



# Washington University in St. Louis

---

## OLIN BUSINESS SCHOOL

**Deep Reinforcement Learning with Applications in Business**

**Final Project**

**Reinforcement Learning Approach**

**to Maximization of Risk-adjusted Returns in Ethereum Investment**

**Project Group 11**

Bosan Hsu (MSBA)

Young Kim (MSBA)

December 14, 2024

Presentation recording:

[https://wustl.zoom.us/rec/share/d\\_CXymiBmP1nUr7i4aTV3MYg8dhhcoEegi7Zw87jhxgU32VHQsPI2pySM2Uj96qW.urjcdd0b51zmm1dM?startTime=1734188876000](https://wustl.zoom.us/rec/share/d_CXymiBmP1nUr7i4aTV3MYg8dhhcoEegi7Zw87jhxgU32VHQsPI2pySM2Uj96qW.urjcdd0b51zmm1dM?startTime=1734188876000)

## Table of Contents

Abstract.....	3
1. Introduction.....	3
2. Literature Review.....	3
2.1. Deep Reinforcement Learning for Bitcoin Trading.....	3
2.2. A Comparative Analysis of RL Approaches to Cryptocurrency Price Prediction.....	4
2.3. Reinforcement Learning with Self-Attention Networks for Cryptocurrency Trading .....	4
3. Problem Description .....	5
3.1. Literature Review Summary & Superiority of Our Method.....	5
3.2. Our Approach .....	5
4. Dataset Background and Data Preprocessing .....	6
5. Model Comparison.....	7
<b>5.1. Deep Q-Learning Network (DQN)</b> .....	7
<b>5.2. Proximal Policy Optimization (PPO)</b> .....	8
6. Conclusion, Discussion and Future Work.....	10
References.....	10
Appendix.....	12

## Abstract

Ethereum, a blockchain platform supporting smart contracts and decentralized applications (DApps), has broad real-life applications but suffers from high price volatility, deterring direct investment. This research explores the use of reinforcement learning (RL) to mitigate Ethereum's volatility, aiming to develop an RL-based algorithmic trading model that maximizes risk-adjusted returns using the Sharpe Ratio as a reward function. Three RL approaches—Deep Q-Learning Network (DQN), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C)—are analyzed, focusing on their ability to manage risk while generating sustainable returns in Ethereum trading.

The study incorporates daily Ethereum trading data (2018–2022) with engineered features such as logarithmic returns, moving averages, and volatility measures. The DQN model outperformed PPO and A2C, with the Non-Normalized DQN achieving the highest portfolio return (180.47%) but exhibiting higher volatility, while the Normalized DQN demonstrated balanced performance, prioritizing risk-adjusted returns (38.78%). Both PPO and A2C yielded suboptimal results due to noisy rewards, limited exploration, and hyperparameter tuning challenges.

Future enhancements include advanced feature engineering with technical indicators, hyperparameter optimization using Bayesian methods, and hybrid RL models combining value-based and policy-based approaches. This research highlights the potential of RL in algorithmic trading for volatile assets like Ethereum, paving the way for more stable and profitable investment strategies in cryptocurrency markets.

## 1. Introduction

Ethereum is a blockchain platform capable of implementing smart contracts and decentralized applications (DApps), with applications spanning various fields in real life. However, due to its high price volatility, skepticism remains around direct investment in Ethereum.

If algorithmic trading could offset such high volatility, Ethereum investments could become more attractive, accelerating the expansion of Ethereum-based technologies. Reinforcement learning (RL), widely used in algorithmic trading, has been successfully applied in stock and forex markets. However, in the context of cryptocurrencies, automatic trading algorithms have seen limited adoption by traditional financial institutions, apart from a few exchanges and startups.

The goal of this research is to construct an optimal RL-based algorithmic trading model that maximizes risk-adjusted returns, rather than merely maximizing profits. To achieve this, we will use the **Sharpe Ratio** as a reward function, balancing returns against asset volatility.

Our primary algorithm will be the **Deep Q-Learning Network (DQN)**, but we will also compare its performance with other models such as **Proximal Policy Optimization (PPO)** and **Advantage Actor-Critic (A2C)**. Additionally, since algorithmic trading involves significantly higher trading frequency than human-driven trading, we must account for transaction costs. Thus, the research will focus on optimizing cumulative long-term rewards adjusted for trading costs.

## 2. Literature Review

### 2.1. Deep Reinforcement Learning for Bitcoin Trading

This paper investigates the use of Deep Reinforcement Learning (DRL) for Bitcoin trading to address challenges associated with volatile asset price movements. The proposed approach combines Dueling Double Deep Q-Learning Networks (DD-DQNs), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C) to create an optimal trading policy. The Sharpe ratio and profit reward functions are

employed to evaluate performance.

More specifically, DRL means the integration of Reinforcement Learning (RL) with Deep Learning (DL) techniques. Their methodology models the trading environment as a Markov Decision Process (MDP) to handle sequential decision-making under uncertainty. In their models, DD-DQNs estimate state-action value functions and leverage convolutional layers for feature extraction. PPO optimizes policy gradients with clipped objective functions to improve sample efficiency. Finally, A2C calculates advantages and refines policy and value functions simultaneously.

Their combined strategy of DD-DQNs, PPO, and A2C outperformed each individual approach. During the three years from 2018 to 2020, the Sharpe ratio of the combined model is 1.41, which surpasses the result of individual models (DD-DQNs: 1.39, PPO: 1.36, A2C: 1.25). The average return (profit reward function) is 12.84%.

They suggested a good performance indicator such as the Sharpe ratio considering a high volatility in the cryptocurrency market. Their approach's modular nature allows for extension to other financial assets and market environments.

## 2.2. A Comparative Analysis of RL Approaches to Cryptocurrency Price Prediction

This study compares various Reinforcement Learning (RL) techniques for cryptocurrency price prediction, focusing on Bitcoin, Ethereum, and Cardano. It examines value-based and policy-based RL methods and evaluates them on their ability to generate profitable trading strategies in volatile markets. Like in the first literature, the authors compared three algorithms, but they used a slightly different value-based model. Instead of DD-DQNs, they used Deep Recurrent Q-Network (DRQN), which combines Deep Q-Learning and Recurrent Neural Networks. According to their analysis, DRQN demonstrated the best performance with a 50% profit during training, far outperforming the other two models (2% profit).

The researchers also scaled their methodologies with regards to reward functions and financial markets. First, they implemented other reward functions such as Sharpe ratio and Sortino ratio to mitigate the risky and unreliable profit. Based on the analysis, DRQN with the Sharpe ratio produced the most consistent results. Second, they applied DRQN to Ethereum and Cardano markets and found a good performance in Ethereum, achieving 3% profit in eight hours.

In conclusion, the study highlights the Deep Recurrent Q-Network with the Shape ratio reward function as the most effective approach for cryptocurrency price prediction. DRQN leverages the sequential learning capabilities of Recurrent Neural Networks, enhancing its ability to handle volatile and dynamic markets like cryptocurrencies. The model also exhibited good generalization, performing well in structurally similar markets like Ethereum, indicating its scalability.

## 2.3. Reinforcement Learning with Self-Attention Networks for Cryptocurrency Trading

Research introduces a deep reinforcement learning system leveraging self-attention network (SA-NET) for cryptocurrency trading. The system is designed to process data from a large number of cryptocurrency assets, enabling dynamic investment strategies. Self-attention layers allow the model to focus on relationships between multiple assets, identifying relevant features in volatile cryptocurrency markets. The network dynamically adapts to changes in market conditions and asset availability.

One interesting point in the study is that researchers incorporate transaction fee optimization through specific algorithms to reduce operational costs, critical for high-frequency trading scenarios. Experiments show the SA-NET significantly outperforms a baseline method in both short-term (4-day) and long-term (1-year) trading setups, achieving higher profitability and robustness. SA-NET achieves

an average daily profit of 4.3% compared to 3.4% for models without transaction fee consideration and 2.24% for the baseline.

The approach of the study has some advantages. First, the SA-NET model achieves higher profits across multiple scenarios due to its ability to analyze inter-asset relationships effectively. Second, the inclusion of self-attention layers enables the model to adapt to new assets and changing market conditions dynamically. Lastly, the transaction cost optimization ensures higher net returns.

### 3. Problem Description

#### 3.1. Literature Review Summary & Superiority of Our Method

We can notice the scalability of RL models in financial markets. Mostly, previous researchers tried to maximize estimated profits from cryptocurrency investment across various cryptocurrency markets. Ethereum can be assumed as the second largest cryptocurrency, and also has its relatively high predictability based on the huge market capitalization.

Although Ethereum is applied in numerous fields in real life, it has been less popular for RL-based research. However, unlike Bitcoin, Ethereum’s ability to execute decentralized applications (DApps) and smart contracts has led to transformative changes across industries, promoting transparency, efficiency, and decentralization. For example, Ethereum reduces transaction costs, speeds up financial services without centralized intermediaries. Ethereum also automates property sales and rentals based on its smart contract in the real estate industry. As its ecosystem grows, its real-life applications will continue to expand, making it a fundamental pillar of the whole commercial contracts.

We choose Ethereum as this paper’s main financial market to expand the research field related to reinforcement learning for the cryptocurrency market leveraging. The research on Ethereum can confirm the scalability of reinforcement learning techniques.

#### 3.2. Our Approach

Our goal is to build autonomous algorithmic trading based on Reinforcement Learning (RL). RL-based algorithmic trading offers several advantages over manual trading, primarily due to its ability to leverage data-driven decision-making, scalability, and adaptability in volatile markets. First, RL-based trading makes objective decisions based on learned patterns, statistical analysis, and optimization, minimizing emotional biases such as fear or greed, which often affect human traders. In addition, algorithmic trading continuously learns and adapts to market dynamics such as volatility, trends, or unforeseen events through interactions with the trading environment. Lastly, this trading method executes trades promptly, capturing opportunities faster than manual traders. In conclusion, RL-based algorithmic trading is far superior in handling complex, fast-paced, and data-intensive market environments.

We leverage both value-based RL and policy-based RL approaches. The first approach is Deep Q-Learning, which integrates Q-Learning with neural networks to handle complex environments. Q-Learning is a value-based RL algorithm that learns the optimal action-selection policy by estimating the Q-value, which is the expected cumulative reward of taking an action  $a$  in a state  $s$  and following the policy thereafter. In complex environments like Ethereum trading, the state space is vast (e.g., historical prices, volumes, indicators). A Deep Neural Network (DNN) is used to approximate the Q-value function, enabling efficient learning and decision-making in high-dimensional spaces. Possible actions for Ethereum trading are Buy, Sell, or Hold. The reward is tied to the Sharpe ratio to maximize risk-adjusted profits ( $R = \text{Average portfolio return} / \text{Portfolio return volatility}$ ).

The second approach is Proximal Policy Optimization (PPO), which is a policy-gradient-based algorithm. The model optimizes a parameterized policy by maximizing a reward function. It introduces constraints to ensure that the updated policy does not deviate too much from the previous one, promoting stability and avoiding large destructive updates. Its clipped objective prevents extreme updates that can lead to instability in trading policies. In addition, the model uses minibatch updates and multiple epochs over collected data, making it efficient for learning.

The third approach is Advantage Actor-Critic (A2C), which combines the strengths of value-based and policy-based methods. It uses a dual network structure to stabilize learning while optimizing decision-making, making it effective for Ethereum trading, where risk-adjusted returns are critical. This approach offers several advantages. First, A2C continuously learns a dynamic trading policy that adapts to changing market conditions. Additionally, its synchronization across workers prevents instability during training, which is crucial for volatile markets.

We assume the same environment design for the PPO and A2C models as in the DQN model. However, some parameters can be added, removed, or modified based on the distinct requirements of each model.

## **4. Dataset Background and Data Preprocessing**

### **4.1. Dataset Source**

The dataset consists of daily Ethereum trading data from 2018 to 2022. The features include:

Date: The trading date.

Open: The opening price of the cryptocurrency.

High: The highest price during the trading day.

Low: The lowest price during the trading day.

Close: The closing price of the cryptocurrency.

Adj Close: Adjusted closing price accounting for dividends and splits (removed during preprocessing).

Volume: The trading volume.

This data provides a foundation for constructing features used in a Deep Q-learning framework.

### **4.2. Derived Features**

To make the data suitable for reinforcement learning and to capture the dynamics of the market, the following features were engineered:

Logarithmic Return: This was included to account for non-stationarity in price movements.

Moving Average (MA 20): A 20-day moving average of the adjusted closing price, used to track medium-term trends.

Volatility: A 20-day rolling standard deviation of the adjusted closing price, representing market uncertainty.

Lagged Features: Price lagged by 1 to 5 days (Lag 1, Lag 2, ..., Lag 5) to provide historical context to the model.

Volatility-MA Interaction (Volatility\_MA20): The product of volatility and the 20-day moving

average, used to capture interaction effects.

Market states were categorized based on the volatility levels:

1. Low Volatility:  $\text{Volatility} < 0.02$
2. Medium Volatility:  $0.02 \leq \text{Volatility} < 0.05$
3. High Volatility:  $\text{Volatility} \geq 0.05$

#### 4.3. Preprocessing Steps

1. Data Cleaning:
  - a. Columns deemed unnecessary (Adj Close) were removed.
  - b. Rows with missing values after feature creation were dropped.
2. Normalization: A sliding window normalization was applied to features (Log Return, MA 20, and Volatility).
  - a. A 50-day window was used to scale features dynamically.
  - b. Values were clipped to the range to handle extreme outliers.
3. Splitting into Training and Testing Sets:
  - a. Training Set (in\_sample): Data from 2018-01-01 to 2020-12-31.
  - b. Testing Set (out\_sample): Data from 2021-01-01 to 2022-12-31.
  - c. Note: Both sets were reset to ensure zero-based indexing.

### 5. Model Comparison

The state space is represented by three features: Log Return, Moving Average (20-day), and Volatility. The action space is what the agent can take one of the following actions:

1. Hold: Maintain the current portfolio.
2. Buy: Use the available balance to purchase cryptocurrency.
3. Sell: Liquidate holdings.

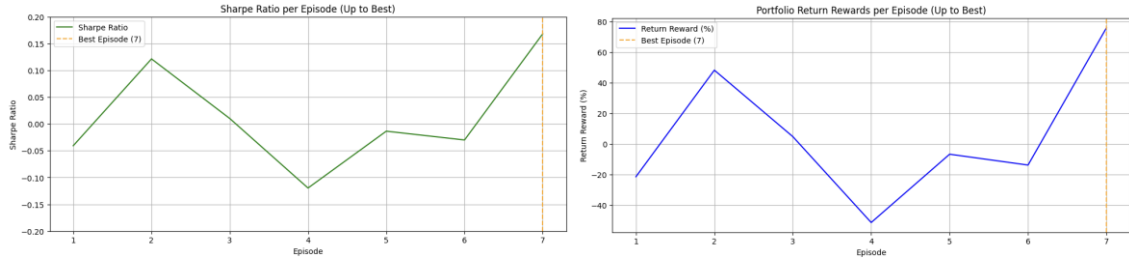
The reward is primarily based on the Sharpe Ratio, encouraging the agent to optimize risk-adjusted returns while maintaining profitability.

The environment tracks the portfolio's total value, balance, holdings, and Sharpe Ratio over time. It terminates when the last trading day is reached or when the maximum number of steps is exceeded.

#### 5.1. Deep Q-Learning Network (DQN)

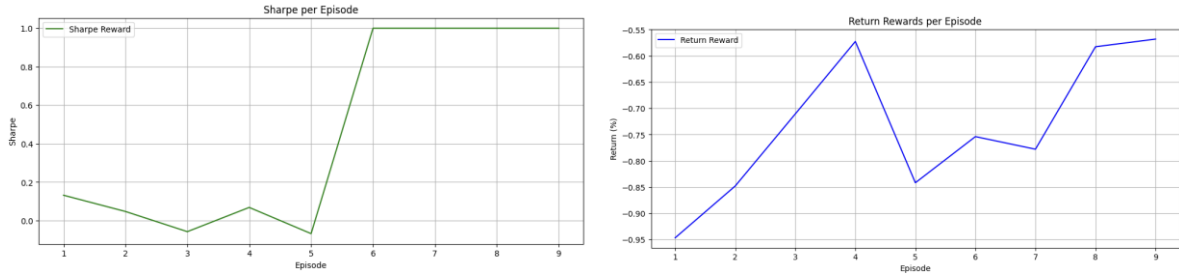
The objective is to train an RL agent to maximize portfolio returns while managing risks using the Sharpe ratio. The agent determines actions (Buy, Hold, Sell) based on market states derived from Ethereum price data. The environment, features, and dataset are shared across experiments.

DQN utilizes a neural network to approximate Q-values for state-action pairs. The agent learns by updating Q-values using the Bellman equation. Key components include three dense layers with ReLU activation followed by a linear output layer, replay buffer to store past experiences for mini-batch training to avoid temporal correlation, and Epsilon-Greedy strategy to balance exploration and exploitation by decaying the exploration rate (epsilon) over episodes. Rewards are focused on a Sharpe ratio for the non-normalized model and are focused on both Sharpe ratio and portfolio returns, normalized for stability for the normalized model.



**[Table: training result in DQN - Non-normalized Model]**

**Testing result in DQN - Non-normalized Model: Sharpe Reward: 0.0555, Portfolio Return Reward: 180.47%**



**[Table: training result in DQN - Normalized Model]**

**Testing result in DQN - Normalized Model: Sharpe Reward: 0.0245, Portfolio Return Reward: 38.78%**

The Non-Normalized Model excels in generating higher portfolio values and stronger Q-values, particularly in high-volatility scenarios where "Buy" and "Sell" actions dominate. However, it tends to exploit high-risk situations, leading to extreme Sharpe ratios across states (e.g., 1.104 for "Buy" in high volatility). Training losses show significant fluctuations, and while Q-values start high, they decline slightly toward the best episode, indicating some instability.

The Normalized Model, on the other hand, emphasizes balanced and consistent decision-making, with smoother training loss and steady Q-values (range ~2-7). While portfolio values are generally lower, the model demonstrates safer, more stable performance, with modest Sharpe ratios (e.g., 0.876 for "Hold" in low volatility) and less aggressive volatility exploitation, reflecting a focus on risk-adjusted returns.

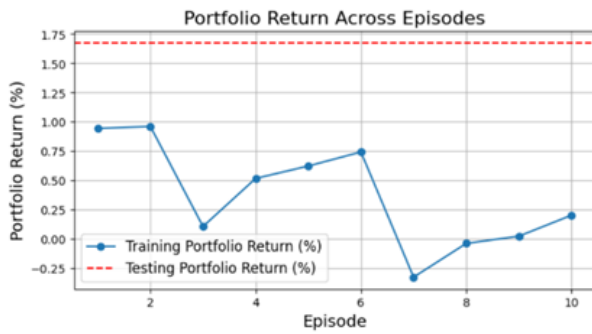
Ultimately, the choice depends on the risk tolerance and objectives of the investor or system designer. For most practical applications, the normalized model is the safer and more robust choice.

## 5.2. Proximal Policy Optimization (PPO)

The objective is to train an RL agent to maximize portfolio returns while managing risks, measured using the Sharpe ratio. The agent learns to take actions (buy, hold, sell) based on market states derived from historical Ethereum price data. The results are evaluated based on portfolio returns (%) and Sharpe ratio. For this approach, we use the same dataset, features and environment as in DQN.

PPO optimizes the policy (probability distribution over actions) by maximizing a clipped surrogate objective, ensuring stability and preventing drastic updates. There are some key components in the approach. First, the policy network outputs a probability distribution over action (Hold, Buy, Sell) It uses neural networks where there are three dense layers with ReLU activation, followed by a softmax output layer. Second, the value network estimates the expected cumulative reward. It is similar to the policy network but has a linear output layer. Third, Generalized Advantage Estimation (GAE) is used

to calculate advantages, balancing bias and variance in reward estimation. Finally, clipping mechanism ensures updates to the policy are not overly large, maintaining stability.



**[Table: training result in PPO]**

**Testing result in PPO: Total portfolio return 1.68%, Sharpe ration 0.0477**

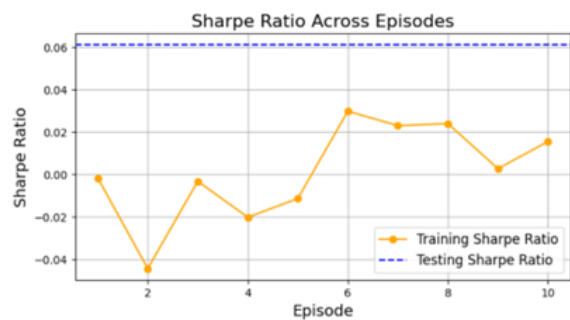
Unfortunately, we noticed that the lack of improvement in training performance and the disappointing testing results. These results can arise due to several reasons. First, the model might be converging to suboptimal actions early because of insufficient exploration. Second, the rewards (portfolio return and Sharpe ratio) are small and noisy, which makes it difficult for the agent to learn effectively. Lastly, Suboptimal hyperparameters might be causing instability or slow convergence.

For future improvement, we can consider some better approaches. First of all, we can increase training episodes and steps to allow for more robust learning. Also, reward scaling and shaping can help amplify meaningful signals. Lastly, using entropy regularization can encourage exploration and avoid premature convergence.

### 5.3. Advantage Actor-Critic (A2C)

The aim of A2C approach is the same with DQN and PPO. It strives to maximize portfolio returns and manage risk effectively, using the Sharpe ratio. The environment, dataset, and features are the same with DQN.

A2C uses the policy network (Actor) and value network (Critic) as PPO does. One differentiation from PPO is the use of advantage calculation. The advantage quantifies how much better or worse a selected action was compared to the expected value. After updating the networks in the training, the process is repeated for multiple episodes to refine the policy and value networks. In the testing process, the policy network is tested on the out-sample dataset.



**[Table: training result in A2C]**

**Testing result: Total portfolio return 1.09%, Sharpe ration 0.0613**

We also got poor performance results as in the PPO. It might be due to the poor hyperparameter tuning. Also, data preprocessing issues could be a reason for the bad performance because insufficient feature engineering or improper normalization of input features could lead to difficulty in learning meaningful patterns.

In order to improve A2C, we can apply hyperparameter optimization in learning rate, discount factor, entropy coefficient, or value loss weight. Grid search or Bayesian optimization could be helpful for systematic hyperparameter tuning. In addition, if we incorporate other technical indicators (e.g., RSI, MACD, Bollinger Bands) and market features (e.g., trade volume), we could get a better result.

## 6. Conclusion, Discussion and Future Work

This study evaluated the performance of various reinforcement learning models, including DQN, PPO, and A2C, in optimizing portfolio returns while managing risks using the Sharpe ratio. Among the approaches, DQN emerged as the most effective, with the Non-Normalized DQN achieving superior portfolio returns (180.47%) at the expense of higher volatility, while the Normalized DQN prioritized stability and risk-adjusted returns (38.78%). In contrast, PPO and A2C struggled with limited exploration and noisy rewards, resulting in suboptimal portfolio performance and Sharpe ratios.

The Non-Normalized DQN's high returns are driven by its aggressive exploitation of high-volatility states, as reflected in extreme Q-values and Sharpe ratios. However, this strategy may expose the portfolio to significant risks in unpredictable market conditions. On the other hand, the Normalized DQN's balanced approach demonstrates the importance of stability and consistent decision-making in financial applications. PPO and A2C failed to achieve satisfactory results, likely due to insufficient reward signal amplification and hyperparameter optimization, highlighting the sensitivity of policy-based models to training configurations and reward structures.

To improve performance, future efforts should focus on enhancing feature engineering by incorporating additional market indicators (e.g., RSI, Bollinger Bands) and reward shaping to amplify meaningful signals for learning. For PPO and A2C, advanced hyperparameter tuning techniques like Bayesian optimization could address instability issues. Additionally, combining elements of DQN with policy-based methods, such as hybrid architectures or ensemble models, may offer a balance between exploitation and exploration. Lastly, extending training time and introducing adversarial testing environments could further improve robustness and generalizability in real-world applications.

## References

Betancourt, Carlos, and Wen-Hui Chen. "Reinforcement Learning with Self-Attention Networks for Cryptocurrency Trading." *Applied Sciences* 11, no. 16 (2021): 7377. <https://www.mdpi.com/2076-3417/11/16/7377>.

Bertillo, Daniele, Carlo Morelli, Giuseppe Sansonetti, and Alessandro Micarelli. "A Comparative Analysis of Reinforcement Learning Approaches to Cryptocurrency Price Prediction." In *Communications in Computer and Information Science, HCI International 2022 – Late Breaking Posters*, 597–604. November 2022. [https://link.springer.com/chapter/10.1007/978-3-031-19682-9\\_75](https://link.springer.com/chapter/10.1007/978-3-031-19682-9_75).

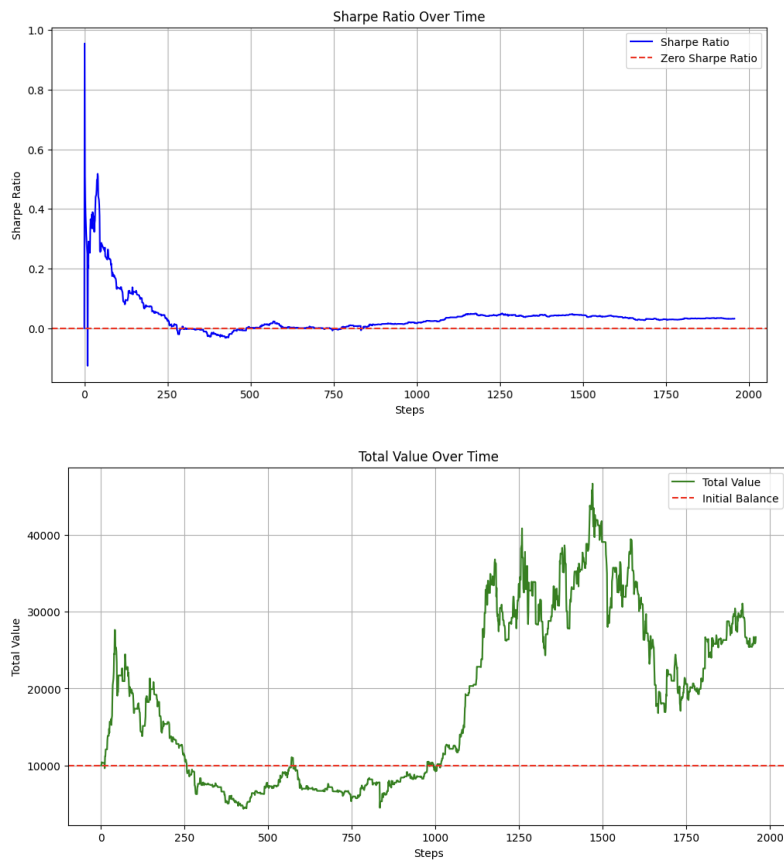
El Akraoui, Bouchra, and Cherki Daoui. "Deep Reinforcement Learning for Bitcoin Trading." *Lecture Notes in Business Information Processing*, May 2022 [https://link.springer.com/chapter/10.1007/978-3-031-06458-6\\_7](https://link.springer.com/chapter/10.1007/978-3-031-06458-6_7).

Kabanda, Gabriel, Colletor Tendeukai Chipfumbu, and Tinashe Chingoriwo. "Utilizing Deep Reinforcement Learning and Q-Learning Algorithms for Improved Ethereum Cybersecurity." *International Journal of Advanced Networking and Applications* 14, no. 6 (2023): 5742–5753.

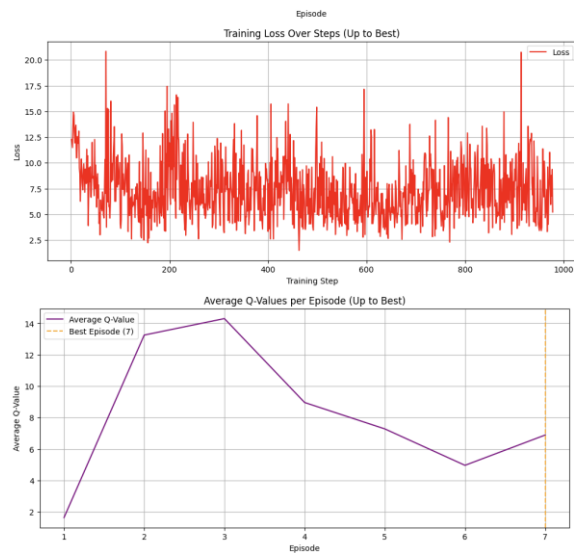
Sane.ai. "Deep Reinforcement Learning for Crypto Trading." *Coinmonks* (blog), Medium, June 2021. <https://medium.com/coinmonks/deep-reinforcement-learning-for-crypto-trading-72c06bb9b04c>.

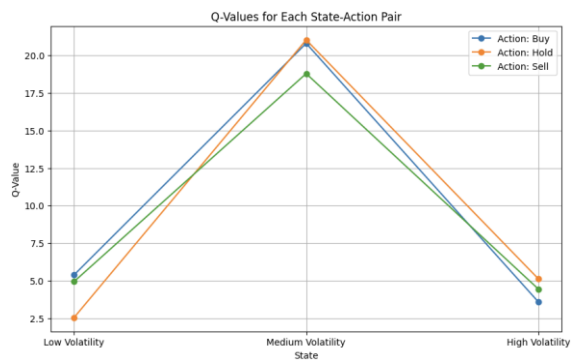
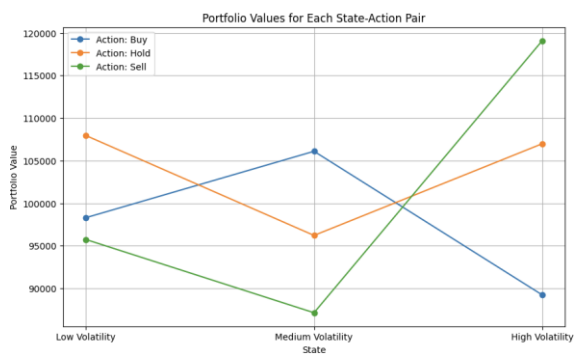
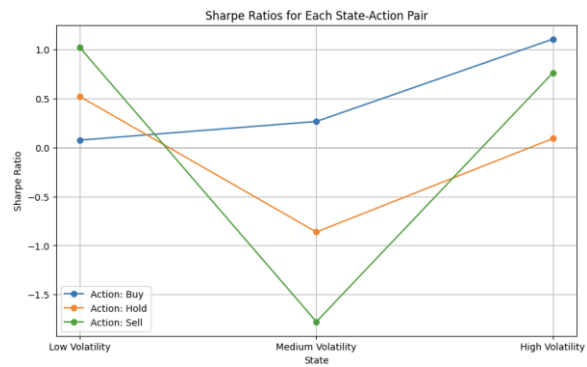
## Appendix

### DQL - Run One Episode



### DQL - Focus only on Sharpe without normalization





## DQL - Focus on Normalized Sharpe and Portfolio Return



