Cryptocurrency Trading with (Double) Deep Q-Networks

Nancy Jiang

Computer Science, Stanford University
yjiang26@stanford.edu

March 15, 2024

1 Introduction

The financial industry dedicates significant resources to studying stock market trends using sophisticated statistical models. However, due to their extreme volatility and a lack of comprehensive fundamental data, cryptocurrency presents a distinctive challenge for traditional quantitative trading methods. Nonetheless, cryptocurrency's accessibility and flexible trading hours make it an attractive alternative to traditional stocks for retail traders. As such, our project aimed to develop an effective systematic trading model tailored to the unique nature of cryptocurrency, with the goal of achieving consistent excess returns for potential real-world deployment. We applied our model to two popular cryptocurrencies, namely Bitcoin (BTC) and Ethereum (ETH), while using stablecoin Tether (USDT) for benchmark purposes.

Reinforcement learning models have demonstrated effectiveness in stock market prediction through reward-driven decision making mechanisms [2][10]; meanwhile, neural networks are known for their capacity to discern patterns from high dimensional data such as the technical indicators utilized in algorithmic trading [6][9]. Given its prior success in traditional stock trading, we hypothesized that the combination of such techniques would also prove effective for cryptocurrency trading. Consequently, we employed two classical deep reinforcement learning models, namely Deep Q-Learning and Double Deep Q-learning, for our objective of systematic cryptocurrency trading. Our approach relied on a two-layer neural network with 256 units per layer, yielding an average validation return of 70% for both Bitcoin (BTC) and Ethereum (ETH).

1.1 Related Work

Traditional machine learning models such as linear regression, random forest, and support vector machines (SVM) have been recognized for their effectiveness in predicting cryptocurrency and stock market prices. Sebastião et al. investigated the performance of these models as trading agents, achieving an annualized return of under 10% on Bitcoin and Ethereum [8].

Recent years have seen a growing interest in the potential of artificial neural networks and reinforcement learning models in stock price prediction and trading environments. Brim utilized a Double Deep Q-Network to engage in regular stock trading, achieving a cumulative return of 131.33% across 38 stocks [1]. More recently, in 2024, Otabek and Choi proposed a multi-level deep Q-network (MDQN) model to incorporate bitcoin historical price data and sentiment analysis from Twitter, a key indicator for Bitcoin price prediction. Their model achieved an excess return of 29.93% [7].

The aforementioned existing literature provided compelling evidence for the feasibility of our approach and supported our confidence in the results obtained through our training process.

2 Dataset

We obtained historical data for three cryptocurrencies (Bitcoin (`ticker='BTC'`), Ethereum (`ticker='ETH'`) and Tether (`ticker ='USDT'`) from Yahoo! Finance. Given the short history of cryptocurrency, we utilized daily high, low, closing prices, and trade volume from January 1, 2016 to December 31, 2022 for Bitcoin, and all available historical data from November 2017 to December 31, 2022 for Ethereum and Tether. We used the most recent annual data (January 1, 2023 to December 31, 2023) as the validation dataset for all three cryptocurrencies.

The feature space consists of the percentage change in closing price over $1, 7, 14$-day periods (representing next-day, week, and two-week returns), along with six additional technical indicators: Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Average True Range (ATR), On-Balance Volume (OBV), Bollinger Bands (BBands), and Stochastic Oscillator. We relied on the library `TA-lib` for computing technical indicators and scaled the results for normalization purposes. It is noteworthy that given the high volatility of cryptocurrencies, Bollinger Bands, which identifies market volatility and short-term price breakouts, are an especially crucial indicator in the context of cryptocurrency trading.

3 Methods

3.1 Environment

The trading simulator class (`TradingSimulator`) emulates the process of cryptocurrency trading over the course of one day, with each episode representing trading process over the span of one year or 365 days. The discrete action space consists of three actions: buy (`action=0`), hold (`action=1`), and sell (`action=2`), where the agent's positions are determined by subtracting 1 from these values. We implemented a reward function commonly used in trading environments, computed as follows:

$$r(t) \;=\; (a(t)-1)\cdot \texttt{market\_return} - (a(t)-a(t-1))\cdot \texttt{total\_transaction\_cost},$$

where $a(t)$ is the action the agent takes at step t, and thus $a(t)-1 \in \{-1, 0, 1\}$ represents the agent's current position. The reward is thus computed as the agent's position multiplied by the future market return minus the total time and transaction cost the action incurs.

At the beginning of an episode, the agent is given 1 unit of cash capital. We asked the agent to always invest a fixed proportion (`buy_frac=0.5`) of its total cash when taking the buy action, and always sells a fixed proportion (`sell_frac=0.5`) of its total shares of the cryptocurrency when taking the sell action. The agent's total (excess) return at the end of the 365 trading days is calculated as the sum of its capital in both cash and shares of the asset, subtracted by its initial 1 unit cash capital.

3.2 Agents

3.2.1 Deep Q-Learning

Deep Q-Learning is a model-free reinforcement learning algorithm ideal for finite sequential decision making. It aims to learn an optimal policy that maximizes expected total rewards across future actions and states. At each step, the agent selects an action $a(t) \in \{0, 1, 2\}$ based on its observation of the current state $s(t)$, and receives a reward $r(t)$ from the environment. The Q in Q-learning refers to the optimal action-value function $Q^*(s(t), a(t))$, defined as the maximum expected return achievable by following any strategy after taking action $a(t)$ at state $s(t)$. In traditional Q-learning, a Q-table stores values for state-action pairs, which is impractical for large state-spaces. Deep Q-Network (DQN) extends this idea with a Convolutional Neural Network (Q-Network) to estimate Q-values for all actions given a state. The Q-value for a state-action pair is the sum of the immediate reward and a fraction ($\gamma =$ discounted factor) of the maximum estimated future reward:

$$Q^*(s(t), a(t) \;=\; r(t) + \gamma \cdot Q(s(t+1), \arg\max_{a'} Q(s(t), a(t)).$$

3.2.2 Double Deep Q-Learning

The Double Deep Q-Learning algorithm is similar to Deep Q-Learning with one key difference: it employs two Q-networks of the same structure: an online Q-Network and a target Q-Network. While the online network selects action, the Q-values it maximizes are updated using the target network, which is periodically updated to enhance stability and reduce network correlation. In Double Deep Q-Learning, Q-values are approximated using the maximum estimated reward given by the target Q-network for the next action given by the online network:

$$Q^*(s(t), a(t)) \approx r(t) + \gamma \cdot Q_{\text{online}}(s(t+1), \arg\max_{a'} Q_{\text{target}}(s(t), a(t)).$$

In both DQN and DDQN models, we employed Q-Networks consisting of two dense layers with 256 units each, ReLU activation functions, and a dropout layer with a dropout rate of 0.1. We used the mean square error (MSE) loss function and the Adam optimizer with a 0.0001 learning rate to update the networks.

3.3 training

The agents balance exploration and exploitation using an epsilon–greedy policy during training. Starting with a value of 1.0, epsilon gradually decreases over 250 linear steps to 0.1, followed by $365 - 250 = 115$ exponential decay steps with a rate of 0.99.

State transitions `(this_state, action, reward, next_state)` are stored in the agent's memory buffer for experience replay; a minibatch of transitions from the buffer is fed to the Q-network to prevent overfitting.

Model profitability is represented by the excess return generated at the end of each of the 300 training episode. Models are subsequently tested on 365 consecutive days of unseen data for validation.

4 Experiments

4.1 Results

We tracked the 10-episode average training loss, 50-episode average training excess return, as well as the 10-day average validation excess return for Bitcoin (BTC), Ethereum (ETH), and Tether (USDT). In Table 1 we provide the final training loss, training excess return, and validation excess return at the end of the 365 validation trading days. We plotted the validation price data and labeled each data point with the action taken on that day during evaluation (Figure 2). Additional validation data such as the total asset breakdown, transaction cost, and actions are available in the project repository.

Table 1: Training and Validation Results

Cryptocurrency	Training Loss		Training Excess Return		Validation Excess Return	
	DQN	DDQN	DQN	DDQN	DQN	DDQN
BTC	0.0004189	0.0005155	1.4009	1.9294	0.6242	0.7690
ETH	0.0007309	0.0006066	0.7466	0.5399	0.6113	0.7576
USDT	6.8048	5.2643	-0.0069	-0.0169	-0.0060	-0.0278

4.2 Discussion

4.2.1 Hyperparameters

Provided the size of our training datasets and action space, we opted for a relatively simple two-layer neural-network with 256 units per layer. We also used a batch-size of 32 during experience replay to balance computational efficiency and learning performance. To ensure convergence and prevent overreaction to noise,

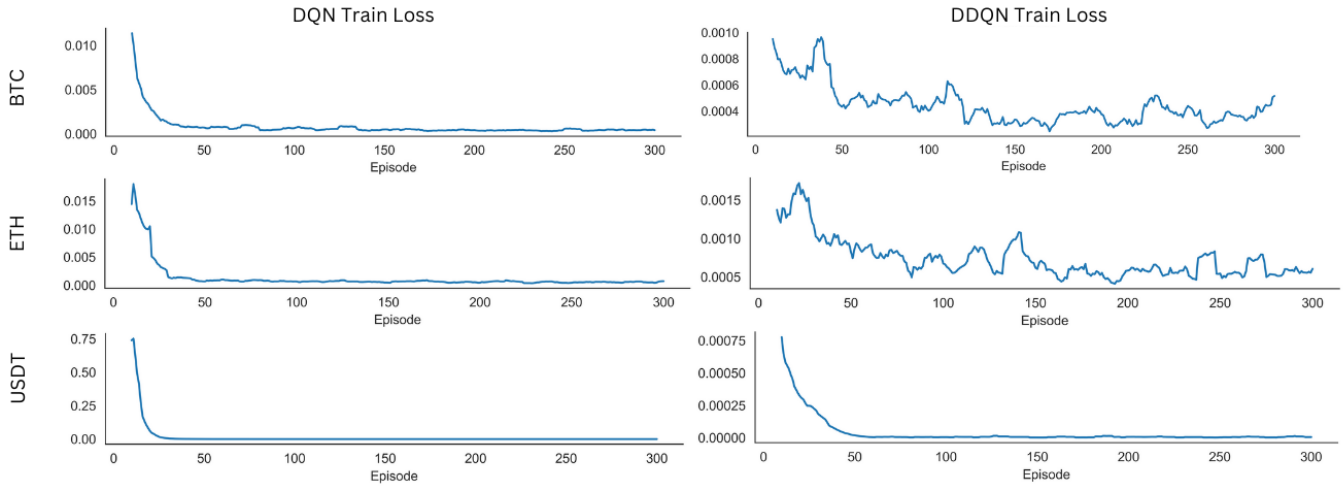

Figure 1: Train Loss

we set a learning rate of 0.0001. To address overfitting tendencies observed after 300 episodes, we reduced the total training episodes from 500 to 300 and adjusted epsilon values and its rate of decay accordingly. We used a target network update frequency of 100 steps.

4.2.2 Loss Analysis

Compared to the DQN agent, the DDQN agent exhibited a significantly more turbulent training loss trajectory. This phenomenon can be attributed to the delayed and periodic updates of the target network, which led to discrepancies between the online and target networks and potentially resulted in higher variability in the agent's training loss.

4.2.3 Agent Performance

Both Deep Q-Learning and Double Deep Q-Learning agents demonstrated significant learning outcomes in Bitcoin and Ethereum trading. The DQN agent achieved validation excess returns of 66.69% and 64.21% respectively, and DDQN agent achieved returns of 78.03% and 79.46% respectively (Table 1). The lack of significant profitability in Tether trading was well within our expectation given its stablecoin nature (i.e., value tied to US Dollars); we thus used Tether as a sanity check for the model learning process.

Evaluating agent performances based on the final excess return, we observed that DDQN agent was able to profit more consistently, notably outperforming DQN on Bitcoin and Ethereum. Since DDQN uses two asynchronous networks—the online network to selection action for the next state and the target network to update the target Q-value of taking that action—the action selected for the next state is not necessarily the "best" action, or the action with the highest Q-value. In contrast, DQN calculates target Q-values with the same network and thus always chooses the action with the highest Q-value. Since a high current Q-value does not guarantee the optimality of the action, DDQN has the advantage of reducing overestimation of Q-values, which tends to result in a faster, more stable learning process.

This difference can be effectively visualized in Figure 2, which captures the action taken by DQN and DDQN agents on each trading day during evaluation. Green, Blue, and Red markers correspond to buy, hold, and sell actions respectively. While both agents demonstrated the ability to buy low and sell at notable peaks and during periods of consistent upward growth, the DDQN agent acquired additional behaviors: notably, it learned to hold during price plateaus (07/2023–09/2023, BTC), and to sell aggressively before a plummet (08/2023–11/2023, ETH).

It was surprising to note that the DDQN agent displayed a stronger tendency to hold, contrary to our initial expectation of more aggressive capitalization due to the additional adaptability provided by the asynchronous Q-networks. One plausible explanation for DDQN Agent's conservative behavior is that while the

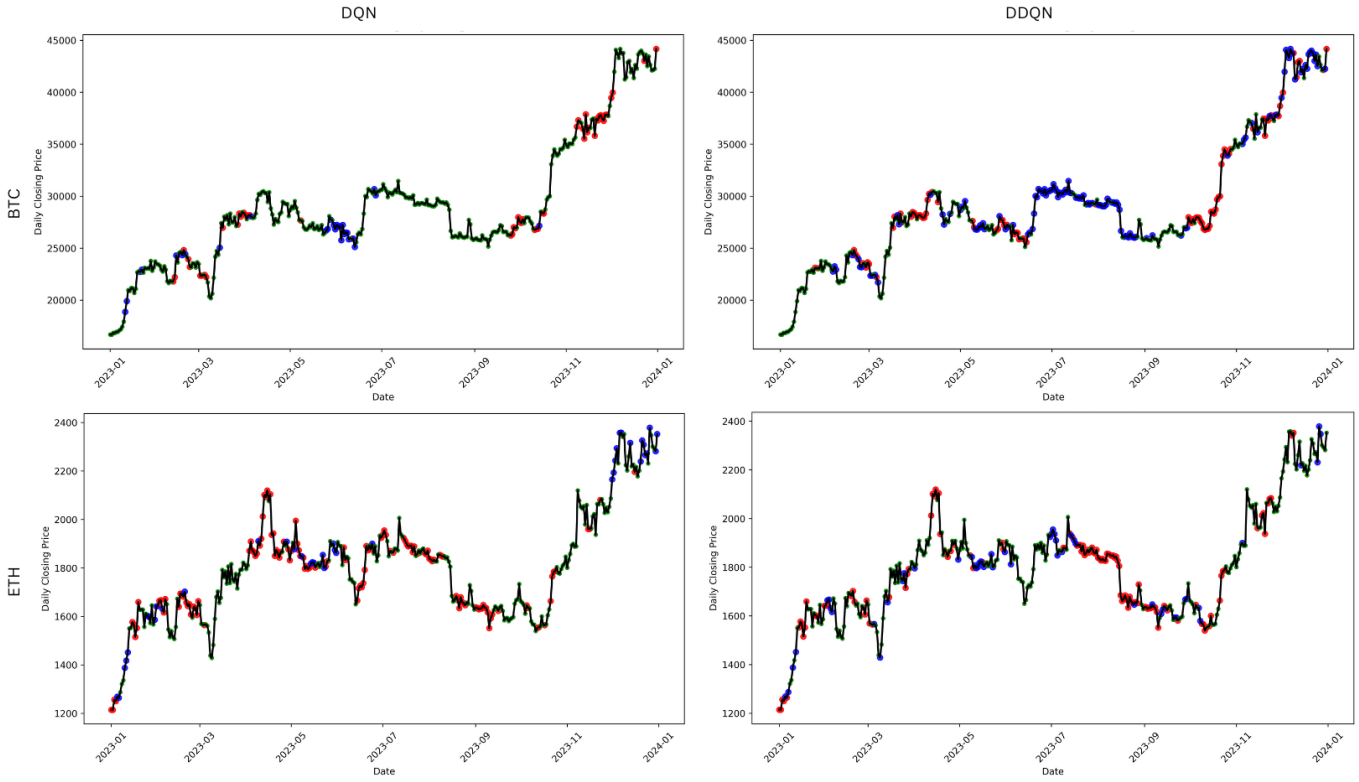

Figure 2: Action Taken Each Trading Day during Evaluation

doubled network effectively avoids overestimation, it also tends to exercise caution towards overly optimistic decisions, which in turn promotes stability by reducing the variance in action–value estimates, resulting in a slightly less variable action sequence. Consequently, during sudden market downturns such as the Ethereum price plummet around 03/2023, the DDQN agent opted to hold instead of buy, diverging from the aggressive buying actions taken by the DQN agent.

5 Conclusion and Future Work

To conclude, in this project we investigated the efficacy of (Double) Deep Q-Network models as agents for cryptocurrency trading in a simulated trading environment. Our agents demonstrated significant learning outcomes, with the DDQN agent displaying higher learning stability and asset profitability than the DQN agent during validation. Therefore, overall we are very satisfied with our training results.

With more time, we'd like to incorporate sentiment analysis, a known Bitcoin price indicator, during feature engineering. We would benchmark our agent performance against more traditional machine learning models such as MDP and SVM. We'd also like to compare our results with other deep reinforcement learning algorithms such as deep long short-term memory (LSTM) networks and deep deterministic policy gradient (DDPG).

Furthermore, the training loss instability in the DDQN agents prompts us to explore alternative loss functions to better capture training progressions. Meanwhile, as the DDQN agent may not consistently outperform the DQN agent, we'd like to identify potential avenues for either agent to outperform the other in order to gain more insights into their respective behaviors.

References

[1] A. Brim, "Deep Reinforcement Learning Pairs Trading with a Double Deep Q-Network", 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, pp. 222-227 (2020).

[2] Chong E., Han, C., Park, F., "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies", *Expert Systems with Applications*, **83** pp. 187–205 (2017).

[3] Hasselt, H., Guez, A., Silver, D., "Deep reinforcement learning with double Q–Learning", Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 2094–2100, (2016).

[4] Huang, J., Huang, W., Ni, J., "Predicting bitcoin returns using high-dimensional technical indicator", *The Journal of Finance and Data Science*, **5**.3, pp. 140–155, (2019).

[5] Jensen, S., *Machine Learning for Algorithmic Trading: Predictive models to extract signals from market and alternative data for systematic trading strategies with Python*, Packt Publishing, 2nd ed, (2020).

[6] Khirbat, G., Gupta, R., and Singh, S. "Optimal neural network architecture for stock market forcasting", 2013 International Conference on Communication Systems and Network Technologies, Gwalior, India, pp. 557–561, (2013).

[7] Otabek, S., Choi, J., "Multi-level deep Q-networks for Bitcoin trading strategies", *Sci Rep*, **14**, 771 (2024).

[8] Sebastião, H., Godinho, P., "Forecasting and trading cryptocurrencies with machine learning under changing market conditions", *Financ Innov*, **7**, 3 (2021).

[9] Shen, J. Shafiq, M.O., "Short-term stock market price trend prediction using a comprehensive deep learning syste", *J Big Data*, **7**, 66 (2020).

[10] Yong Shi, Wei Li, Luyao Zhu, Kun Guo, Erik Cambria, "Stock trading rule discovery with double deep Q-network", *Applied Soft Computing*, **107** (2021).