Deep Reinforcement Learning in CryptocurrencyTrading: A Profitable Approach

Abstract: This study proposes an Automatic Cryptocurrency Trading System using Deep Reinforcement Learning (DRL). Six popular cryptocurrencies were used: Bitcoin, Ethereum, BinanceCoin, DogeCoin, Cardano, and WAVES. Development of the trading system started with building three timeseries models – Temporal Convolutional Neural Network (TCNN), Long Short-Term Memory Network (LSTM), and Gated Recurrent Unit Network (GRU) – to predict future prices. Then, cryptocurrency sentiment data was scraped using the Alternative.me API. Data on historical prices, predicted future prices, cryptocurrency sentiment index, technical indicators, and trading account information was fed as input states to three DRL Agents — Deep Q Network (DQN), Advantage Actor Critic (A2C), and Recurrent Proximal Policy Optimization (RPPO) — which were trained using a custom-developed trading environment. Each agent was given $1000 initial capital for all six cryptocurrencies to trade using three possible actions — Buy, Sell and Hold — and were back-tested on one year of unseen data. Our DQN model had the highest overall return on investment (ROI) of $740, an average 12.3% ROI across all six cryptocurrencies, with an ROI of 63.98% achieved for BinanceCoin. However, A2C and RPPO both had negative ROI.

Keywords: Investment, Inflation, Cryptocurrency, Timeseries, Deep Reinforcement Learning

Introduction

The Department of Statistics Malaysia has reported that Malaysia's inflation rate has risen to 1.8% year-on-year as of February 2024 which is an increase compared to January 2024 and December 2023, which were both at 1.5% (DOSM, 2024a; 2024b; 2024c). The worrying trend of inflation means investing is becoming more important for wealth preservation. Popular investments used by Malaysians include the Employees' Provident Fund (EPF) and the Amanah Saham Bumiputera 2 (ASB2), which had returns of 5.5% and 5%, respectively, in FY2023 (VethaSalam & Ibrahim, 2024; "ASNB declares FY23 dividends", 2023). Despite these investments providing relatively favourable returns, we believe that Deep Reinforcement Learning (DRL) agents capable of ingesting and analysing complex patterns in data can outperform these traditional investing methods by taking advantage of the high volatility present in cryptocurrency prices to yield much higher returns (Tradingview, n.d.) .

In this study, we developed an investing system that trades cryptocurrencies in the daily time frame. The cryptocurrencies selected are Bitcoin (BTC-USD), Ethereum (ETH-USD), BinanceCoin (BNB-USD), DogeCoin (DOGE-USD), Cardano (ADA-USD), Polygon (MATIC-USD), Avalanche (AVAX-USD), and WAVES (WAVES-USD). (While eight cryptocurrencies are listed here, two are later excluded from the final analysis.) The data collected are the historical daily price data of the aforementioned cryptocurrencies retrieved using Yahoo's yFinance API.

The scope of this study includes three DRL algorithms, which are Dueling Deep Q Network (DQN), Advantage Actor Critic (A2C) and Recurrent Proximal Policy Optimization (RPPO). Agents are trained using a sophisticated reward function, which rewards them for profit generation and punishes the agents heavily for losses. Specifically, the proposed reward function multiplies the agents' losses by 3 and profits by 2 in each time step. The impact of losses is intensified to ensure the agents learn risk-averse strategies. The performance metric used to evaluate the agents is the annual return given $1000 during back-testing.

Besides cryptocurrency prices, another source of data is cryptocurrencies' sentiment index, also known as the Fear and Greed index, which is scraped using Alternative.me's API. This API is reliable since Binance has used it to report cryptocurrency sentiments (Binance Square, n.d.). After data ingestion, data preprocessing, and feature engineering, the data is passed as input to the DRL agents. The objective is to enable the agents to leverage both the market data and sentiment analysis for more informed decision-making.

Timeseries prediction is also performed using a four-day window of historical closing prices of the cryptocurrencies. The output of the prediction is used as an additional input feature to the DRL agents to help them learn faster. The timeseries prediction models used were Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and Temporal Convolutional Neural Network (TCNN). The objective is to enhance the agents' training efficiency by providing insights into expected future price movements, thereby simplifying their decision-making process during training.

The aim of this study is to develop an intelligent investment system that is able to help users mitigate the impact of inflation and explore a novel investment approach to earn higher returns than those provided by conventional investments. By leveraging Deep Reinforcement Learning and incorporating additional insights, such as sentiment analysis and future price movements, the system aims to make informed trading decisions and learn profitable trading strategies. The ultimate impetus is to help users achieve earlier retirement, financial peace and more control over their lives.

Related Works and Algorithms

Reinforcement learning

The core of this project is based on a type of machine learning called reinforcement learning (RL). This type of learning involves an AI agent that interacts with an environment by observing the state of the environment and taking an action with the goal of maximizing the reward gained by the agent. Some RL algorithms are Value-based, in which they find the value of being in each state (how good a particular state is). Another type of RL algorithm is Policy-based, where they map the observed states to the probability distribution of different actions to take, and then taking the action with the highest probability (Or, 2021). Deep Reinforcement Learning (DRL) is a technique which combines Deep Learning with Reinforcement Learning to solve more complex challenges. These include tasks relying on visual inputs (e.g., gaming with pixels as input).

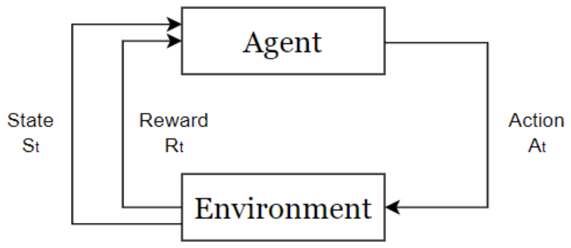


Figure 1. A Reinforcement Learning cycle diagram

Duelling Deep Q Network

In traditional Q learning, a Q table maps state and action pairs to their expected future reward using the Bellman optimality equation:

$$Q^*(s,a) = E\ [R_{t+1} + \gamma\ max_{a'}\ Q_*\ (S_{t+1}, a') |\ S_t = s, A_t = a] \qquad (1)$$

where the optimal action-value function $Q^*(s,a)$ is the immediate reward, R_{t+1}, of taking action a, added to the maximum action-value function of all possible actions at the next time step, a', during the future state, S_{t+1}, achieved after taking action a (Sutton & Barto, 2018).

With Deep Q Network (DQN), a neural network is used to approximate the Bellman optimality equation:

$$Q(s,a;\ \theta) \approx Q^*(s,a) \tag{2}$$

This eliminates the need for a large Q table when state-action pairs scale infinitely. The parameter updates are computed with the following equation (Or, 2021):

$$\theta_{k+1} = \theta_k + \ \alpha(r + \gamma max_{a'} Q(s', a';\ \theta_k) \ - Q(s,a;\ \theta_k)\ ) \nabla_{\theta_k} Q(s,a;\ \theta_k) \tag{3}$$

With Duelling Deep Q Network, the computation of how much better an action is compared to all other possible actions in the action space is done with the addition of an Advantage (A) value. The Advantage function providing this value can be represented as:

$$A(s_t,\ a_t) = \ Q_\theta(s_t,\ a_t) - V_v(s_t) \tag{4}$$

The Q value computes the value of taking a specific action at the given state, whereas the V value computes the value of a given state independently of the action taken (Yoon, 2019a). To represent the Q function of the Duelling DQN in a simpler manner (Or, 2021), we can express the function as:

$$Q(s,a) = A(s,a) + V(s) \tag{5}$$

Sornmayura (2019) shows a study using DQN to trade forex such as EUR-USD and USD-JPY with historical price and technical indicators as input features. The author's DQN agent achieved a 43.8% mean annual return on 12 years of EUR-USD historical data and 26.7% for that of USD-JPY, whereas the Commodity Trading Advisor (CTA) only achieved 3.93% on both currencies using the same test sets. However, the author assumed unrealistically in the study that there were no transaction costs involved.

Advantage Actor Critic

Two types of Reinforcement Learning algorithms are Actor methods (Policy-based) and Critic methods (Value-based). DQN is a critic-only method in which we approximate the Bellman equation (1) to estimate the value of each state-action pair, which does not optimize a policy for a given state directly. Actor methods estimate the performance of the gradient with respect to the actor's parameters directly through simulation. This introduces high variance in training, as gradients are calculated independently of past training (Konda & Tsitsiklis, 2003).

The Actor-only method uses the policy gradient for weight updates:

$$\nabla_\theta J(\theta) = \ E_{\ \tau}[\textstyle\sum_{t=0}^{T-1} \nabla_\theta \ log\ \pi\theta\ (a_t|s_t)\ G_t] \tag{6}$$

Equation (6) shows that the previous weights have no effect on gradients (Yoon, 2019b).

A combination of both methods – the Actor-Critic method – is proposed to leverage the strength and eliminate the weakness of both methods. Through simulation, the Critic network will learn the value approximation, which is then used to update the actor's policy parameters. This way, the algorithm will learn a policy with low variance, as the weights are updated using the Q function calculated from the Critic network, which relies on previous updates, as shown in equation (3). The Actor-Critic weight update formula (Yoon, 2019b) is given by:

$$\nabla_\theta J(\theta) = E_{\tau}[\textstyle\sum_{t=0}^{T-1} \nabla_\theta \log \pi\theta\ (a_t|s_t)\ Q_\theta(s_t,\ a_t)] \tag{7}$$

Advantage Actor Critic just subtracts from the Q function a state-value function $V(s_t)$. This calculates the advantage of taking action, compared to the average action at a given state s. The formula for advantage value (Yoon, 2019b) is:

$$A(s_t, a_t) = r_{t+1} + \gamma\ V_v(s_{t+1}) - V_v(s_t) \tag{8}$$

The Critic network will then be the V function of the Bellman optimality equation, which represents the value of a given state independent of the action taken (Sutton & Barto, 2018). With the Advantage function, equation (7) will be rewritten as:

$$\nabla_\theta J(\theta) = \textstyle\sum_{t=0}^{T-1} \nabla_\theta \log \pi\theta\ (a_t|s_t)\ A(s_t, a_t) \tag{9}$$

With these simple mathematical tricks, we have the Advantage Actor Critic (A2C) algorithm (Yoon, 2019b).

The study in Yang *et al.* (2020) shows the results of using A2C in automated stock trading. The authors used the algorithm to trade the Dow Jones 30 Constituent Stocks. The data from January 2009 to September 2015 was used for training, October to December 2015 for validation; and January 2016 to May 2020 was used for back-testing. The state space includes data about the stock prices, stock shares, and remaining balance. The authors used the annual return as well as the Sharpe ratio as an evaluation metric that compares the returns on investment with risks. The ratio divides an investment's excess returns by volatility:

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p} \tag{10}$$

R_p represents the return of portfolio, R_f represents risk-free rate, and the denominator represents the standard deviation of the excess return (Fernando, 2024b). The study in Yang *et al.* (2020) achieved an average Sharpe ratio of 0.1878 and an annual return of 11.4%. According to the authors, A2C is more adaptive to risk, especially when the market is bearish.

Recurrent Proximal Policy Optimization

Proximal Policy Optimization (PPO) is another A2C-based algorithm introduced to ensure that the deviation from previous policy updates is relatively small. The policy objective function at equation (9) is sensitive towards the step size and usually leads to destructively large policy updates (Schulman *et al.*, 2017). PPO ensures the policy updates are conservative, by clipping it between $[1-\varepsilon, 1+\varepsilon]$.

The objective function of PPO can be expressed as:

$$Lt(\theta) = \hat{\mathrm{E}}_t\big(min(\, r_t(\theta)\hat{\mathrm{A}}_t \;, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{\mathrm{A}}_t\,)\big) \tag{11}$$

The ratio of the probability of the new policy and previous policy is denoted by r_t. Intuitively, this ratio is used to calculate the divergence between the policy updates. $\hat{\mathrm{A}}_t$ denotes the estimated advantage at the time step, t. The parameter ε is used to control the clipping, usually set to 0.1 or 0.2. The Minimum function ensures the update does not become too large (OpenAI, 2017).

Due to the sequential nature of financial data, this study will use LSTM layers as input to the PPO algorithm; hence the name Recurrent Proximal Policy Optimization (RPPO).

In addition to A2C, the study in Yang *et al.* (2020) also shows the results of using PPO in trading the Dow Jones 30 Constituent Stocks. With PPO, the authors achieved an average Sharpe ratio of 0.1133 and annual return of 15%, which is the highest among all algorithms tested in the study. According to the authors, PPO has the ability to follow trends and acts well in generating more returns. However, the Sharpe ratio is lower than A2C's, meaning PPO could involve more risks.

To summarize this section, all three proposed algorithms have been used by other studies to perform trading. All methods have been proven profitable. However, our study will improve upon these studies by modelling the trading behaviours of real fund managers. For example, Yang *et al.* (2020) only uses the stock price and trading account information in the state space. Our research extends beyond this by incorporating the process of curating cryptocurrencies, feature engineering of technical indicators, such as Relative Strength Index (RSI), and Moving Average Convergence Divergence (MACD and Bollinger Bands) (Chen, 2021; O'Hara, 2023), as well as understanding market sentiment.

Methodology

Data collection

The primary source of data was the finance API provided by Yahoo Finance. This was used to download the historical price data of the cryptocurrencies. The cryptocurrency data downloaded was for BTC-USD, ETH-USD, BNB-USD, DOGE-USD, ADA-USD, MATIC-USD, AVAX-USD and WAVES-USD. The date of the historical data ranged from 28 January 2018 to 20 December 2021 (most recent data as of the time of this study). This is because the cryptocurrency sentiment (Fear and Greed) index data is only available starting from 1 February 2018, and a 4-day window is used as input. The data was saved into .csv files.

Timeseries prediction

We believe that feeding future price predictions to the DRL agent will help it learn faster, as it will not need to learn predicting future prices on its own. Instead, it can focus on learning trading policies (strategies). Therefore, the second phase of this study involves training a predictive model to predict the future closing prices of the cryptocurrencies.

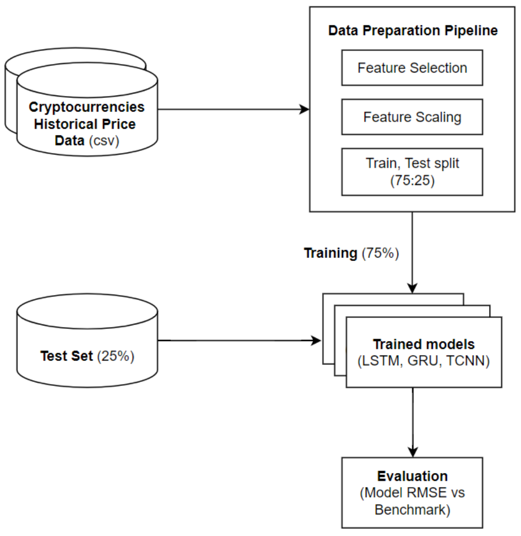

Data Preparation Pipeline
Feature Selection
Feature Scaling
Train, Test split (75:25)
Cryptocurrencies Historical Price Data (csv)
Training (75%)
Test Set (25%)
Trained models (LSTM, GRU, TCNN)
Evaluation (Model RMSE vs Benchmark)

Figure 2. Timeseries prediction flowchart

The data preparation pipeline begins with feature selection. In this phase, we drop the Open, High, and Low columns and only use the Closing Price to do prediction. This is because the other values are highly correlated and are redundant. This will also help prevent overfitting.

The second phase of data preparation is feature scaling. We used the MinMaxScaler from Scikit-learn to scale the values of all cryptocurrencies between 0 and 1. MinMaxScaler largely preserves the relative distance between data points, ensuring the temporal relationships in the data are maintained. This phase ensures the model training is more uniform, as there are no large fluctuations in numbers.

The third phase is train-test split. We split the data using a 75:25 ratio for training and testing, respectively. The predictor variables are the 4-day historical price and the target variable is the actual price of the fifth day. This is the last phase of data preparation.

The next step is to train the prediction models. This study experimented with three different neural network architectures suitable for timeseries applications. They are: Long Short-Term Memory (LSTM); Gated Recurrent Unit (GRU); and Temporal Convolutional Neural Network (TCNN) (Durairaj & Krishna Mohan, 2022; Hamayel & Owda, 2021). To further improve the models' performance, different optimization algorithms, such as Adam, RMSprop, and Stochastic Gradient Descent, were used as well (Fatima, 2020).

Once the models have been trained, they are evaluated with the test set. The prediction is then inverse-scaled for better interpretation of the evaluation results. The metric used is Root Mean Squared Error (RMSE). To make sure the future price prediction is accurate enough to help the DRL agent with prediction, a benchmark RMSE for each cryptocurrency must be achieved, or else the cryptocurrency will be dropped. The benchmark RMSE is finally calculated as:

$$RMSE_{cryptocurrency} < \mu_{closing\ price} * 0.06 \tag{12}$$

The best training results and calculated benchmarks for each cryptocurrency are shown in Table 1.

Table 1. Timeseries prediction benchmarks and results

Cryptocurrency	Best Model	Achieved RMSE	Benchmark RMSE
BTC-USD	TCNN	1688.5353	2052.00
ETH-USD	TCNN	157.5501	146.00
BNB-USD	GRU	22.8944	23.00
DOGE-USD	GRU	0.0087	0.008
ADA-USD	LSTM	0.0722	0.08
MATIC-USD	GRU	0.0975	0.08
AVAX-USD	TCNN	1.5376	3.08
WAVES-USD	GRU	2.0744	0.7

Table 1 shows that only half of the cryptocurrencies exceeded the benchmark given. However, we decided not to drop them yet. To justify this decision, we noted that the cryptocurrencies that achieved an RMSE above the benchmarks only exceeded them by a negligible amount, except WAVES-USD. The second justification is that the benchmark equation (12) is arbitrarily defined. The third justification is that we need more training data for the DRL agents as we only have less than three years of training data for each cryptocurrency.

Web scraping – Cryptocurrency Fear and Greed Index (sentiment analysis)

Market sentiment is another important factor when making trading decisions (Lin *et al.*, 2023). However, it is difficult for us to curate data that accurately represents the market's true sentiment. Limitations, such as data storage, computational power, time constraints and web scraping limitations imposed on different websites, can make it much more difficult to implement sentiment analysis from scratch.

Luckily, the Alternative.me API provides this information. Through the use of the Urllib and BeautifulSoup libraries in Python, we are able to retrieve the cryptocurrency sentiment (Fear and Greed) index calculated daily by Alternative.me.

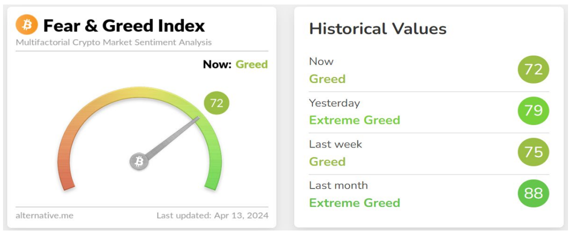


Figure 3. Alternative.me's Fear and Greed Index (Alternative.me, n.d.)

The index ranges from 0 to 100, or "Extreme fear" to "Extreme greed", respectively. The developers of this API ingest data from multiple sources to analyse the emotions and sentiments of Bitcoin and other large cryptocurrencies (Alternative.me, n.d.). The data sources and their weights in the sentiment analysis are shown in Table 2.

Table 2. Fear and Greed index data sources

Data Source	Weight (%)
Volatility	25
Market Momentum/Volume	25
Social Media (Reddit)	15
Surveys	15
Dominance	10
Trends	10

After we have scraped the index, we performed a Pearson's Correlation Coefficient (*r*) analysis to determine whether each of the cryptocurrencies is correlated to this index. Logically speaking, the Fear and Greed Index should be positively correlated with the price of the cryptocurrencies. Therefore, we only select the cryptocurrencies with a positive correlation for DRL training.

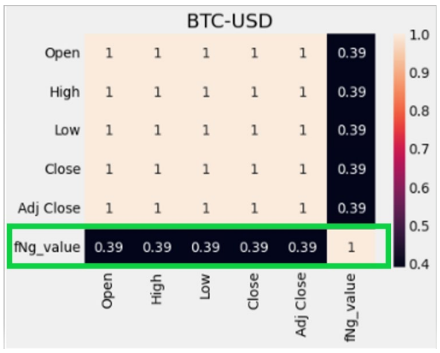


Figure 4. Bitcoin Pearson Correlation Coefficient (*r*) matrix

As expected, Bitcoin has a moderately high positive correlation of 0.39 with the Fear and Greed Index (fNg_value). This means that the index can be used to estimate the future value of Bitcoin. We performed the same analysis for all eight of the cryptocurrencies, using the Adjusted closing price (Table 3).

Table 3. Pearson Correlation Coefficient (*r*) between cryptocurrency prices and Fear and Greed Index

Cryptocurrency	Pearson Correlation Coefficient (r)
BTC-USD	0.39
ETH-USD	0.22
BNB-USD	0.18
DOGE-USD	0.064
ADA-USD	0.19
MATIC-USD	-0.095
AVAX-USD	-0.16
WAVES-USD	0.24

Using the results from Table 3, we decided to drop MATIC and Avalanche (AVAX) from the DRL training set, because they have a negative correlation with the cryptocurrency sentiment (Fear and Greed Index) and likely will not benefit from using it.

Feature engineering

Before going into DRL, there is one extra feature-engineering step to perform, which is computing technical indicators for all cryptocurrencies using their historical price data. The formulas for the technical indicators used are:

Relative Strength Index (RSI):

$$RSI_{step\ one} = 100 - \left[\frac{100}{1+\frac{Average\ gain}{Average\ loss}}\right] \qquad \text{(Fernando, 2024a) (13)}$$

Moving Average Convergence Divergence (MACD):

$$MACD = 12\ period\ EMA - 26\ period\ EMA \qquad \text{(Dolan, 2024) (14)}$$

where EMA is Exponential Moving Average.

Bollinger Bands:

Upper band = 20-day SMA + (20-day σ x 2) (Lund, 2023) (15)

Lower band = 20-day SMA - (20-day σ x 2) (Lund, 2023) (16)

where SMA is Simple Moving Average and σ is variance.

Deep Reinforcement Learning

The core phase of our study is training the DRL agents to learn the optimal trading policy. In this phase, a custom trading environment class is designed to modularize the processes of storing input action, calculating current and future states, and reward calculations, and a terminator to decide when an agent should stop trading. The environment is a core component of the entire Reinforcement Learning training loop as many training components, such as loss function calculation, input spaces, input observation/states, and training termination, all depend on the calculation of the trading environment.

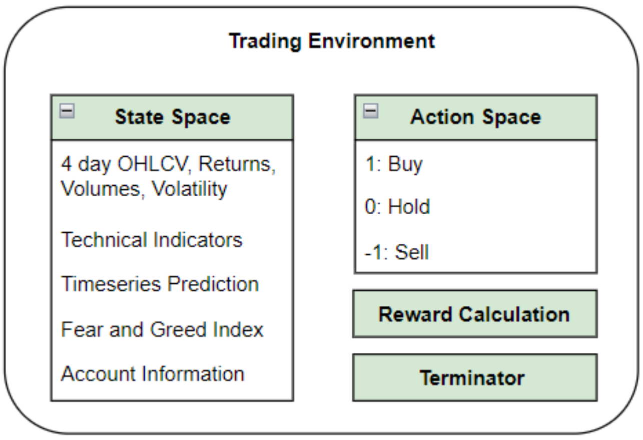


Figure 5. Custom trading environment

The four main components in the trading environment are:

1. State space:
   - 4 days of Open, High, Low, Close, Volume (OHLCV), Returns, Volumes, and Volatility.
   - Technical Indicators: RSI, MACD, Bollinger Bands.
   - Timeseries Prediction.
   - Cryptocurrency sentiment (Fear and Greed) index.
   - Account information: $\frac{Current\ Capital}{Initial\ Capital}$, $\frac{Running\ Capital}{Current\ Capital}$, Asset Units, Previous Action
2. Action space:
   - 1: Buy
   - 0: Hold
   - -1: Sell
3. Terminator / Stopping case:

   If the agent reaches the end step (last row of the dataset).
   If the agent's capital falls below threshold capital (e.g. Threshold of 50% = $500).
4. Reward Calculation:

   The algorithm to calculate reward in our custom-trading environment is outlined in Figure 6.

Algorithm to calculate reward

```
    Input:   $C_t$: Current capital, $C_{t-1}$: Previous capital
             $C_1$: Initial capital
             $A_t$: Current action, $A_{t-1}$: Previous action
             AH: Asset held
             $C_{min}$: Minimum capital required to continue trading
             $Step_{current}$ : Current step
             $E$: End step (last row of data reached)
    Output:  R: The calculated reward
1   R $\leftarrow C_t - C_{t-1}$
2   // Prevent holding too long
    if $A_t = 0$ AND $A_{t-1} = 0$, then R $\leftarrow$ R – 5
3   // Prevent selling nothing
    if $AH = 0$ AND $A_t = -1$, then R $\leftarrow$ R – 50
```

```
4   if $R < 0$, then
        R ←R * 3 // Amplify punishment for losses
6   else if R >= 0, then
        R ←R * 2 // Amplify reward for gains
8   end if
9   // Rewards agent for long term profits.
    if $E$, then R ←R + 10 * ( $\frac{C_t}{C_1}$ – 1)
10  if NOT $E$ AND $C_t < C_{min}$, then
11      R ←R – MAX (0.5, 1 – $\frac{Step_{current}}{Step_{end}}$)
        // capital less than threshold
12  end
```

Figure 6. Reward calculation pseudocode

After the four components have been set up, all that is left is to train the DRL agents. The algorithms used are the aforementioned Duelling DQN, A2C, and Recurrent PPO. During training, the DRL agents interact with our custom trading environment iteratively, learning optimal trading strategies through trial and error. The agents explore different actions based on their current state, S_t, receive feedback in the form of rewards or penalties, R_t, from the environment, and adjust their behaviour over time to maximize profits. This iterative learning process (Figure 7) enables the agents to adapt their trading policies dynamically, leveraging deep learning to identify patterns and optimize trading strategies in the market environment.

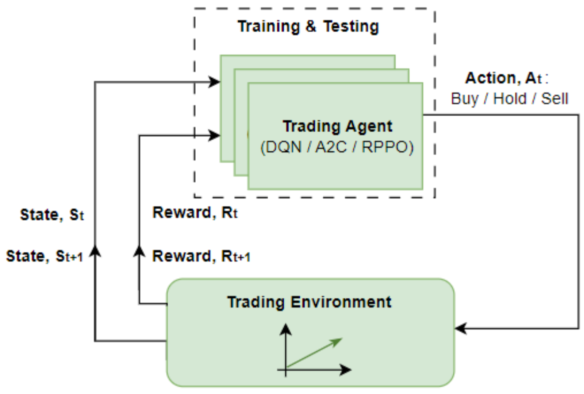


Figure 7. DRL training and testing loop

The models are trained with 60 episodes (equivalent to epochs). In each episode, the models will have traded all 6 cryptocurrencies selected from Table 3 from their earliest date to the latest date in the dataset. GPU acceleration has been enabled for the training via Nvidia's CUDA framework (Pandey *et al.*, 2022). The implementation for DQN and RPPO are in PyTorch, whereas A2C uses Tensorflow.

The important hyperparameters used for the individual models are described in the following subsections.

Duelling Deep Q Network

Table 4. DQN hyperparameters

Hyperparameter	**Selected value**
Epsilon decay	0.9
Epsilon end	0.1
Gamma	0.9995
Tau	0.001
Learning Rate	0.00005
Memory Length	10000
Memory Threshold	500
Optimizer	Adam
Batch Size	200

On top of Duelling DQN, we also implemented memory replay, which randomly samples training data stored in memory buffers. This removes the sequential dependency of the algorithm as well as makes the algorithm more sample efficient as the experiences are used multiple times (Liu & Zou, 2017). Epsilon is a hyperparameter used to allow the DQN agents to perform exploration (Gimelfarb *et al.*, 2020).

In Reinforcement Learning, exploitation means performing the best action every step based on previous learnings, whereas exploration means the agents are allowed to perform suboptimal actions at the given timestep in hope of escaping local minima and finding a better, previously undiscovered policy (Zangirolami & Borrotti, 2024). For our study, we allow the model to perform more exploration in the beginning, and the epsilon value will decrease to 0.1 near the end of the training, signifying a transition from exploration to exploitation, the justification being that the model should have conducted adequate exploration and found an optimal policy at the end.

Advantage Actor Critic

Table 5. A2C hyperparameters

Hyperparameter	**Selected value**
Gamma	0.99
Learning Rate	0.0003
Optimizer	Adam

In policy-based learning algorithms like A2C, we have fewer hyperparameters to tune as exploration is built into the algorithm. This is because the algorithm outputs a probability distribution and then samples from that distribution to decide which action to perform. For this algorithm, we decided not to use memory replay as it has slower computation and tends to learn slower compared to PPO.

Recurrent Proximal Policy Optimization

Table 6. RPPO hyperparameters

Hyperparameter	Selected value
Gamma	0.99
Learning Rate	0.0003
Batch Size	64
Policy Clip	0.2
Optimizer	Adam

Similar to A2C, we do not need an epsilon value for training PPO as it has exploration built in. However, unlike A2C, we can use memory replay for RPPO, albeit with a smaller batch size of 64 compared to DQN, as LSTM layers require more memory. The policy clip, which has been explained in equation (11), will be set to 0.2 to limit the policy updates.

The training and testing have been performed on Nvidia's GTX1050 GPU and Intel's Core i5-8300H CPU with a RAM of 8GB.

Table 7. Training and testing time

Agent	Time taken (HH:MM)
DQN	00:54
A2C	05:37
RPPO	21:15

Table 7 shows that a value-based model such as DQN is much faster compared to A2C and RPPO, which requires two neural networks to be trained. Furthermore, RPPO uses LSTM cells, which further slows down the training and inference.

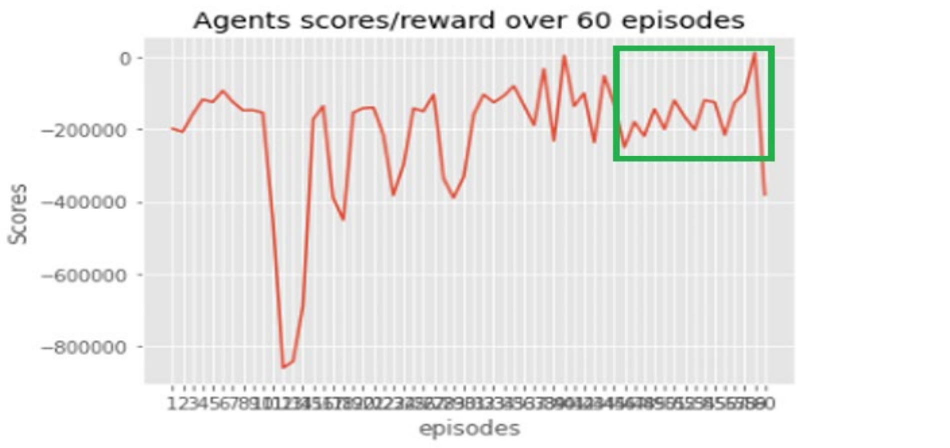
Results and Discussion

Training Rewards

To determine the learning behaviours of the DRL agents during training, we can plot the training scores gained over the 60 training episodes.

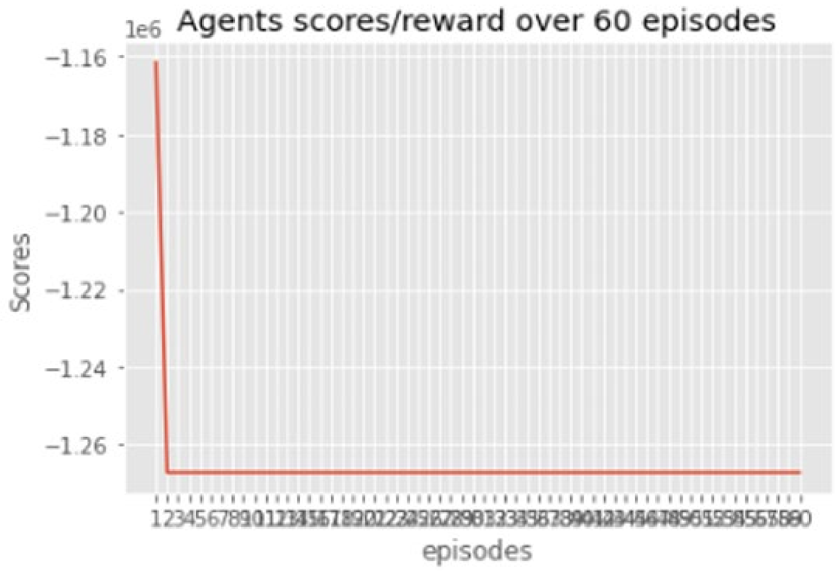
Figures 8–10 show the improvement in training rewards over 60 training episodes. The scale of the rewards in the y-axis is unimportant as there are more punishments in the reward calculation, as shown in Figure 6. Instead, the main purpose of these plots is to show the training of the agents.

DQN has been proven to be effective at learning trading policies. We can see that DQN training rewards fluctuate more during the early episodes. However, when DQN reaches the later episodes (as denoted in the green box of Figure 8), we can see that the training becomes more stable, trending slightly upwards and reaching maximum at episode 59. The model weights at the peak reward will be saved for evaluation.



First episode : -198128.72906135904, Last episode : -380867.9027028481
Increase : -182739.17364148903

Figure 8. DQN training rewards

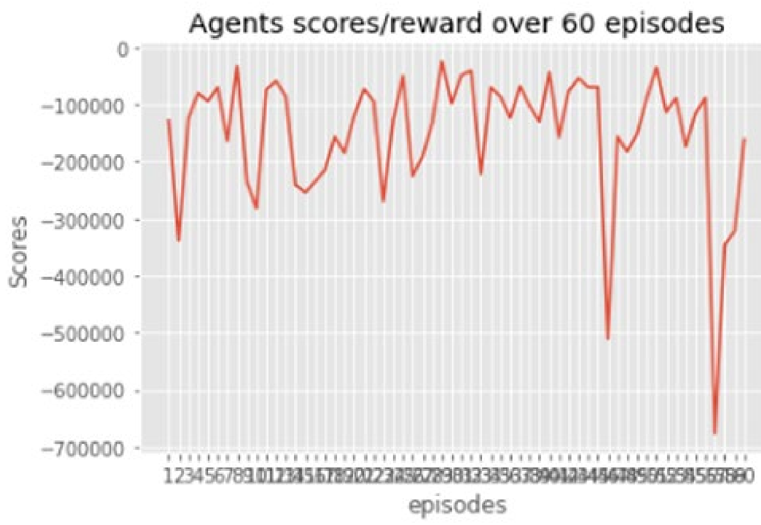


First episode : -1161565.224168371, Last episode : -1267200.0
Increase : -105634.77583162906

Figure 9. A2C training rewards

In contrast, A2C shows no sign of improvement since episode 1 (Figure 9). Several reasons could have led to this phenomenon, such as the lack of experience replay or the high variance in the policy updates.

The RPPO agent shows that the model was trying to learn an optimal policy (Figure 10); however, it did not show much sign of improvement as it fluctuates around the mean. Comparing the results of DQN and RPPO, it seems that DQN performed better during training, reaching positive rewards at the end of the training, whereas RPPO stayed below 0 the whole time.



First episode : -128882.65344901558, Last episode : -161485.1648422925
Increase : -32602.511393276916

Figure 10. RPPO training rewards

Annual Return

As discussed before, the performance metric used to evaluate all the models was the annual return of trading $1000 on the last year of each cryptocurrency (test set). We also compared them to a random agent that took actions randomly. The random agent was expected to make a loss (due to transaction costs) and acted as a baseline to determine if DRL is effective at learning trading.

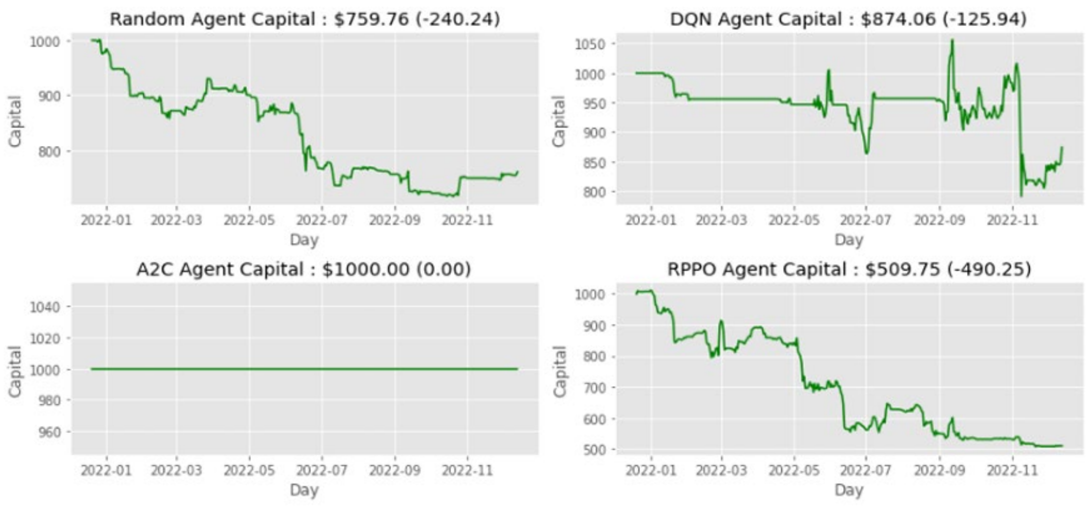


Figure 11. BTC-USD annual return

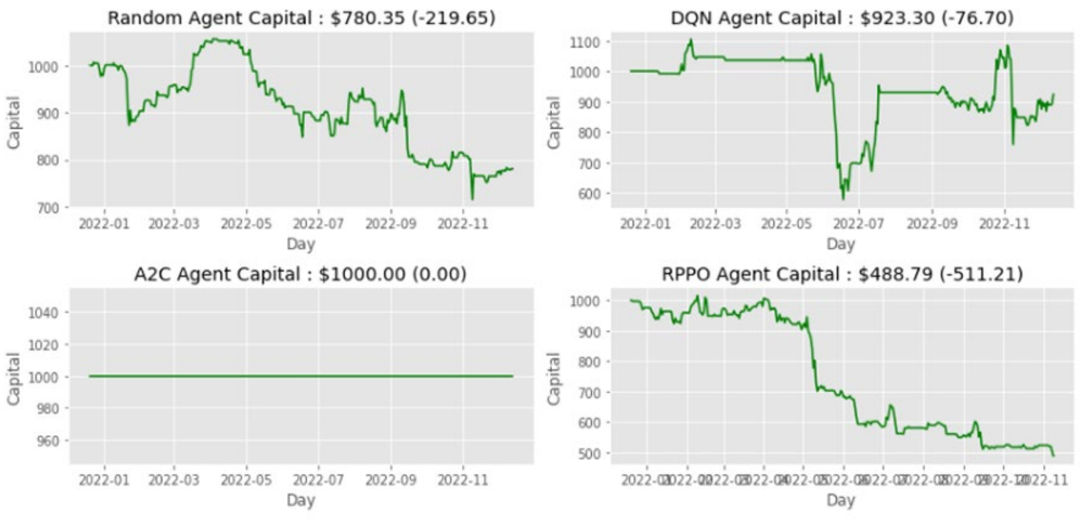


Figure 12. ETH-USD annual return

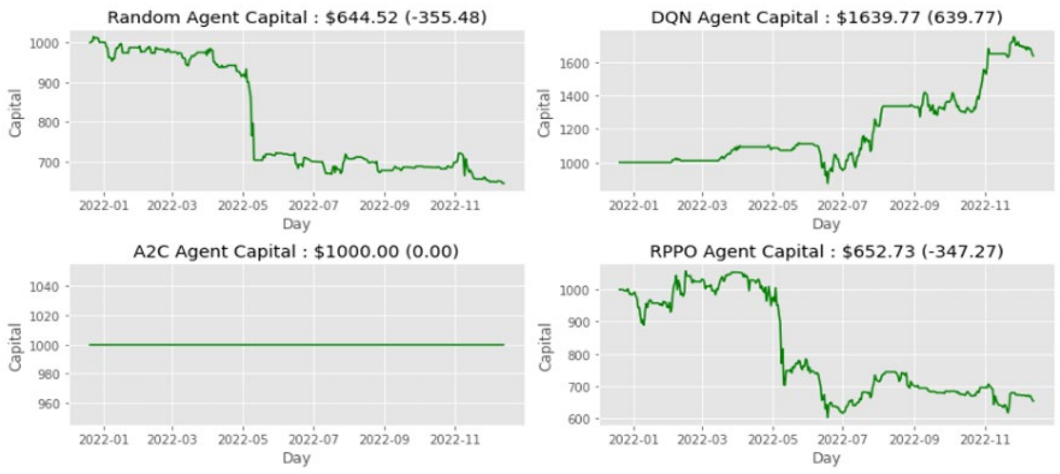


Figure 13. BNB-USD annual return

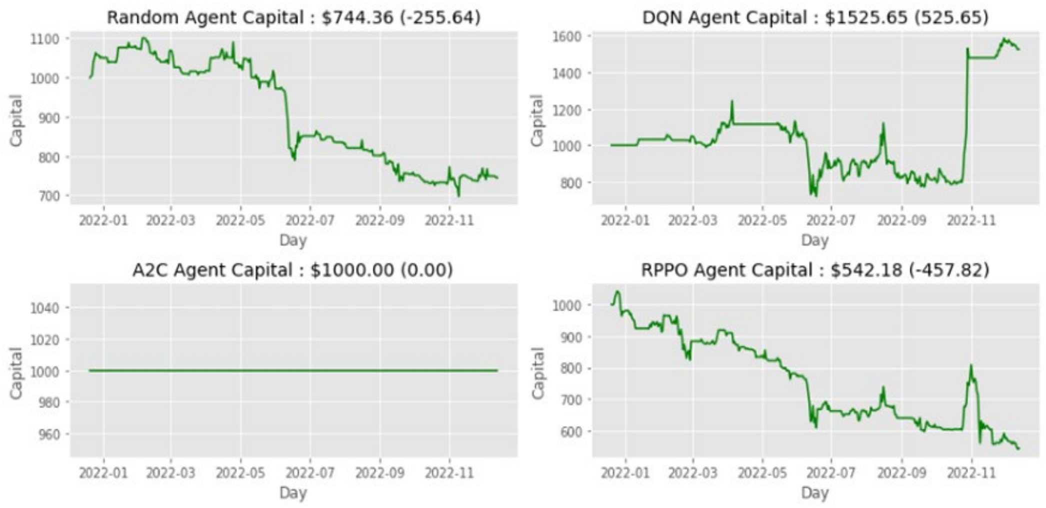


Figure 14. DOGE-USD annual return

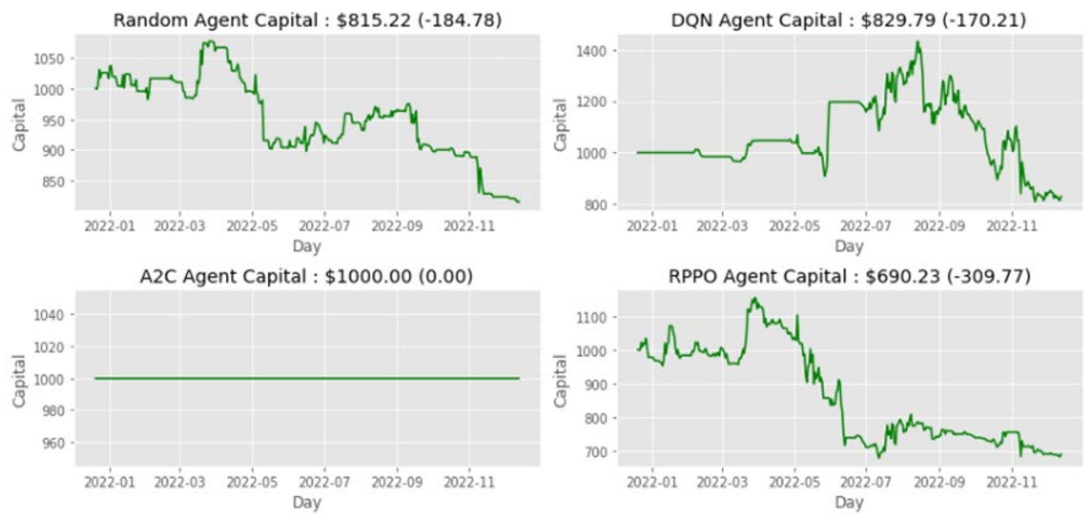


Figure 15. ADA-USD annual return

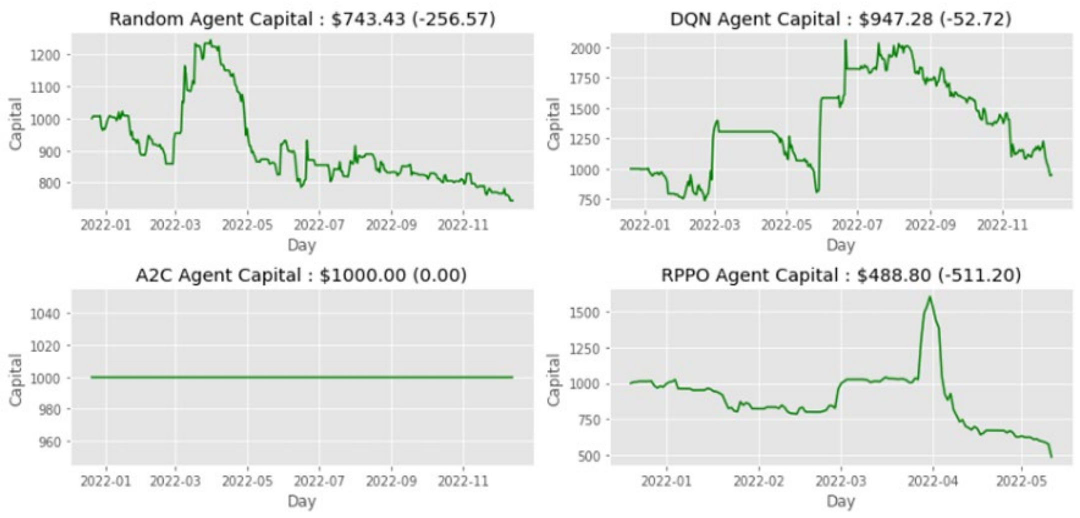


Figure 16. WAVES-USD annual return

Table 8. Summary of annual return achieved

Cryptocurrency	Best Agent	Random Agent's profit	Profit gained	Annual Return
BTC-USD	DQN	-$240.24	-$125.94	-12.59%
ETH-USD	DQN	-$219.65	-$76.70	-7.67%
BNB-USD	DQN	-$355.48	$639.77	63.98%
DOGE-USD	DQN	-$255.64	$525.65	52.57%
ADA-USD	DQN	-$184.78	-$170.21	-17.02%
WAVES-USD	DQN	-$256.57	-$52.72	-5.27%

Table 8 shows that the DQN is the best performing agent for all cryptocurrencies. It also shows that it is better than the random agent, which means it was able to learn some profitable trading policies. The agent profited at trading BinanceCoin and DogeCoin but made a loss when trading the other cryptocurrencies. In BNB-USD, the agent was able to earn the highest return of 63.98%. More importantly, it achieved this return through steady growth of capital rather than winning a few lucky trades, as shown in Figure 13. The average annual return across all cryptocurrencies was 12.33%, which means the agent is profitable overall. If we only trade the cryptocurrencies that the agent is profitable in, we can get an average annual return of 58.27%. This is much better than our initial expectation.

To better visualize the trading decisions made by the DQN agent, we can plot the trading decisions taken by the agent on the most profitable and least profitable cryptocurrencies, which are BNB-USD (Figure 17) and ADA-USD (Figure 18), respectively.

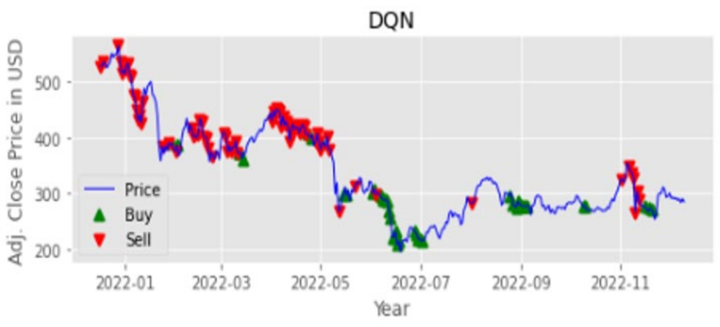


Figure 17. DQN trading decisions on BNB-USD

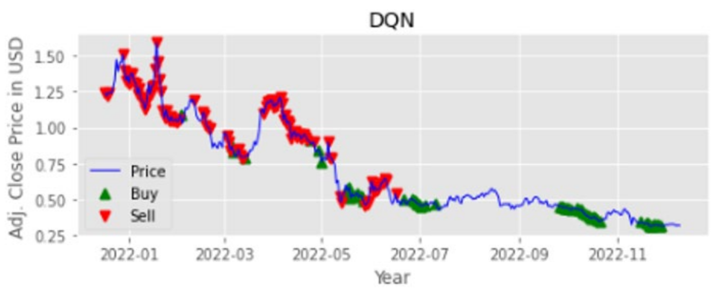


Figure 18. DQN trading decisions on ADA-USD

By analysing Figure 17 and Figure 18, we note that the model was able to capture the trading signals "buy low" and "sell high". The reason ADA-USD appears to have made a loss in Table 8 is because the agent bought low around October 2022. This is most likely due to the anticipation of a retracement. However, the dataset cuts off before we can prove that theory. Furthermore, if we disregard the data before September 2022, we can see that the agent actually profits, as shown in Figure 15.

Conclusion

Based on the results and our analysis, we can conclude that with Deep Reinforcement Learning combined with profound feature engineering, environment setup, and reward design, a profitable trading policy is attainable. The most impressive result is that our best agent, DQN, did not once hit the terminal state (losing more than 50% of the initial capital) for all cryptocurrencies. Cryptocurrencies are volatile in nature and possess a lot of risk. However, our agent was able to control its trading risk and maintain a capital of above $500. The driving factor of this behaviour is attributable to the detailed design of the reward calculation algorithm, which we have experimented with and tuned.

Several technical insights can be gained from this study. Firstly, careful consideration of the reward calculation is an important factor in the success of Deep Reinforcement Learning. Unlike traditional supervised learning, we can creatively design our own reward calculation to cater for our own preferences and needs. Secondly, the utilization of the memory replay in reinforcement learning has shown good results. DQN and RPPO both use replay and were able to learn some trading policies, despite RPPO being ultimately unprofitable. On the other hand, A2C was not able to learn anything, as shown in Figure 9.

This study highlighted the feasibility and potential of Deep Reinforcement Learning in the domain of investing. This provides users the ability to invest without intense study and constant monitoring of the market, since this can be done automatically by our proposed system. Our DQN model achieved a return of 63.98% on BinanceCoin and 12.33% overall, which is able to help investors beat the rising inflation rate of 1.8% (as of February 2024).

Future work can include using the Sharpe ratio for evaluation metrics; carrying out more hyperparameter tuning; using better compute, which allows larger neural network architectures; experimenting with different technical indicators; and trading different asset classes such as stocks. Technical indicators other than the ones used in this work (RSI, MACD, Bollinger Bands) may also be experimented with to determine their effectiveness. Finally, more exploration of the timeseries prediction can be carried out, such as experimenting with different algorithms, features and scaling techniques.

References

Alternative.me. (n.d.). Crypto Fear & Greed Index. *Alternative.me*. Retrieved April 10, 2024, from https://alternative.me/crypto/fear-and-greed-index/

"ASNB declares FY23 dividends". (2023, March 31). *The Star*. https://www.thestar.com.my/business/business-news/2023/03/31/asnb-declares-fy23-dividends

Binance Square. (n.d.). Crypto Fear & Greed Index. *Binance*. Retrieved April 10, 2024, from https://www.binance.com/en/square/fear-and-greed-index

Chen, J. (2021, September 29). Technical Indicator: Definition, Analyst Uses, Types and Examples. *Investopedia*. https://www.investopedia.com/terms/t/technicalindicator.asp

Dolan, B. (2024, March 8). What Is MACD? *Investopedia*. https://www.investopedia.com/terms/m/macd.asp

DOSM. (2024a, January). Consumer Price Index December 2023. Department of Statistics Malaysia. https://www.dosm.gov.my/portal-main/release-content/consumer-price-index-december-2023

DOSM. (2024b, March). Consumer Price Index February 2024. Department of Statistics Malaysia. https://www.dosm.gov.my/portal-main/release-content/consumer-price-index-february-2021

DOSM. (2024c, February). Consumer Price Index January 2024. Department of Statistics Malaysia. https://www.dosm.gov.my/portal-main/release-content/consumer-price-index-january-2024

Durairaj, M., & Krishna Mohan, B. H. (2022). A convolutional neural network based approach for financial time series prediction. *Neural Computing and Applications, 34*, 13319–13337. https://doi.org/10.1007/s00521-022-07143-2

Fatima, N. (2020). Enhancing Performance of a Deep Neural Network: A Comparative Analysis of Optimization Algorithms. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *9*(2), 79–90. https://doi.org/10.14201/ADCAIJ2020927990

Fernando, J. (2024a, April 10). Relative Strength Index (RSI) Indicator Explained With Formula. *Investopedia*. https://www.investopedia.com/terms/r/rsi.asp

Fernando, J. (2024b, January 30). Sharpe Ratio: Definition, Formula, and Examples. *Investopedia*. https://www.investopedia.com/terms/s/sharperatio.asp

Gimelfarb, M., Sanner, S., & Lee, C. G. (2020). Epsilon-BMC: A Bayesian Ensemble Approach to Epsilon-Greedy Exploration in Model-Free Reinforcement Learning. Proceedings of the 35th Uncertainty in Artificial Intelligence Conference, PMLR, *115*, 476–485. https://doi.org/10.48550/arXiv.2007.00869

Hamayel, M. J., & Owda, A. Y. (2021). A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms. *AI*, *2*(4), 477–496. https://doi.org/10.3390/ai2040030

Konda, V. R., & Tsitsiklis, J. N. (2003). Actor-Critic Algorithms. *Neural Information Processing Systems, 42*(4), 1143–1166.

Lin, M., Meng, Y., & Zhu, H. (2023). How connected is the crypto market to risk to investor sentiment? *Finance Research Letters, 56*, 104177. https://doi.org/10.1016/j.frl.2023.104177

Liu, R., & Zou, J. (2017). The Effects of Memory Replay in Reinforcement Learning. *arXiv, 1*(06574). https://doi.org/10.48550/arXiv.1710.06574

Lund, B. (2023, July 11). Bollinger Bands: A powerful technical tool for traders. *Britannica Money*. https://www.britannica.com/money/bollinger-bands-indicator

OpenAI. (2017, July 20). Proximal Policy Optimization. https://openai.com/research/openai-baselines-ppo

Or, B. (2021, January 30). Value-based Methods in Deep Reinforcement Learning. *Towards Data Science*. https://towardsdatascience.com/value-based-methods-in-deep-reinforcement-learning-d40ca1086e1

O'Hara, N. (2023, August 31). The Multiple Strategies of Hedge Funds. *Investopedia*. https://www.investopedia.com/articles/investing/111313/multiple-strategies-hedge-funds.asp

Pandey, M., Fernandez, M., Gentile, F., Isayev, O., Trospha, A., & Stern, A. C. (2022). The transformational role of GPU computing and deep learning in drug discovery. *Nature Machine Intelligence, 4*, 211–221. http://dx.doi.org/10.1038/s42256-022-00463-x

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv, 2*(06347). https://doi.org/10.48550/arXiv.1707.06347

Sornmayura, S. (2019). Robust FOREX Trading with Deep Q Network (DQN). *ABAC Journal, 39*(1), 15–33. https://core.ac.uk/download/pdf/233618241.pdf

Sutton, R. S., & Barto, A. B. (2018). *Reinforcement Learning: An introduction* (2nd ed.). London, England: The MIT Press.

Tradingview. (n.d.). BTCUSD chart. *Tradingview*. Retrieved April 10, 2024, from https://www.tradingview.com/chart/cZxzp0Jc/?symbol=BITSTAMP%3ABTCUSD

VethaSalam, R., & Ibrahim, J. (2024, March 3). EPF declares a 5.5% dividend for conventional savings for 2023. *The Star*. https://www.thestar.com.my/news/nation/2024/03/03/epf-declares-55-dividend-for-conventional-savings-for-2023

Yang, H., Lim, X. Y., Zhong, S., & Walid, A. (2020). Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. In ACM International Conference on AI in Finance, *31*, 1–8. https://dx.doi.org/10.2139/ssrn.3690996

Yoon, C. (2019a, October 20). Dueling Deep Q Networks. *Towards Data Science*. https://towardsdatascience.com/dueling-deep-q-networks-81ffab672751

Yoon, C. (2019b, February 6). Understanding Actor Critic Methods and A2C. *Towards Data Science*. https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f

Zangirolami, V. & Borrotti, M. (2024). Dealing with uncertainty: Balancing exploration and exploitation in deep recurrent reinforcement learning. *Knowledge-Based Systems, 293*, 111663. https://doi.org/10.1016/j.knosys.2024.111663