Journal of Advanced College of Engineering and Management, Vol. 10, 2025, Advanced College of Engineering and Management

STOCK TRADING INDICATOR USING REINFORCEMENT LEARNING

Ashmita Tiwari [1], Anand Kumar Sah [2], Gaurav Pandeya [3], and Laxmi Prasad Bhatt [1]

[1,2] Pulchowk Engineering Campus, Kathmandu, Nepal
[3] School of Mathematical Sciences, Kritipur, Kathmandu, Nepal
Email: 079mscsk002.ashmita@pcampus.edu.np, anand.sah@pcampus.edu.np, gaurav.795552@sms.tu.edu.np, lpbhatta0828@gmail.com

ABSTRACT

Traditional trading strategies often struggle to consistently achieve profitable results in highly volatile and complex financial markets. This research explores the potential of Reinforcement Learning (RL) to develop a trading indicator capable of learning and adapting to market changes. By leveraging RL, the trading indicator aims to improve decision-making and enhance trading performance through continuous learning from experiences.

The outcomes of this research include a flexible and adaptive RL-based trading indicator, trading strategies that generate higher returns with better risk management, and a thorough comparison with traditional methods. This study demonstrates how advanced machine learning can be effectively applied to enhance trading in financial markets. In contrast to traditional technical analysis, which focuses on statistical trends from trading activity, RL offers an experience-driven approach that can adapt to evolving market conditions. The RL-based trading indicator utilizes Q-learning, a model-free reinforcement learning algorithm, to learn optimal action-selection policies. Through iterative updates of Q-values, the agent can derive the optimal policy by selecting the action with the highest Q-value in any given state.

The research evaluates the performance of the RL-based trading indicator against traditional indicators like MACD and RSI across multiple stocks. The results consistently demonstrate the superior performance of the RL-based indicator in terms of profitability and adaptability. This highlights the potential of RL as a promising tool for enhancing financial trading strategies.

Keywords: Reinforcement Learning, Stock Trading, Financial Markets, Moving Average Convergence Divergence, Relative Strength Index, Risk Management.

1. Introduction

Financial markets are highly volatile, influenced by factors like economic indicators and investor sentiment. Traditional trading strategies, relying on historical data, often struggle to adapt to sudden market shifts, leading to suboptimal decisions in dynamic conditions.

1. Traditional Stock Trading Analysis

Traditional stock analysis involves fundamental analysis, which evaluates a company's financial health, and technical analysis, which focuses on price trends and trading activity. Fundamental analysis looks at financial statements, ratios, and economic indicators, while technical analysis uses tools like moving averages, RSI, Bollinger Bands, and MACD to predict price movements. Both approaches have limitations, including reliance on historical data and fixed rules, making them less adaptable to rapidly changing market conditions [1].

2. Modern stock Trading Analysis

In recent years, machine learning (ML) and reinforcement learning (RL) have gained attention for enhancing stock analysis and prediction in financial markets. Machine learning enables systems to learn from data for tasks like risk assessment and trading strategy development, while RL focuses on an agent learning optimal strategies by interacting with an environment to maximize cumulative rewards. Q-learning, a model-free RL algorithm, updates Q-values based on the Bellman equation [2]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma a' \max Q(s',a') - Q(s,a))$$

where s is the current state, aa is the action, r is the reward, s' is the next state, $\alpha\alpha$ is the learning rate, and γ is the discount factor. Over time, this update allows the agent to derive an optimal policy by choosing actions with the highest Q-values.

Deep Q-learning extends Q-learning by using deep neural networks to approximate the Q-value function, making it suitable for high-dimensional state spaces. The loss function used in Deep Q-learning is [3]:

$$L(\theta) = E[(r + \gamma a' \max Q(s',a';\theta^-) - Q(s,a;\theta))2]$$

where θ are the network parameters, and θ^- are the target network parameters. Experience replay and target networks stabilize training, making Deep Q-learning effective for complex environments like video games and robotics. Combined, these techniques provide a robust framework for developing optimal trading strategies in dynamic financial markets.

2.1. Limitations of Traditional methods

Traditional trading methods have several limitations:

1. **Lack of Adaptability**: They rely on predefined rules and historical data, making them slow to adjust to new market conditions, such as sudden economic or political events, leading to ineffective decisions and potential losses.

2. **Rigidity**: Fixed parameters and thresholds (e.g., specific RSI levels) are used in trading strategies, which may not suit all market conditions. This inflexibility results in false signals and suboptimal trading decisions.

3. **Limited Use of Data**: Traditional methods typically use only a small subset of data, such as price movements, while ignoring other important factors like trading volume and economic indicators, leading to incomplete analysis and missed opportunities.

1.2. Challenges in Developing Adaptive Trading Strategies:

- **Market Complexity**: Financial markets are influenced by many factors, requiring sophisticated algorithms to process and learn from large data volumes.

- **Risk Management**: Strategies must balance profitability and risk, adjusting dynamically, which traditional methods often fail to do.

- **Real-time Adaptability**: Effective strategies need to adapt to real-time changes, something static traditional methods lack.

- **Computational Resources**: Developing machine learning-based strategies requires significant computational power, making it time-consuming and resource-intensive.

Reinforcement Learning (RL) offers a dynamic, adaptable approach to trading by learning from market interactions and improving over time, addressing these challenges.

1.3. Research Questions

After the thorough review of the existing major problems, the main questions the project aims to answer are as follows:

- How effective are reinforcement learning algorithms, such as Q-learning in developing profitable stock trading strategies compared to traditional trading indicators?

- Which features significantly impact the performance of RL-based trading strategies?

- How do RL-based trading indicators compare with other compared to indicators from traditional techniques in terms of profitability, risk management, and computational efficiency?

1.4. Objectives

The primary objectives of this project are as follows:

- To Use Reinforcement Learning to Develop Stock Trading Indicators.

- To Evaluate Performance of RL based algorithm for maximum profit reward.

- To Compare Performance of RL algorithm against traditional method.

- To Provide easy and intuitive indicator responses to users.

1.5. Significance of The Study

- **For Individual Traders**: Provides better decision-making tools, improving profitability and risk management.

- **For Financial Institutions**: Enhances trading operations, reducing risks, and increasing returns.

- **For Algorithmic Trading Firms**: Helps stay competitive by refining algorithms based on market dynamics.

This study contributes to financial machine learning by showcasing RL's advantages over traditional methods, providing insights into its effectiveness in real-world trading.

2. Related Work

2.1. Classical Approaches

"Does the Bond Market Predict Bankruptcy Settlements?" (Eberhart & Sweeney, 1992):
This study examines whether the bond market can predict bankruptcy settlement outcomes, particularly regarding violations of the absolute priority rule (APR), which favors senior creditors. Findings suggest that the bond market anticipates APR violations during bankruptcy filings, reflecting informational efficiency as bond prices align with expected outcomes [4].

Robert F. Engle's "Autoregressive Conditional Heteroscedasticity (ARCH)":
Engle introduces the ARCH model to account for varying conditional variances in econometric analyses, addressing limitations of constant variance assumptions. Applied to UK inflation data, particularly during the 1970s, the model captures significant ARCH effects. Engle demonstrates that maximum likelihood estimators are more efficient than ordinary least squares (OLS), enhancing forecast precision [5].

2.2. Early Machine Learning Approaches

"Financial Time Series Forecasting Using Support Vector Machines" (Kim, 2003):
This study evaluates the effectiveness of Support Vector Machines (SVMs) in predicting financial time series data, such as stock prices, compared to back-propagation neural networks. Using metrics like mean squared error (MSE) and mean absolute percentage error (MAPE), SVMs are shown to outperform neural networks in accuracy and generalization. The superior performance is attributed to SVMs' ability to manage non-linear relationships and avoid overfitting, making them a robust tool for financial forecasting [6].

"Stock Market Prediction System with Modular Neural Networks" (Kimoto et al., 1990):
This research investigates modular neural networks for predicting stock market movements, focusing on the Tokyo Stock Exchange Prices Indexes (TOPIX). By breaking prediction tasks into smaller sub-tasks, the modular approach improves learning efficiency and accuracy. The system demonstrates high prediction accuracy and profitability in simulations, emphasizing its effectiveness in handling the complexities of financial data [7].

2.3. Reinforcement Learning and Its Approaches

Watkins' "Learning from Delayed Rewards": Introduced Q-learning, demonstrating its robustness in learning optimal policies in environments with delayed rewards, highlighting its foundational role in reinforcement learning [8].

"A Deep Reinforcement Learning Framework for Financial Portfolio Management" (Jiang et al., 2017): Proposed a DRL framework for portfolio management, achieving superior risk-adjusted returns and adaptability, though requiring extensive training data and computational resources [9].
"Model-based Deep Reinforcement Learning for Financial Portfolio Optimization" (Yu et al.): Developed a DRL framework integrating predictive market models to improve returns and risk management in dynamic markets [10].

"Multi-Feature Supervised Reinforcement Learning for Stock Trading" (Fu et al.): Combined LSTM and DDPG for faster convergence and better market trend analysis, outperforming traditional methods in profitability and stability [11].

"Deep Reinforcement Learning for Trading Automation" (TD3-based model): Created an autonomous stock trading system incorporating market and sentiment data, achieving high profitability and risk-reward balance with a Sharpe ratio of 2.68 [12].

3. Material and Methods

3.1. Project Design

The methodology for developing the stock trading indicator involves several key steps:

- **Data Collection:** Historical stock price data and relevant market indicators from reliable financial data sources, such as NEPSE stock data will be gathered. The data will cover daily stock information about open, close, high, low, etc. To ensure the RL model can learn from various of these parameters.

- **Data Pre-processing:** Data pre-processing will involve cleaning and preparing the data to make it suitable for training the RL models. This step includes normalizing the data to ensure consistency, handling missing values, and performing feature engineering to extract relevant features that will enhance the learning process of the RL algorithms.

- **Model Selection:** Evaluation and selection of appropriate reinforcement learning algorithms based on their suitability for the trading environment. The algorithm will be Q-learning and it will be assessed for its ability to handle the complexities of stock trading, such as continuous action spaces and the need for real-time decision-making.

- **Training the Model:** The selected RL models will be trained using historical data. This involves setting up a simulated trading environment where the RL agent can interact with the market data, make trading decisions, and learn from the outcomes. The training process will focus on allowing the RL agent to discover optimal trading strategies through continuous iteration and feedback.

- **Validation and Testing:** Once trained, the RL models will be evaluated using historical data previously unseen by RL model to test their performance. This step will involve back testing the RL-based trading strategies and comparing the indicators obtained as results through the RL algorithm against traditional trading indicators obtained from methods such as moving average

convergence/divergence (MACD) and the Relative Strength Index (RSI). The evaluation will focus on key performance metrics such as profitability, risk-adjusted returns, and drawdown.

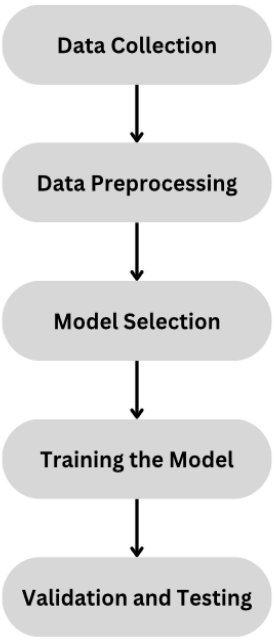

Fig. 1. Proposed Project Flow Chart

3.2. Reinforcement Learning Stock Trading Process

1. Algorithm and Flowchart

Algorithm Steps for Stock Trading with Q-Learning

1) Data Preparation:

(a) Load stock price data from CSV file.

(b) Split the data into training and testing sets based on a specified timestamp.

2) Environment Setup:

(a) Initialize the StockTradingEnv with the training prices and testing prices.

(b) Define the action space (buy or hold or sell) **and observation space.**

3) Q-LearningAgent Setup:

(a) Initialize the Q-LearningAgent with the number of states, actions, learning

rate, discount factor, and exploration rate.

4) Training Phase:

(a) For each episode (n times):

i. Reset the environment to get the initial state.

ii. Loop through the steps until the episode is done:

A Choose an action based on the current state using the Q-learning policy.

B Perform the action in the environment to get the next state and reward.

C Update the Q-table based on the reward and next state.

D Set the current state to the next state.

5) Testing Phase:

(a) Reset the environment to get the initial state.

(b) Loop through the steps until the episode is done:

- Choose an action based on the current state using the learned policy.
- Perform the action in the environment to get the next state and reward.
- Accumulate the total reward.
- Print the action taken, date, price, and capital at each step.

6) Render Results:

(a) Plot the historical price trends with the actions taken (buy/hold or sell)

during the testing phase while also comparing with RSI and MACD indicators.

(b) Print the total reward during the testing phase.

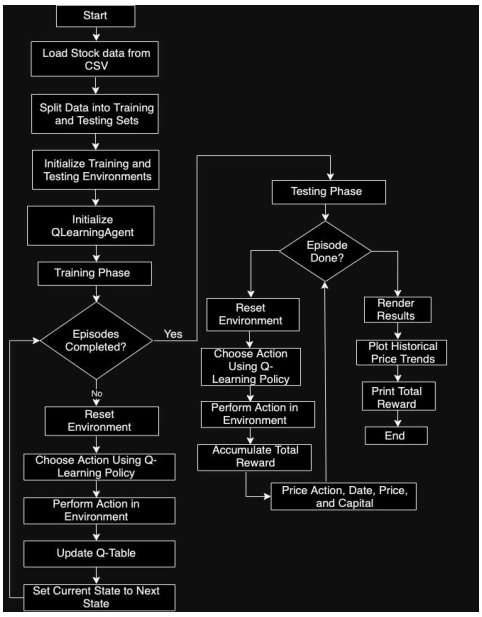

Fig. 2. Reinforcement Learning Stock Trading flow chart

3.3. Experimental Setup

This chapter details the implementation of a stock trading simulation using a Q-Learning agent within a custom OpenAI Gym environment. The following sections cover the design and development of the environment and the Q-Learning agent, and the processes for training and testing the model.

1. Data Collection

The NEPSE stock data is gathered using the nepsealpha.com Open-source API, providing a comprehensive daily overview of the stock market. This dataset includes essential features such as:

- Open: The opening price of the stock for the day.
- Close: The closing price of the stock at the end of the trading day.
- High: The highest price the stock reached during the trading day.
- Low: The lowest price the stock reached during the trading day.
- Volume: The total number of shares traded during the day.

This detailed dataset is crucial for analyzing market trends, making informed trading decisions, and conducting thorough financial research.

2. Environment Design

- **Initialization of environment:** The **StockTradingEnv** environment simulates real-world stock trading scenarios. It is initialized with historical stock prices and an initial capital amount, allowing the agent to trade over a specific period. Key elements include:

- **Prices**: Historical stock prices for making trading decisions.
- **Initial Capital**: The agent's starting amount of money.
- **State Space**: Defined by stock prices for each day in the trading period.
- **Action Space**: Includes actions like buying, selling, or holding a stock.

The environment is compatible with the OpenAI Gym framework, facilitating standard interaction in reinforcement learning.

- **Reset and Step Function:** The **reset function** reinitializes the environment at the start of each episode, resetting the agent's capital and setting the trading day to the first day of the dataset for consistency.

  The **step function** manages the agent's actions (buy, sell, hold) and updates the environment:

  - **Action Processing**: Adjusts the agent's capital based on its action (buying, selling, or holding a stock).
  - **State Transition**: Moves to the next trading day.
  - **Reward Calculation**: Determines the reward based on changes in capital.
  - **Episode Termination**: Checks if the trading period has ended.

  This feedback helps the agent improve its learning process.

- The **render function** visualizes the agent's trading strategy by plotting historical stock prices and marking the agent's actions (buy, sell, hold) on specific days. This helps analyze the agent's performance and decision-making process, using **matplotlib** for clear and informative visualizations.

3. Q-Learning Agent

- **Initialization**: The agent is initialized with parameters like the number of states, actions, learning rate, discount factor, and exploration rate.
- **Action Selection**: The epsilon-greedy policy balances exploration and exploitation by selecting random actions (exploration) or the highest Q-value actions (exploitation).
- **Q-Table Update mechanism**: The **action selection** is based on an epsilon-greedy policy, where:
  - With probability ϵ, the agent randomly explores actions to discover new strategies.
  - Otherwise, it **exploits** the action with the highest Q-value to maximize rewards.

  The **Q-table update** follows the Bellman equation, involving:

- **State and Action Clipping**: Ensuring valid indices.
- **Old Q-Value Retrieval**: Fetching the current Q-value for a state-action pair.
- **Next State Processing**: Finding the maximum Q-value for the next state.
- **New Q-Value Calculation**: Using the reward and maximum next Q-value to compute the new Q-value.
- **Q-Table Update**: Updating the Q-table with the new value.

This process helps the agent improve its trading strategy over time.

4. **Training and Testing Process**

- **Training Procedure**: The **training process** involves multiple episodes where the agent interacts with the environment, learns from experiences, and updates its Q-table. Key steps include:
  - **Episode Initialization**: Resetting the environment with initial state and capital.
  - **Action Selection**: The agent chooses actions based on its epsilon-greedy policy.
  - **Environment Interaction**: The selected action is executed, transitioning to a new state with a reward.
  - **Q-Table Update**: The agent updates its Q-table based on the observed reward and next state.
  - **Episode Termination**: The episode ends when the trading period concludes, and the process repeats for the next episode.
- **Testing Procedure**: In the **testing phase**, the trained agent is evaluated on a separate dataset to assess its ability to generalize to new market conditions. Key steps include:
  - **Environment Reset**: Ensuring consistency by resetting the testing environment.
  - **Action Selection**: The agent selects actions based on the learned Q-table without exploration.
  - **Environment Interaction**: The agent executes actions, and the environment transitions, providing rewards.
  - **Performance Measurement**: The total accumulated reward during the testing period is recorded to evaluate the agent's performance.

4. Results and Discussion

The reinforcement learning (RL) model's performance was tested on a separate dataset to evaluate its ability to make profitable trading decisions based on its training experience. Over a 63-day period, the historical stock price trends were analyzed alongside the agent's actions, which included buying (green triangles), selling (red inverted triangles), and holding (blue circles). These actions demonstrated the agent's capacity to respond dynamically to market conditions, aligning its decisions with price movements to optimize trading outcomes.

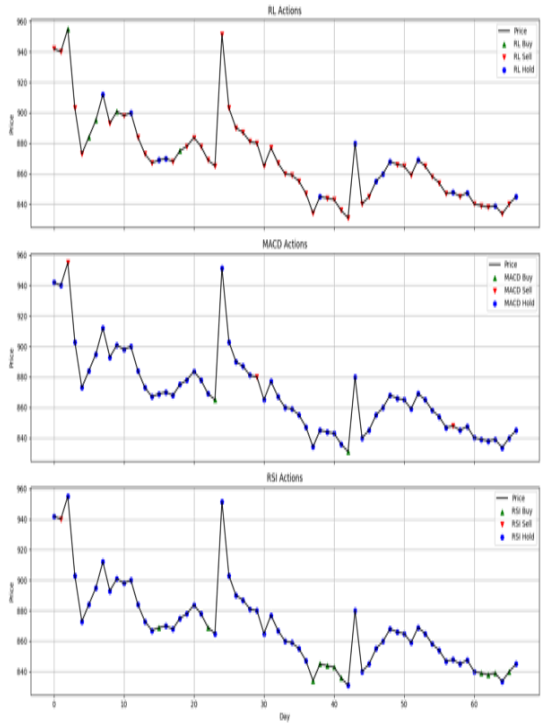

Fig. 3. Price Trends of NTC Stock with Actions Taken by the RL Agent VS Other Indicators.

The graph compares the trading actions of Reinforcement Learning (RL), MACD, and RSI strategies on the NTC stock over 63 days. RL takes more frequent buy/sell actions, adjusting dynamically to price movements, especially during volatile periods, showing a proactive approach to maximizing rewards. In contrast, both MACD and RSI exhibit fewer actions, with MACD focusing on key price peaks and troughs, and RSI mostly issuing buy signals during oversold conditions. Overall, RL demonstrates more flexibility and responsiveness, while MACD and RSI adopt a more conservative Strategy.

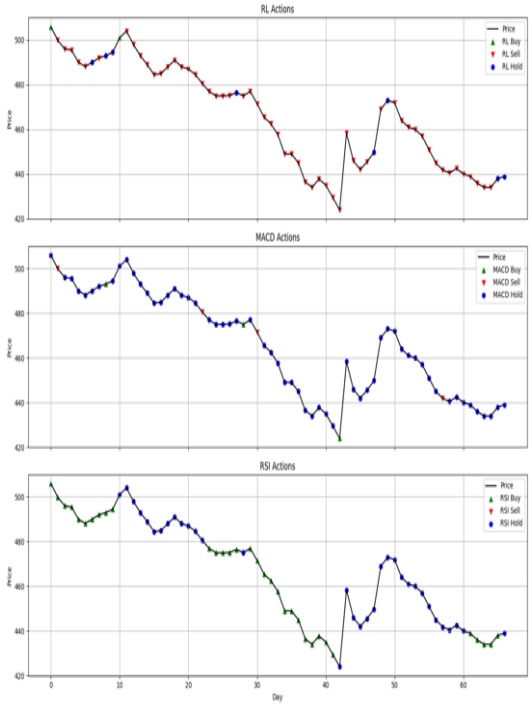

Fig. 5. Price Trends of NABIL Stock with Actions Taken by the RL Agent VS Other Indicators.

The graph presents a comparison of the trading actions taken by RL, MACD, and RSI strategies on the NABIL stock over 63 days. The RL agent exhibits more frequent buy/sell actions, closely following the price decline, which indicates a more active and responsive trading strategy. The MACD strategy shows fewer actions, with a focus on holding during downward trends, while executing fewer buy/sell signals around significant price movements. In contrast, the RSI strategy mainly issues buy signals, particularly during the price's downward trend, trying to capture oversold conditions, but remains conservative with fewer overall trades compared to RL.

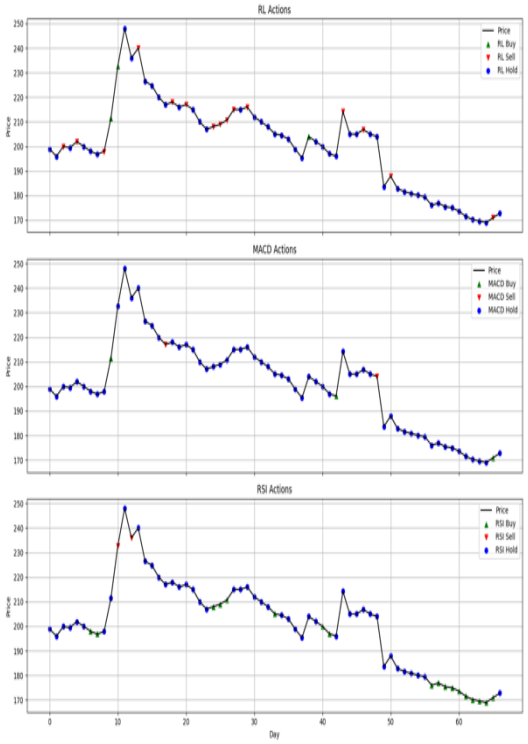

Fig. 6. Price Trends of UPPER Stock with Actions Taken by the RL Agent VS Other Indicators

The graph compares RL, MACD, and RSI trading strategies on UPPER stock over 63 days. The RL agent is highly active, frequently buying and selling in response to price changes, especially during declines. MACD takes a more conservative approach, with fewer actions, mostly holding through trends and reacting only to significant price shifts. RSI focuses primarily on buy signals during price drops, indicating attempts to capture oversold conditions but trades less frequently than RL. Overall, RL is the most responsive, while MACD and RSI are more selective in their trading actions.

4.1. Result Summary

From the table, it is evident that the Reinforcement Learning (RL) model significantly outperforms the traditional MACD and RSI indicators in terms of rewards. For the NTC stock, the RL reward is 37149.90, while the MACD and RSI indicators produce rewards of 781.50 and -8334.50, respectively. Similarly, for the NABIL stock, the RL reward is much higher at 25456.80, with MACD at 449.20 and RSI at -15294.60. The UPPER stock shows a similar trend, where RL results in a reward of 2465.10, significantly outperforming MACD (-218.00) and RSI (-2863.80). These results indicate that the RL agent was able to make better decisions in terms of maximizing rewards as compared to traditional indicators, which may not adapt as well to changing market conditions.

Table 1. Result Summary of different stocks

Stock	RL Reward	MACD Reward	RSI Reward
NTC	37149.90	781.50	-8334.50
NABIL	25456.80	449.20	-15294.60
UPPER	2465.10	-218.00	-2863.80

5. Conclusions and Future work

The study demonstrates that reinforcement learning (RL)-based trading strategies significantly outperform traditional indicators like MACD and RSI in terms of profitability and adaptability. The RL agent effectively learned from historical price patterns, dynamically adjusting its trading actions (buy, sell, hold) based on stock price movements, and consistently identified profitable opportunities while avoiding low-profit periods. It achieved higher rewards across multiple stocks, showcasing robustness in various market conditions, unlike the inconsistent results of traditional methods. The adaptive and continuously improving nature of RL highlights its potential as a promising tool for developing dynamic and profitable stock trading strategies, with further refinements expected to enhance its performance and applicability.

The study identifies several areas for enhancing RL in stock trading:

1. **Model Enhancements:** Future work could explore advanced RL algorithms like DQN, PPO, or A2C for better scalability and handling complex environments.

2. **Feature Engineering:** Expanding features to include technical indicators (e.g., Bollinger Bands, stochastic oscillators), fundamental data (e.g., earnings, interest rates), and real-time insights from news sentiment and social media.

3. **Training and Market Scope:** Extend training with larger datasets to capture diverse market phases and apply the model to broader markets, including multi-asset classes like currencies and cryptocurrencies.

4. **Risk Management:** Integrate risk controls such as drawdowns, stop-loss strategies, and position sizing for balanced trading.

5. **Portfolio Optimization:** Develop multi-asset portfolio strategies to dynamically balance risk and reward, improving practical applicability.

References

1. V. Drakopoulou, “A review of fundamental and technical stock analysis techniques,” Journal of Stock & Forex Trading, vol. 5, 2021.

2. J. Clifton and E. Laber, “Q-learning: Theory and applications,” Annual Review of Statistics and Its Application, vol. 7, pp. 279–301, 2020.

3. H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in Proceedings of the AAAI conference on artificial intelligence, vol. 30, 2016.

4. A. C. Eberhart and R. J. Sweeney, “Does the bond market predict bankruptcy settlements?,” The Journal of Finance, vol. 47, no. 3, pp. 943–980, 1992.

5. R. F. Engle, “Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation,” Econometrica: Journal of the econometric society, pp. 987–1007, 1982.

6. K.-j. Kim, “Financial time series forecasting using support vector machines,” Neurocomputing, vol. 55, no. 1-2, pp. 307–319, 2003.

7. T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka, “Stock market prediction system with modular neural networks,” in 1990 IJCNN international joint conference on neural networks, pp. 1–6, IEEE, 1990.

8. Watkins, Christopher John Cornish Hellaby. "Learning from delayed rewards." (1989).

9. Li, Jinyang. "A Deep Reinforcement Learning Framework For Financial Portfolio Management." arXiv preprint arXiv:2409.08426 (2024).

10. Yu, Pengqian, et al. "Model-based deep reinforcement learning for financial portfolio optimization." RWSDM Workshop, ICML. Vol. 1. 2019.

11. Fu, Kui, Yidong Yu, and Bing Li. "Multi-Feature Supervised Reinforcement Learning for Stock Trading." IEEE Access (2023).

12. Z. Jiang, D. Xu, and J. Liang, “A deep reinforcement learning framework for the

13. financial portfolio management problem,” arXiv preprint arXiv:1706.10059, 2017.