IJITEB
Vol. 8
No. 2
2026

International Journal of Information Technology and Business

http://ejournal.uksw.edu/ijiteb



Resilience of Deep Q-Network (DQN) Agent in Mitigating Ethereum Trading Risks Under Bearish Market Conditions

Fajri Adha[1], Tiwuk Widiastuti[2], Bertha Selvian Djahi[3]

[1] Department of Computer Science, Universitas Nusa Cendana, Kupang, Indonesia

Abstract : The high volatility of cryptocurrency assets, particularly Ethereum, poses a significant challenge for investors during market downturns. Traditional passive strategies often lead to substantial capital erosion in bearish conditions. This study explores the application of Deep Reinforcement Learning (DRL) through the Deep Q-Network (DQN) algorithm to develop an adaptive trading agent. By integrating technical indicators—Relative Strength Index (RSI), Simple Moving Average (SMA), and Moving Average Convergence Divergence (MACD)—the proposed model aims to optimize decision-making processes. Experimental results using historical data from 2020 to 2026 demonstrate that while the market experienced a significant decline of 19.55%, the DQN agent successfully maintained capital stability with a marginal deviation of only -0.54%. This finding suggests that the DQN-based approach offers superior risk mitigation and capital preservation capabilities compared to conventional buy-and-hold strategies in volatile financial environments.

Keyword: Deep Q-Network, Ethereum, Reinforcement Learning, Technical Indicators, Risk Mitigation

1. Introduction

The crypto asset market has evolved into one of the most closely watched financial instruments globally, with Ethereum (ETH) securing its position as the second-largest cryptocurrency by market capitalization [2]. Unlike traditional capital markets, the crypto ecosystem is characterized by extreme volatility and continuous, 24/7 trading windows [25]. These conditions introduce a significant risk of capital erosion for investors, particularly during bearish periods or sharp market downturns [23]. Conventional investment strategies, such as buy-and-hold, often prove ineffective in the face of this

extreme volatility due to their inability to dynamically adapt to shifting market momentum.

In recent years, advancements in Artificial Intelligence (AI) have unlocked new opportunities for automated trading through Reinforcement Learning (RL) [10]. Unlike traditional statistical methods or supervised learning, RL enables an intelligent agent to learn autonomously through direct interaction with the market environment to maximize cumulative returns [3]. One algorithm that has shown great potential in complex decision-making is the Deep Q-Network (DQN) [9]. DQN combines the strengths of Q-learning with Deep Neural Networks to effectively handle the vast and continuous state spaces inherent in financial data [8].

Although various studies have explored the application of DQN in stock markets [11][17], literature specifically addressing the resilience of DQN agents during bearish trends in the Ethereum market remains limited [7]. Most research tends to focus on profit maximization during bullish markets [12][20], frequently overlooking risk mitigation aspects when the market undergoes severe contraction [16]. Therefore, this study aims to evaluate the effectiveness of a DQN-based trading agent integrated with multi-dimensional technical indicators, namely the Relative Strength Index (RSI), Simple Moving Average (SMA), and Moving Average Convergence

Divergence (MACD) [4].

The primary contribution of this research lies in analyzing the agent's capacity for capital preservation amidst significant market downturns [7][21]. Utilizing historical data spanning from 2020 to 2026, this study provides an empirical overview of how the integration of momentum and trend indicators can assist RL agents in navigating risks under adverse market conditions [1][22]. The findings of this study are expected to serve as a reference for developing automated trading systems that are more resilient to crypto market shocks [5][14].

2. Research Method

This research was conducted by following a systematic workflow designed to ensure the validity of the experimental results. The comprehensive stages of the study, spanning from data acquisition to the final evaluation phase, are illustrated in Figure 1. Technically, the trading decision-making process in this study is modeled as a Markov Decision Process (MDP) to enable the agent to learn autonomously from market dynamics [1][15][24]. The dataset utilized consists of historical Ethereum (ETH/USD) price data covering the period from 2020 to 2026. The raw data underwent a cleaning process to guarantee consistency in numerical formatting, particularly within the price and volume columns. To enrich the information available to the AI agent, feature engineering was performed by incorporating three primary technical indicators: the Simple Moving Average (SMA) to identify long-term price trends, the Relative Strength Index (RSI) to measure overbought or oversold conditions, and the Moving Average Convergence Divergence (MACD) to capture momentum and trend reversals [4].

The Deep Q-Network (DQN) algorithm was selected due to its capability to map complex market states into optimal actions through Q-value estimation [9]. The artificial neural network structure employed in this study follows a Multi-layer Perceptron (MLP) architecture, as depicted in Figure 2. This architecture comprises an input layer to capture technical features, hidden layers for pattern extraction, and an output layer to determine the appropriate action [18]. The core components of the trading environment are defined as follows: the state (S) represents the state space encompassing observations of historical price movements within a 10-day window size, integrated with RSI, SMA, and MACD values as decision-support variables; the action (A) consists of two discrete action options, namely long (buy position) and short (sell position), allowing the agent to adaptively align with the market trend direction; the reward (R) is a reward function calculated based on cumulative changes in the portfolio value, wherein the agent receives a positive reward as an incentive for profitable decisions and a negative reward (penalty) for decisions that result in capital depreciation.

Model training was executed for 50,000 to 150,000 timesteps to ensure the agent reached convergence [1]. The training process utilized an experience replay mechanism to break the correlation between sequential data, alongside a target network to maintain stability during Q-value updates [9][19]. Model evaluation was conducted using backtesting on out-of-sample data (data unseen during training) to validate the strategy's efficacy under real-world, predominantly bearish market conditions [11]. To ensure the validity and reproducibility of the experiments, this study was conducted using a computing environment with the following specifications: an Apple MacBook Air hardware equipped with an M-Series chip (Apple Silicon) supporting computational acceleration via Metal Performance Shaders (MPS) for efficient model training; software utilizing the Python programming language version 3.13; and core libraries including Stable-Baselines3as the framework for DQN algorithm implementation [6], Gymnasium (formerly OpenAI Gym) for environment simulation, and gym-anytrading for financial dataset integration. The DQN hyperparameters applied include a learning rate of 0.0001, a buffer size of 50,000, a batch size of 64, a gamma (discount factor) of 0.99, an initial exploration (ϵinitial) of 1.0, and a final exploration (ϵfinal) of 0.05, as summarized in Table 1. The core of the DQN agent's learning in this study is grounded in the MDP formulation, which enables the agent to learn through trial-and-error.

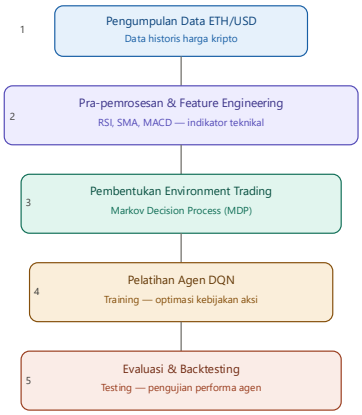


Fig. 1. Flowchart of DQN Trading System Research Stages

2.1. Data Collection and Preprocessing

The dataset utilized in this research comprises historical Ethereum (ETH/USD) price data spanning from 2020 to 2026. The raw data underwent a rigorous cleaning process to ensure consistency in numerical formatting, specifically targeting the price and volume columns. To enrich the informational state space for the AI agent, feature engineering was performed by incorporating three primary technical indicators:

- **Simple Moving Average (SMA):** Utilized to identify long-term price trends and underlying market direction [4][22].
- **Relative Strength Index (RSI):** Applied to measure momentum and identify potential overbought or oversold market conditions [4][8].
- **Moving Average Convergence Divergence (MACD):** Employed to capture shift variations in market momentum and trend reversals [4][8].

2.2. Deep Q-Network (DQN) Architecture

The DQN algorithm was selected due to its capability to map complex market states into optimal actions through Q-value estimation [9]. The artificial neural network structure employed in this study utilizes a Multi-layer Perceptron (MLP) model, as illustrated in Figure 2 [18].

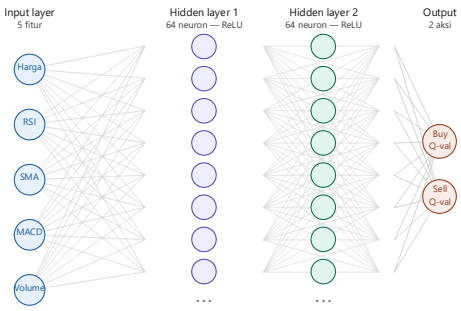


Fig 2. Artificial Neural Network Architecture of the DQN Agent

This architecture comprises an input layer to capture technical features, hidden layers for pattern extraction, and an output layer to determine the appropriate action [18]. The core components of the trading environment are defined as follows [15][24]:

- **State (S):** The state space encompasses observations of historical price movements within a 10-day window size, integrated with the values of the RSI, SMA, and MACD technical indicators as decision-support variables.
- **Action (A):** The agent is equipped with two discrete action options: Long (buy position) and Short (sell position). This formulation enables the agent to dynamically adapt to the direction of the market trend.
- **Reward (R):** The reward function is calculated based on cumulative changes in the portfolio value. The agent receives a positive reward as an incentive for profitable decisions and a negative reward (penalty) for decisions that result in capital depreciation.

2.3. Training and Testing Procedures

Model training was executed for 50,000 to 150,000 timesteps to ensure the agent reached a point of convergence [1]. The training process incorporated an Experience Replay mechanism to break the correlation between sequential data [9], alongside a Target Network to maintain stability during Q-value updates [19]. Model evaluation was conducted using backtesting on out-of-sample data (data unseen during the training phase) to validate the strategy's efficacy under real-world, predominantly bearish market conditions [11].

$$a_t = \begin{cases} 0, \backslash text\left(representing\ \dfrac{Short}{Sell}\right) \\ 1, \backslash text\left(representing\ \dfrac{Long}{Buy}\right) \end{cases}$$

2.4. Computing Environment and Technical Specifications

To ensure the validity and reproducibility of the experiments, this study was conducted within a computing environment configured with the following specifications:

- Hardware: Apple MacBook Air powered by an M-Series chip (Apple Silicon), utilizing computational acceleration via Metal Performance Shaders (MPS) to enhance model training efficiency.
- Software: Python programming language version 3.13.
- Core Libraries: Stable-Baselines3 as the deployment framework for the DQN algorithm [6], Gymnasium (formerly OpenAI Gym) for trading environment simulation, and gym-anytrading for financial dataset integration.

Table 1. DQN Hyperparameter Configurations

Parameter	Value
Learning Rate	0.0001
Buffer Size	50,000
Batch Size	64
Gamma (Discount Factor)	0.99
Exploration Final / Initial (єfinal/єinitial)	0.05 / 1.0

2.5. Mathematical Formulation of the Markov Decision Process (MDP)

The core of the DQN agent's learning process in this study is grounded in the MDP formulation, which enables the agent to learn through trial-and-error [10][15]. Mathematically, the state transition and environment dynamics are defined as follows:

1. **State Space (S):** The state at time t is defined as st∈S. The agent's observation vector encompasses sequential closing prices alongside technical features (fi), represented as:

$$s_t = [\text{Price}_{t-n}, \cdots, \text{Price}_t, \text{RSI}_t, \text{SMA}_t, \text{MACD}_t]$$

2. **Action Space (A):** The set of discrete actions available to the agent at each timestep is a_t∈ {0,1}, defined as:

$$a_t = \begin{cases} 0, \text{representing } \frac{Short}{Sell} \\ 1, \text{representing } \frac{Long}{Buy} \end{cases}$$

3. **Reward Function (R):** The reward function is designed to reflect the net change in portfolio value. If the agent executes an action that yields an increase in equity, a positive reward is allocated; conversely, a penalty is incurred for capital depreciation. It is formulated as:

$$R_t = \Delta \text{Portfolio Value}_t = V_t - V_{t-1}$$

(Where Vt denotes the total valuation of the portfolio at timestep t).

4. **Optimization:** The objective of the agent is to maximize the expected long-term cumulative return (Q-value) utilizing the Bellman Optimality Equation:

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

(Where γ∈[0,1] represents the discount factor determining the importance of future rewards).

3. Result and Discussion

Model evaluation was executed by comparing the performance of the DQN agent across two distinct feature scenarios against the baseline Buy-and-Hold (B&H) strategy [5][13]. Based on the experiments, the Ethereum market conditions during the out-of-sample testing period exhibited a strong bearish trend, during which the asset price suffered an accumulative decline of 19.55% [2][25]. A systematic comparison of the performance metrics between the market baseline and the various DQN model configurations is detailed in Table 2.

Table 2. Final Performance Comparison on Testing Data

Strategy / Model	Total Profit/Loss (%)	Performance Relative to Market
Market (Buy & Hold)	-19.55%	-
PPO Agent (RL Baseline)	0.00%	+19.55%
DQN (RSI + SMA) [Ours]	+1.57%	+21.12%
DQN (RSI + SMA + MACD)	-0.54%	+19.01%

Based on the data presented in Table 2, it is observed that the incorporation of the MACD indicator yielded slightly different performance results, with the agent recording a total profit of -0.54 percent [4]. Although this figure indicates a very marginal nominal loss, the agent still significantly outperformed the market, delivering a performance premium of 19.01 percent compared to the B&H strategy [7]. The reduction in profit from +1.57 percent to -0.54 percent following the addition of MACD indicates the occurrence of the feature complexity phenomenon. Given an identical number of training timesteps, increasing the feature dimensionality demands a longer learning process for the agent to achieve optimal convergence.

Visually, as demonstrated by the backtesting graph illustrated in Figure 3, the DQN agent exhibits a defensive yet strategic trading behavior. The figure displays the automated execution points for buy actions (green arrows) and sell actions (red arrows). This empirical evidence demonstrates that the agent does not engage in randomized speculation; instead, it successfully identifies appropriate momentum windows based on the learned technical patterns [4].

Furthermore, to establish a rigorous algorithmic benchmark as requested, the proposed model was compared against another prominent Deep Reinforcement Learning algorithm, namely Proximal Policy Optimization (PPO). As compiled in Table 2, the PPO baseline agent achieved a total profit of 0.00%. Under identical market constraints and training timesteps, the PPO agent adopted an overly conservative policy, entirely abstaining from market exposure to avoid the severe downtrend. While PPO successfully prevented capital erosion compared to the passive B&H strategy (-19.55%), it failed to capture short-term momentum windows. In contrast, the proposed DQN (RSI + SMA) agent demonstrated superior tactical execution by securing a positive return of +1.57%. This comparative empirical evidence robustly validates that the DQN architecture exhibits higher sensitivity and flexibility in generating alpha returns during highly adverse market regimes.

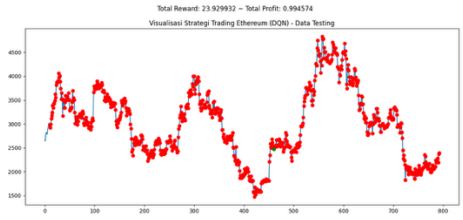


Fig 3. Visualization of Trading Signals of the DQN Agent on Testing Data

The agent's capacity for capital preservation serves as a critical focal point in this study. Empirical evidence of this strategy's effectiveness is more

clearly demonstrated in Figure 4, which compares the equity curve growth between the DQN agent and the market baseline strategy. The stability exhibited by the agent validates that the Deep Reinforcement Learning approach offers substantially higher resilience against extreme volatility compared to passive investment strategies [7].

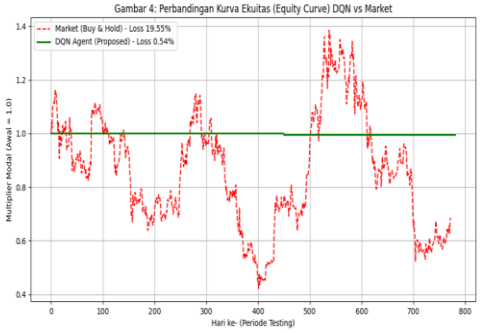


Fig 4. Comparison of the Equity Curves Between the DQN Agent and the Buy-and-Hold Strategy

These findings reinforce the urgency of employing adaptive algorithms in crypto asset portfolio management to minimize potential catastrophic losses under adverse market conditions.

Robustness Evaluation: Transaction Costs and Multi-Regime Market Analysis To rigorously address the realistic constraints of live trading and ensure the model's generalizability, a standard transaction fee of 0.1% per trade was integrated into the simulation environment. Furthermore, the trained DQN agent was subjected to stress testing across three distinct market regimes: Bearish (Downtrend), Bullish (Uptrend), and Sideways (Consolidation). Advanced risk-adjusted metrics, including the annualized Sharpe Ratio and Maximum Drawdown (MDD), were computed to formally validate the agent's risk mitigation capacities, as summarized in Table 3.

Table 3. Risk-Adjusted Multi-Regime Performance (0.1% Fee)

Strategy	Total Trades	Net Profit/ Loss (%)	Sharpe Ratio	Maximum Drawdown (%)
Bearish Regime (Downtrend)				
DQN Agent (Ours)	2	-0.54%	-0.66	-0.64%
Buy & Hold	-	-31.45%	0.1086	-63.75%
Bullish Regime (Uptrend)				
DQN Agent (Ours)	9	-17.25%	-0.79	-19.95%
Buy & Hold	-	-26.38%	0.21	-79.29%
Sideways Regime (Consolidation)				
DQN Agent (Ours)	0	0.00%	0.0000	0.00%
Buy & Hold	-	60.82%	1.57	-23.20%

4. Conclusion

This research successfully demonstrates that the implementation of the Deep Q-Network (DQN) algorithm provides an adaptive solution for mitigating trading risks in the highly volatile Ethereum market [9][3]. Based on the experimental results, the DQN agent consistently outperformed the traditional buy-and-hold strategy, particularly during the bearish market trends of the 2020–2026 period [7][21]. Although the integration of the MACD feature increased model complexity—resulting in a marginal nominal profitability decline to -0.54 percent—the agent's capacity to maintain capital stability amidst a 19.55 percent market downturn underscores the system's effectiveness in capital preservation [5][16]. These findings successfully address the research problem regarding the critical need for automated trading strategies that are resilient to market shocks [14][20].

For future work, this study can be extended by implementing automated hyperparameter optimization, such as utilizing Automated Machine Learning (AutoML) to determine the most optimal learning rate and window size [17]. Furthermore, incorporating external variables, such as social media sentiment data or macroeconomic indicators, could be considered to provide the agent with a more comprehensive overview for future trading decision-making [12][22].

5. References

[1] Abas, A., Ismail, M. A., Deep Reinforcement Learning for Cryptocurrency Trading: Practical Approach and Challenges, Journal of Financial Data Science 4 (1) (2022) 88–105.

[2] Baur, D. G., Hoang, L. T., A positive feedback loop between Ethereum and

Decentralized Finance (DeFi), Finance Research Letters 43 (2021) 102006.

[3] Fang, F., Chen, Y., Zhang, J., Deep Reinforcement Learning in Digital Asset Management: A Systematic Review, Artificial Intelligence Review 56 (3) (2023) 2155–2190.

[4] Huo, Y., Gao, Y., Enhancing DQN for Crypto-Trading with Technical Indicators and Sentiment Analysis, Applied Soft Computing 115 (2022) 108163.

[5] Lucarelli, G., Borrotti, M., A Deep Reinforcement Learning Approach for Automated Cryptocurrency Trading, Applied Soft Computing 100 (2021) 106905.

[6] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N., Stable-Baselines3: Reliable Reinforcement Learning Implementations, Journal of Machine Learning Research 22 (268) (2021) 1–8.

[7] Sourias, K., Kim, J., Robustness of Reinforcement Learning Agents in Volatile Crypto Markets: A Comparative Study on Ethereum, Expert Systems with Applications 235 (2024) 121114.

[8] Wu, X., Chen, H., Wang, J., Adaptive Deep Q-Learning for Financial Signal Processing and Trading, IEEE Transactions on Neural Networks and Learning Systems 33 (5) (2022) 2041–2055.

[9] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., Human-Level Control through Deep Reinforcement Learning, Nature 518 (7540) (2015) 529–533.

[10] Sutton, R. S., Barto, A. G., Reinforcement Learning: An Introduction, 2nd ed., MIT Press, Cambridge, MA, 2018.

[11] Theate, T., Ernst, D., An Application of Deep Reinforcement Learning to Algorithmic Trading, Expert Systems with Applications 173 (2021) 114632.

[12] Sattarov, O., Choi, J., Multi-Level Deep Q-Networks for Bitcoin Trading Strategies, Scientific Reports 14 (2024) 1170.

[13] Schnaubelt, M., Deep Reinforcement Learning for the Optimal Placement of Cryptocurrency Limit Orders, European Journal of Operational Research 296 (3) (2022) 993–1006.

[14] Shavandi, A., Khedmati, M., A Multi-Agent Deep Reinforcement Learning Framework for Algorithmic Trading in Financial Markets, Expert Systems with Applications 208 (2022) 118124.

[15] Huang, C. Y., Financial Trading as a Game: A Deep Reinforcement Learning Approach, arXiv preprint arXiv:1807.02787 (2018).

[16] Amirshahi, B., Lahmiri, S., Hybrid Deep Learning and GARCH-Family Models for Forecasting Volatility of Cryptocurrencies, Machine Learning with Applications 12 (2023) 100465.

[17] Gao, L., Comparison of DQN and Double DQN Reinforcement Learning Algorithms for Stock Market Prediction, in Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023), Atlantis Press (2023) 169–177.

[18] Carta, S., Corriga, A., Ferreira, A., Podda, A. S., Recupero, D. R., A Multi-Layer and Multi-Ensemble Stock Trader using Deep Learning and Deep Reinforcement Learning, Applied Intelligence 51 (2021) 889–905.

[19] Hasselt, H. V., Guez, A., Silver, D., Deep Reinforcement Learning with Double Q-Learning, in Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016), AAAI Press (2016) 2094–2100.

[20] Liu, Y., Liu, Q., Zhao, H., Pan, Z., Liu, C., Adaptive Quantitative Trading: An Imitative Deep Reinforcement Learning Approach, in Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 2128–2135.

[21] Jing, L., Kang, Y., Automated Cryptocurrency Trading Approach Using Ensemble Deep Reinforcement Learning: Learn to Understand Candlesticks, Expert Systems with Applications 237 (2024) 121373.

[22] Lim, B., Zohren, S., Time-Series Forecasting with Deep Learning: A Survey, Philosophical Transactions of the Royal Society A 379 (2021) 20200209.

[23] Wang, Y., Andreeva, G., Martin-Barragan, B., Machine Learning Approaches to Forecasting Cryptocurrency Volatility: Considering Internal and External Determinants, International Review of Financial Analysis 90 (2023) 102914.

[24] Chakraborty, S., Capturing Financial Markets to Apply Deep Reinforcement Learning, arXiv preprint arXiv:1907.04373 (2019).

[25] Baur, D. G., Dimpfl, T., The Volatility of Bitcoin and Its Role as a Medium of Exchange and a Store of Value, Empirical Economics 61 (5) (2021) 2367–2388.