en/master/rllib-algorithms.html

[20] CoinGecko. (2021, 11 april). *Nano (NANO) - price, market capitalization, chart, and information.* https://www.coingecko.com/nl/coins/nano

[21] 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

[22] Akyildirim, E., Goncu, A., & Sensoy, A. (2020b). Prediction of cryptocurrency returns using machine learning. *Annals of Operations Research*, *297*(1–2), 3–36. https://doi.org/10.1007/s10479-020-03575-y

[23] Wu, Y. (2020, June 29). *OpenAI Baselines: ACKTR & A2C*. OpenAI. https://openai.com/blog/baselines-acktr-a2c/