Deep Q Learning for Cryptocurrency Trading

Carlos Fernandez
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14623
csf5856@rit.edu

***Abstract*—This project consists of an implementation of a Deep Q Learning trading model that can determine whether to buy, sell or hold, based on some past price information and the analysis of the contribution that technical indicators have in the performance of the aforementioned model. The model performed similarly to previous work, and we showed that using technical indicators can significantly improve performance.**

***Index Terms*—trading, cryptocurrency, neural networks, deep Q, reinforcement learning, technical indicators.**

I. Introduction

Since the creation of cryptocurrencies, professional stock traders have noticed their similarities with the stock market. Thus, a lot of tools were created to trade cryptocurrency in a similar way to stocks. In 2016, more than 80% of the trades in the forex market were automated [2]. Due to the recent rise in popularity of cryptocurrencies, many have started to see them as a viable form of investment. However, not everyone has the necessary trading expertise to trade profitably on such a volatile market. This is why we will be exploring the possibility of using machine learning to automate the trading process. **We hypothesize that using technical indicators to train the machine learning model will improve performance and make the solution more robust.**

II. Background

A. Basic Trading Knowledge

In order to understand our approach, it is essential to understand some basics trading concepts. Since the target reader of this paper might not know, we will briefly explain these concepts.

Trading is the act of buying, selling or holding an asset with the objective of making a profit. Since cryptocurrencies behave similarly to the stock market, many stock trading concepts are used for cryptocurrency trading. One of these concepts is the candlestick graph.

A candlestick graph is a method of visualizing a given market. It consists of "candles," which contain information about four aspects of the price in a given time interval, the highest price reached, the lowest price reached, the price at the beginning of the time interval, and the price at the end. Candles are colored depending on whether the opening price is higher than the closing price or not. The y axis is the price of the cryptocurrency, and the x axis is the time. Professional traders use these graphs to make trades.

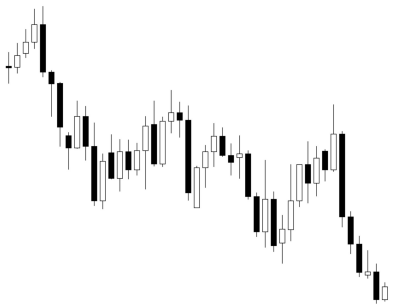

Fig. 1. Example of a candlestick graph.

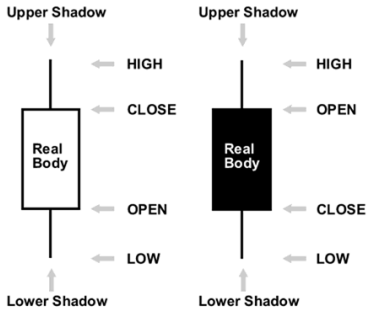

Fig. 2. Structure of a candle.

B. Data Structure & Trading Strategy

Unlike most other implementations of AI in cryptocurrency trading, our objective is not to train our AI to predict the price of the cryptocurrency but instead train the AI to make the best decisions possible based on the data we have available. We need to feed the AI model with data similar to what a human trader would need to determine the decision that maximizes profit.

Using the Binance API, we created a dataset with the data necessary to plot the candlestick graph of the cryptocurrency XRP in 15-minute intervals (each candle represents 15 minutes). This dataset contains the candlestick graph's information. It also includes key information about the market

like the number of trades made and the volume (amount of currency traded), and other information (See table VI).

In addition to the raw data obtained from the Binance API, we also calculated some technical indicators that professional traders use to decide when and how to trade. These indicators will be discussed in detail in the following section.

C. Technical Indicators

To improve the performance of our model, we decided to use common technical indicators used by traders [9]. This section will define these indicators and explain how they can be used to generate trading signals.

- **Simple Moving Average (SMA) & Exponential Moving Average (EMA):** The Simple and Exponential moving averages are technical indicators that provide information about emerging and common trends in an market environment. They achieve this by averaging the last n prices, with the key difference between them being that the EMA gives more weight to the most recent data points. We can calculate the SMA and EMA as follows:

$$SMA_t = \frac{1}{n}\sum_{k=0}^{n-1} p_{t-k} \qquad \text{with } t \geq n-1 \tag{1}$$

$$EMA_t = p_t\left(\frac{\alpha}{n+1}\right) + EMA_{t-1}\left(1 - \frac{\alpha}{n+1}\right) \tag{2}$$

With p_t being the price at time t, n being the size of the window, and α is a smoothing factor. A smoothing factor of 2 was used.
To generate a trading signal from this information, traders look for intersections between 2 moving averages with different window sizes. The one with a smaller window size is called “fast", and the other is called “slow". If the fast-moving average rises over the slow one, this signals for a buying position. On the other hand, if the fast-moving average falls below the slow one, this signals a selling position.
- **Relative Strength Index (RSI):** This indicator provides information about the momentum of price trends and whether the market is overbought or oversold and belongs to the oscillator family of indicators, which means that its value oscillates between 0 and 100. We can calculate the RSI with this formula:

$$RSI_t = \begin{cases} 100 - \left(\dfrac{100}{1 + \frac{g_{avg}}{l_{avg}}}\right) & \text{for } t < n \\ 100 - \left(\dfrac{100}{1 + \frac{(pg_{avg}\cdot(n-1))+g_t}{-(pl_{avg}\cdot(n-1))+l_t}}\right) & \text{for } t \geq n \end{cases} \tag{3}$$

With t begin the time, n the window size, g_{avg} the average gain of all the previous periods, l_{avg} the average loss of all the previous periods, g_t is the current gain, l_t is the current loss, pg_{avg} is the average gain of the previous n periods excluding the current one and pl_{avg} is the average loss of the previous n periods excluding the current one.
If the RSI is below 30 and then raises over 30, that would be considered a buy signal. On the other hand, if the RSI is above 70 and then falls below 70, that would be interpreted as a sell signal.
- **Average Directional Index (ADX):** This indicator is a measure of the strength of a trend. Because ADX itself does not provide information about the direction of the trend, it is commonly paired with two other signals: the negative directional indicator (-DI) and the positive directional indicator (+DI). To calculate these three signals, we will require multiple formulas.

$$+DM_t = h_t - h_{t-1} \tag{4}$$

$$-DM_t = l_t - l_{t-1} \tag{5}$$

$$S(DM)_t = DM_t - \left(\frac{1}{n} - 1\right)\sum_{k=1}^{n} DM_{t-k} \tag{6}$$

$$TR_t = max\left[(h_t - l_t), |h_t - p_t|, |l_t - p_t|\right] \tag{7}$$

$$ATR_t = \frac{1}{n}\sum_{k=0}^{n-1} TR_{t-k} \tag{8}$$

$$-DI_t = 100 \times \left(\frac{S(-DM)_t}{ATR_t}\right) \tag{9}$$

$$+DI_t = 100 \times \left(\frac{S(+DM)_t}{ATR_t}\right) \tag{10}$$

$$DX_t = 100 \times \left(\frac{|(+DI_t) - (-DI_t)|}{|(+DI_t) + (-DI_t)|}\right) \tag{11}$$

$$ADX_t = \frac{(ADX_{t-1}\cdot(n-1)) + DX_t}{n} \tag{12}$$

With h_t being the highest price at time t, l_t the lowest price, p_t the current price and ATR_t is the average true range.
Now that we have our three signals (ADX, +DI, -DI), we can generate a trading signal. Traders consider an ADX greater than 25 as an indication of a strong trend, so if the ADX is greater than 25, we might want to consider trading, but should we buy or sell? To answer this question, we must look at the other two signals. If the +DI is greater than -DI, this means prices are rising. Therefore, if the ADX over 25, this would be considered a buying signal. On the contrary, if the +DI is greater than -DI, this means prices are decreasing. So, if the ADX over 25, this would be a selling signal.
- **Stochastic oscillator (STOCH):** This is a momentum indicator that compares the current closing price to a range of its previous prices over n timesteps. Similar to RSI, it tells us if the market is overbought or oversold, and its value oscillates between 0 and 100. It consists

of a fast signal K and a slow signal D, and they can be calculated as follows:

$$L_n = min\left[l_t, l_{t-1}, \cdots, l_{t-n-1}\right] \tag{13}$$

$$H_n = max\left[h_t, h_{t-1}, \cdots, h_{t-n-1}\right] \tag{14}$$

$$K_t = 100 \times \left(\frac{p_t - L_n}{H_n - L_n}\right) \tag{15}$$

$$D_t = \frac{1}{3}\sum_{k=0}^{3} K_{t-k} \tag{16}$$

With l_t being the lowest price at time t, h_t the highest price at time t, L_n the lowest price of the last n timesteps, and H_n the highest price of the last n timesteps.

Like SMA and EMA, to generate a trading signal using the stochastic oscillator, we will be identifying intersections between the fast and slow values. If the fast signal (K) rises above the slow signal (D), this would be a buying signal unless the values are over 80. And if K falls below D, we would consider this a selling signal unless the values are lower than 20.

- **Moving average convergence divergence (MACD):** This is another momentum indicator that provides insight into the power of price movements in the market. The calculations for this indicator involve 3 EMAs corresponding to the fast, slow, and signal values. To calculate the MACD lines, we need to use the following formulas:

$$MACD_t = EMA_m - EMA_n \tag{17}$$

$$sig_t = MACD_t\left(\frac{\alpha}{o+1}\right) + sig_{t-1}\left(1 - \frac{\alpha}{o+1}\right) \tag{18}$$

$$his_t = MACD_t - sig_t \tag{19}$$

With $n > m > o$, EMA_m being the exponential moving average of the last m prices, EMA_n the exponential moving average of the previous n prices, sig_t the signal line of the indicator, and his_t a histogram that shows the directional momentum of the trends.

To generate a trading signal using the three values calculated above, we will again use intersections between the $MACD$ line and the faster sig line as a trigger to perform a trade. If the sig line crosses above the $MACD$ line, we will consider this a buy signal. And if the sig line falls below the $MACD$ line, it would be a sell position.

When used individually, these indicators can not provide a profitable trading strategy because they can produce false trading signals. This is why traders use multiple indicators to validate trends and market conditions before making a trade. **Incorporating these indicators into the raw data obtained from the Binance API could improve the performance of our model.**

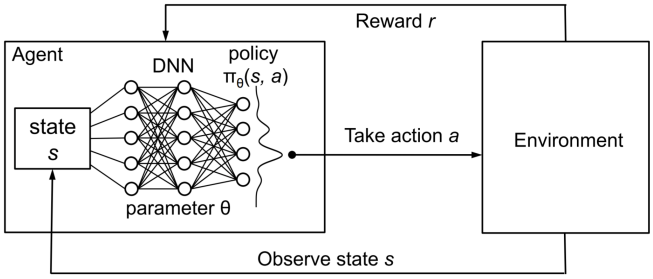

Fig. 3. Reinforcement Learning with policy represented via DNN. [12]

D. Deep Q-networks Algorithm

Deep Q-Network (DQN) is a Reinforcement Learning (RL) algorithm that can learn how to act in an environment by estimating the value of taking a given action in any given state of the environment using a deep neural network. This algorithm, introduced by Mnih et al. [13], has been proven to work well with high-dimensional states like frames from a video game. This is why we will be using it for the task of automating cryptocurrency trading. An overview of the structure of DQN can be seen in Fig. 3.

To further improve the performance of this algorithm, we have deviated a little from its original implementation. We applied some of the recent improvements to the machine learning algorithms involved in DQN. Some of the modifications we made coincide with the ones made by Théate et al. [16]. These modifications are:

- **Xavier initialization:** To prevent the gradients from vanishing or exploding during the training of our neural network, we used the method described by Glorot et al. [6]. This will ensure that the variance of the gradients stays constant across all the layers of our neural network.
- **Double DQN:** The original DQN algorithm can suffer from some instability issues because the same network is used to get the values from the current state and the next state. Because updating the values for the current state would affect the values of the next state, this would be like the network is chasing its tail. This is why we used two neural networks (policy network and a target network), as is described in van Hasselt et al. [17].
- **ADAM optimizer:** After our experiments we determined that the optimizer described by Kingma et al. [10] improved stability and reduced training time.
- **Smooth L1 Loss:** We decided to change the loss function of the original implementation because, during our experiments, we noticed that when the error decreased below 1, the loss became too small and thus increased training time and prevented the model from converging on a correct solution.

- **Batch normalization layers:** As described by Ioffe et al. [7], adding batch normalization layers to our neural network helped decrease training times and further improved stability.
- **Gradient clipping:** During our experiments, we were still facing issues regarding exploding gradients even with all the modifications mentioned above. This is why we added gradient clipping during the training process.
- **Data normalization:** We normalized the data by dividing by its standard deviation.
- **Action Simulation:** At each time step, after we choose an action to perform, we conduct a simulation of performing the remaining actions and push the result of these simulations into replay memory along with the outcome of actually performing the originally chosen action.

III. Implementation

A. Environment

After the data was downloaded and the technical indicators were calculated and added to the data, we used this dataset to create an environment for the DQN agent. The environment keeps track of 2 accounts, one for USD and one for cryptocurrency. Each observation of the environment consists of the last n entries $(t, t-1, \cdots, t-n-1)$ in the dataset (this is the window size), the latest price traded by the agent, a boolean variable indicating if there is money left on the USD account, a boolean variable indicating if there is crypto left in the crypto account, and trading signals generated from each of the indicators discussed earlier. This results in a large observation space.

When a trade is performed in the environment, a trading cost is applied. This is consistent with real trading systems.

The environment reaches a terminal state when the profit percentage falls below -30% or when we reach the end of the dataset.

B. Reward function

The most important part of a DQN is the reward function. It defines the agent's behavior, and if it is not robust enough, the agent might behave in unexpected ways. This is why it is best to keep the reward function as simple as possible. We decided to use the profit difference percentage as reward, and it is calculated as:

$$r_t = 100 \times \frac{P_t - P_{t-1}}{P_{t-1}} \tag{20}$$

Where P_t is the profit at time t.
This reward function is simple enough for our agent to be stable. However, if the agent performs an invalid trade, the reward gets updated to -10. An invalid trade is when, for example, the agent tries to buy, but there is no money in the USD account.

C. Dadaset information

The data downloaded was for the XRP-USD cryptocurrency pair, and it starts on January 1, 2019, and ends on April 13, 2021. Because it comes in 15-minute intervals, it contains a total of 79794 rows.

D. Performance Metrics

Since DQN is an unsupervised learning method, meaning that we don't have any labeled data to compare to, we need other ways of measuring the performance of our algorithm. So, to measure the performance of the trading model, we will be using the following methods:

- **Sharpe Ratio:** Metric that measures the performance of an investment compared to a risk-free investment. It is calculated with the following formula:

  $$S_a = \frac{P_a - P_b}{\sigma_a} \tag{21}$$

  With P_a being the profit of our DQN, P_b being the return rate of a risk-free investment like Treasury bonds, and σ_a the being the standard deviation of the $P_a - P_b$.
- **Profit:** Amount of money gained from trading with the initial investment.

  $$P_t = B_{USD} + (B_c * p_t) - I \tag{22}$$

  With B_{USD} being the current USD balance, B_c the current crypto balance, p_t the current price, and I the initial investment in dollars.
- **Max Dragdown:** Minimum profit achieved during a trading session.

E. Parameters

To choose the hyper-parameters for our model, we tried different combinations of parameters and tested them for convergence. Some hyper-parameters resulted in vanishing gradients, exploding gradients, or overfitting. When it comes to the neural network itself, many rule-of-thumb methods exist to determine the size of the hidden layers [8]. We decided to take $\frac{2}{3}(l_{in} + l_{out})$ as our rule-of-thumb to chose the number of neurons on our hidden layers, where l_{in} is the number of input layers and l_{out} is the number of output layers. We then divided the resulting number into three hidden layers. An overview of the hyper-parameters used in our experiments can be seen in Table II.

IV. Results

After training our model multiple times with different initial weights, we picked the best performing model and compared it with the results from previous implementations of similar trading algorithms.

The testing dataset consists of the last 10% of the pre-processed dataset, which means that the testing starts around January 2021. The model has never seen this data.

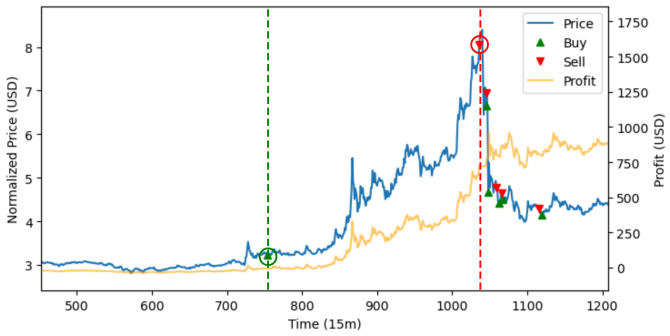

Fig. 4. Our model preemptively identifies an emerging upward trend.

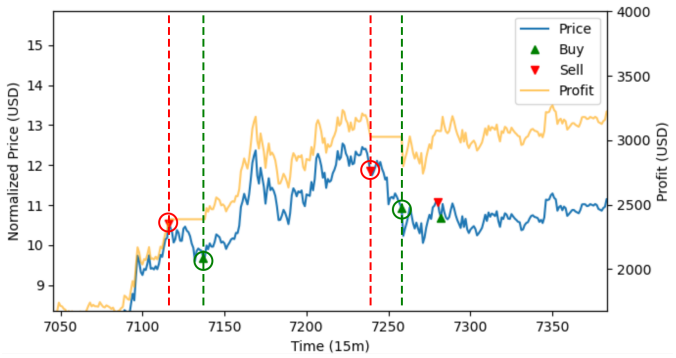

Fig. 5. Our model detects small downward trends and sells to minimize losses.

A. Baseline

To assess the performance of our model, we need a baseline to compare it to. The strategy we chose for this was the Buy and Hold strategy (B&H), which consists of buying at the beginning of the trading session and holding until the end. If our model can outperform this baseline, it would mean that the model can perform well regardless of market conditions.

B. Our model

In general, our model performed well, doubling the profit achieved by the B&H strategy and scoring a better Max Dragdown. However, due to the added risk introduced by our agent's trades, our model achieved a slightly lower Sharpe ratio than the B&H strategy. See Table I and Figure 8.

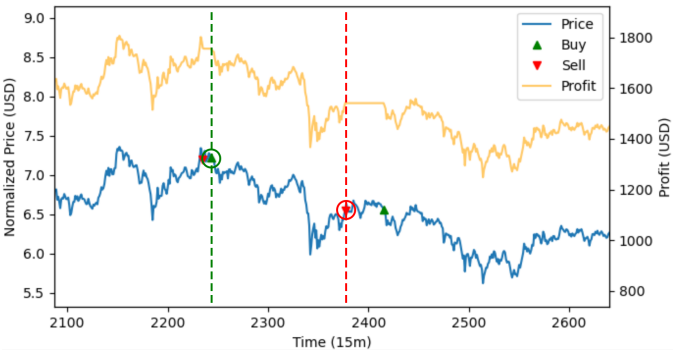

Fig. 6. Our model performs a losing trade.

C. Behavior Analysis

In this section, we will analyze the agent's behavior with some examples extracted from figure 8. We will show examples of good and bad actions taken by the agent.

- **Good behavior:** Figures 4 and 5 were obtained by zooming in on Figure 8. In figure 4, we can see that the agent successfully detected that an upward trend was about to emerge and therefore performed a buying trade. It held the position and then sold at a reasonable price, almost tripling the initial value. After this, the agent made some subsequent trades that made a reasonable profit as well. In figure 5 we can see how the agent identifies small downward trends and sells the asset to minimize losses.
- **Bad behavior** In figure 6, we can see that our model failed to identify the downwards trend and decided to buy, resulting in a loss in profit. This might be due to fake buying generated by some of the indicators. In figure 7, we can see that the agent is trading erratically. We can also see (circled in purple) that the agent bought and sold at the same price. This would result in a direct loss because of the trading costs. This behavior is undesirable as it can decrease the sharpe ratio.

D. Comparison with previous work

To properly compare our work with the previous work cited in this report ([16], [19], [11]), we would have to implement all their solutions and apply them to the data we have gathered, but due to the time frame of this project, that would be unfeasible. However, we will still compare the performance indicators achieved by previous work as a form of context. The performance of the previous models can be seen in table III.

From table III, we could compare how each model performed against their baseline. The percentage difference between each of the models performance and their B&H performance can be seen in table IV.

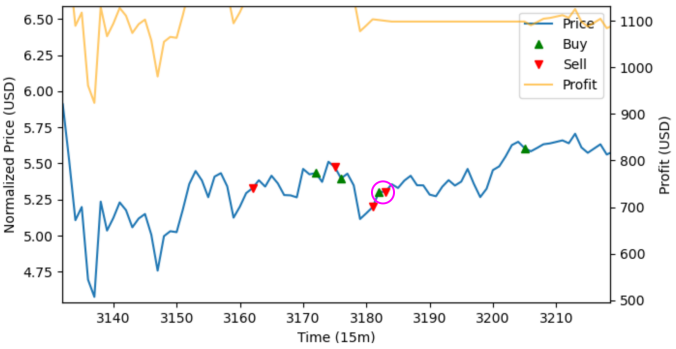

Fig. 7. Our model exhibits erratic behavior.

E. Using Technical Indicators vs. Not using them

To show that our hypothesis holds, we need to evaluate the performance of a model trained using the technical indicators that we calculated and compare it to the performance of training the model without these indicators. To make this comparison, we performed additional tests. Using the same parameters, reward function, and data, we trained two models, one using only the raw data as downloaded from the Binance API, and another one using the same data but including the technical indicators and their respective trading signals as described in section II-C.

In figures 9 and 10, we can see that the model that utilized technical indicators converged faster and arrived at a higher profit than the one that was trained without technical indicators. The model with technical indicators reached its maximum profit at 34 episodes of training while the one without the indicators reached a much lower maximum profit at 86 episodes.

The model that didn't use the technical indicators could not outperform the baseline strategy regardless of the number of episodes we train it for, as you can see in table V. This behavior confirms our hypothesis that technical indicators improve the performance of the model.

V. Future Work

In this section, we will be discussing some of the work that could be done in the future with the knowledge gathered in this paper. Implementing these improvements could help turn our model into a robust method of investment.

A. Add more technical indicators

In this project, we used 6 of the most popular technical indicators. However, there might be several more indicators that could further improve the performance of the model. Furthermore, new indicators could be developed that are specifically designed for machine learning applications.

We could also perform feature analysis on these indicators to determine which of them are the most impactful to the model's performance.

B. Incorporate sentiment analysis data

Like in the stock market, the general sentiment of the public has an impact on the cryptocurrency market. This sentiment data could provide additional information that can't be inferred from the raw data or the technical indicators. Incorporating this data into our solution might help improve performance even more.

C. Monitor multiple currencies

It has been shown before that cryptocurrencies, in general, are interdependent [3]. This could mean that monitoring multiple currencies at the same time could improve the performance of the model proposed in this project.

TABLE I
Performance comparison against the baseline.

Performance indicator	B&H	Ours
Profit (%)	534.77	1069.71
Shape Ratio	6.01	5.79
Max Dragdown (%)	−11.38	−7.99

TABLE II
List of parameters used for training.

Parameter	Value
Trade cost (%)	0.1
Window Size	30
Learning Rate	0.0001
Hidden Layers	[294, 294, 294]
Batch Size	128
Episodes	300
Discount	0.99
Initial Epsilon	0.6
Final Epsilon	0.0
End Epsilon Decay	190000
Replay memory Size	200000
Update Target Network	10000

VI. Conclusion

In this project, we implemented a working DQN Trading model. We analyzed the behavior of the model and determined that it could preemptively identify emerging trends in the market. We showed that using technical indicators while training such DQN can significantly improve the model's convergence time and general performance. This project can be the starting point for creating a robust Machine Learning Trader that can perform good investments and manage a portfolio autonomously.

References

[1] Akhil Azhikodan, Anvitha Bhat, and Mamatha Jadhav. *Stock Trading Bot Using Deep Reinforcement Learning*, pages 41–49. 05 2019.

[2] Alessandro Bigiotti and Alfredo Navarra. Optimizing automated trading systems. In Tatiana Antipova and Alvaro Rocha, editors, *Digital Science*, pages 254–261, Cham, 2019. Springer International Publishing.

[3] Pavel Ciaian, Miroslava Rajcaniova, and d'Artis Kancs. Virtual relationships: Short- and long-run evidence from bitcoin and altcoin markets. *Journal of International Financial Markets, Institutions and Money*, 52:173–195, 2018.

[4] Georg Dorffner. Neural networks for time series processing. *Neural Network World*, 6:447–468, 1996.

[5] Prakhar Ganesh and Puneet Rakheja. Deep neural networks in high frequency trading. *arXiv preprint arXiv:1809.01506*, 2018.

TABLE III
PERFORMANCE OF PREVIOUS WORK.

Performance indicator	Théate et al. (AAPL) B&H	Théate et al. (AAPL) Model	Koker et al. (BTC) B&H	Koker et al. (BTC) Model	Wang et al. (HSI) B&H	Wang et al. (HSI) Model	Ours (XRP) B&H	Ours (XRP) Model
Profit (%)	79.82	100.29	135.20	339.80	154.00	350.00	534.77	1069.71
Shape Ratio	1.24	1.48	3.70	6.40	0.28	0.59	6.01	5.79
Max Dragdown (%)	−38.51	−17.31	−61.60	−43.40	−65.00	−42.00	−11.38	−7.99

TABLE IV
PERFORMANCE COMPARISON AGAINST PREVIOUS WORK.

	Théate et al. (AAPL)	Koker et al. (BTC)	Wang et al. (HSI)	Ours (XRP)
Profit Difference (%)	25.65	151.33	127.27	100.03
Shape Ratio Difference (%)	19.35	72.97	110.71	−3.66
Max Dragdown Difference (%)	55.05	29.55	35.38	29.79

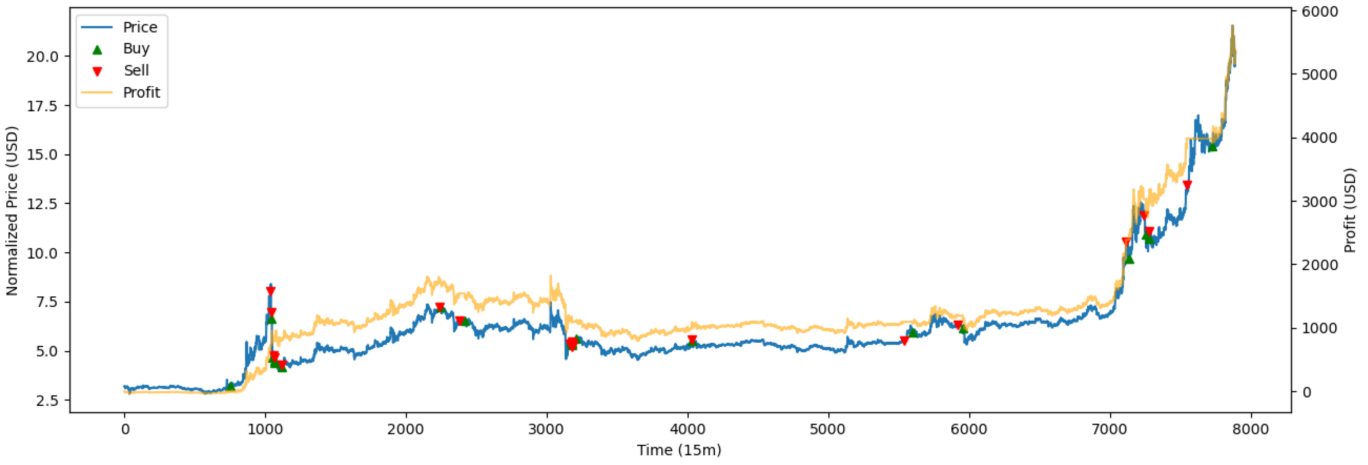

Fig. 8. Trades performed by our model on the testing data after training

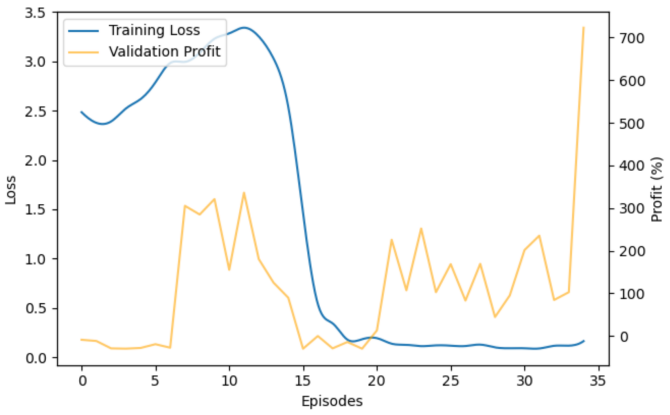

Fig. 9. A Model trained using data with technical indicators.

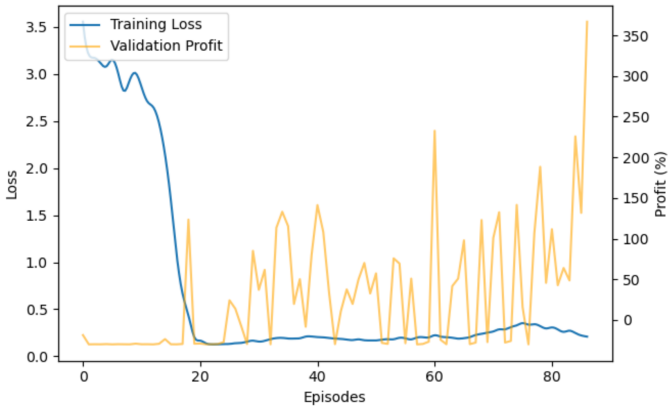

Fig. 10. A Model trained without technical indicators.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating

TABLE V
PERFORMANCE COMPARISON OF THE USE OF TECHNICAL INDICATORS.

Performance indicator	B&H	With Technical Indicators	Without Technical Indicators
Profit (%)	534.77	736.56	356.99
Shape Ratio	6.01	5.68	4.35
Max Dragdown (%)	−11.38	−8.05	−12.03

TABLE VI
BINANCE API DATA STRUCTURE

Column	Description
Date	Starting date
Open	Price at the starting date
High	Highest price beetween the starting date and the closing date
Low	Lowest price beetween the starting date and the closing date
Close	Price at the closing date
Volume	Volume of USD transacted beetween the starting date and the closing date
Close Date	Closing date
Asset Volume	Volume of crypto transacted beetween the starting date and the closing date
Number of Trades	Number of trades made beetween the starting date and the closing date
Taker buy base asset volume	Volume of crypto received by the buyers after taking an order
Taker buy quote asset volume	Volume of USD payed by the buyers to take an order

deep network training by reducing internal covariate shift, 2015.

[8] Somayeh Kazemi. How to decide the number of hidden layers and nodes in a hidden layer?, 07 2017.

[9] Anzél Killian. 10 trading indicators every trader should know, Jun 2019.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[11] Thomas E. Koker and Dimitrios Koutmos. Cryptocurrency trading using machine learning. *Journal of Risk and Financial Management*, 13(8), 2020.

[12] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks*, pages 50–56, 2016.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[14] David MQ Nelson, Adriano CM Pereira, and Renato A de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International joint conference on neural networks (IJCNN)*, pages 1419–1426. IEEE, 2017.

[15] A. Radityo, Q. Munajat, and I. Budi. Prediction of bitcoin exchange rate to american dollar using artificial neural network methods. In *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 433–438, 2017.

[16] Thibaut Théate and Damien Ernst. An application of deep reinforcement learning to algorithmic trading, 2020.

[17] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

[18] Au Vo and Christopher Yost-Bremm. A high-frequency algorithmic trading strategy for cryptocurrency. *Journal of Computer Information Systems*, pages 1–14, 12 2018.

[19] Yang Wang, Dong Wang, Shiyue Zhang, Yang Feng, Shiyao Li, and Qiang Zhou. Deep q-trading. *cslt. riit. tsinghua. edu. cn*, 2017.

GLOSSARY

batch Group of examples to feed into a neural network at once. 3, 6

discount Discounting factor for future reward. 6

episode The period in which the agent acts on the environment, starting in the initial state and ending on a terminal state. 6

epsilon Probability of choosing a random action. 6

replay memory Collection of experiences the agent has seen before. 4, 6

validation profit Profit obtained by testing a model each training episode.