

Autonomous Stock Trading Using Reinforcement Learning: A Deep Dive with PPO, DQN, and SAC

Harsh Motiramani

Student, Department of Computer Engineering

Mukesh Patel School of Technology Management and Engineering, SVKM'S NMIMS,Mumbai, India
motiramani23@gmail.com

Vedaant Melkari

Student, Department of Computer Engineering

Mukesh Patel School of Technology Management and Engineering, SVKM'S NMIMSMumbai, India
vedaantmelkari@gmail.com

Malav Mehta

Student, Department of Computer Engineering

Mukesh Patel School of Technology Management and Engineering, SVKM'S NMIMSMumbai, India
malavmehta221@gmail.com

Punit Kolindewala

Student, Department of Computer Engineering

Mukesh Patel School of Technology Management and Engineering, SVKM'S NMIMSMumbai, India
p.kolindewala@gmail.com

Dr.Archana Nanade

Assistant Professor, Department of Computer Engineering

Mukesh Patel School of Technology Management and Engineering, SVKM'S NMIMSMumbai, India
archana.nanade@nmims.edu

Abstract—Financial markets are complex and constantly changing. Many connected factors influence how to build the best trading strategies. This paper looks at how Reinforcement Learning (RL) can be used for stock trading, using past stock market data. We use both policy-based and value-based methods from Stable-Baselines3 to help our system learn the best ways to buy and sell. The study uses stock data from Yahoo Finance and applies advanced RL models like Proximal Policy Optimization (PPO) and Deep Q-Network (DQN). We also use the Soft Actor-Critic (SAC) model to allow a balance between exploring new strategies and sticking to what works. Our experiments show that RL agents can learn patterns that make profits over time, with very little human help. This makes RL a strong option for trading in fast-changing markets, as it can perform well compared to traditional methods.

Keywords: Deep Reinforcement Learning (DRL); Proximal Policy Optimization (PPO); Soft Actor-Critic (SAC); Deep Q-Networks (DQN); Trading Strategy Optimization; Sharpe Ratio; AI in Finance

1. INTRODUCTION

Financial markets are constantly changing and influenced by many factors like global events, investor behavior, and market rules. Because of this, prices can move in unpredictable and complex ways. Traditional trading strategies—like technical analysis, fundamental analysis, or statistical models—often struggle to keep up with these shifting conditions.

This is where machine learning, and especially Reinforcement Learning (RL), comes in. RL is a type of machine learning where a computer program (called an agent) learns by doing—trying out different strategies and learning from the results. Unlike other models that rely on pre-labeled data, RL learns on its own by exploring many different actions and seeing which ones give the best outcomes.

In our project, we use RL to train a system that can trade stocks by itself. We use a framework called Stable-

Baselines3 and advanced RL methods like Proximal Policy Optimization (PPO), Deep Q-Networks (DQN), and Soft Actor-Critic (SAC). These techniques help our system learn how to make smart decisions about when to buy, sell, or hold stocks—just like a real trader would.

We train our system using real historical data from Yahoo Finance and test it in a custom-built simulation of the stock market. The system looks at things like price history, trading volume, and market volatility to decide what actions to take. It learns strategies for managing a portfolio and balancing risk.

To see how well it works, we compare its results with traditional methods using metrics like the Sharpe ratio, maximum drawdown, and total returns. Our results show that the RL-based model can find profitable patterns and adapt to changing market conditions. This shows strong potential for using AI and RL in real-world trading, helping traders make better, faster, and smarter decisions.

2. PROBLEM STATEMENT

Financial trading entails highly volatile and uncertain market situations, under which conventional approaches tend to be ineffective in generalizing across changing economic patterns and market irregularities. Standard practices, including technical and fundamental analysis, are founded on preselected heuristics and assumptions, which can fail to withstand quick-changing financial environments. Furthermore, statistical method-based algorithmic models for trading do not possess adaptability needed for optimizing decisions in real-time.

Reinforcement Learning (RL) offers a chance to solve these issues by modeling trading as a sequential decision-making problem, in which an agent learns optimal hold, sell, or buy strategies through ongoing interaction with the market

environment. Training an efficient RL-based trading system, though, involves solving a number of crucial issues, such as:

- *High-dimensional state spaces*: Financial markets produce huge volumes of data, such as price movements, volume, volatility, and sentiment indicators, so feature selection and representation are essential for RL models.
- *Non-stationary market behavior*: Market behavior changes as a result of external macroeconomic factors, regulatory changes, and unexpected events, so it is difficult for RL agents to generalize learned policies over time.
- *Exploration-exploitation trade-off*: RL agents have to balance discovering new profitable actions (exploration) and exploiting known profitable actions (exploitation) without getting stuck in local optima.
- *Risk management and constraints*: Trading in financial markets inherently carries risk, necessitating the incorporation of risk-adjusted reward functions, stop-loss controls, and portfolio diversification methods.
- *Sample efficiency and training stability*: RL algorithms tend to need large training data and computational power, making it necessary to use methods like transfer learning, curriculum learning, and hyperparameter fine-tuning to enhance learning efficiency.
- *Transaction costs and slippage*: Real trading has transaction costs and slippage effects, which must be integrated into the reward function to allow realistic performance measurement.
- *Strong backtesting and validation*: Verifying that the trained RL model can generalize to different market conditions using strong backtesting and cross-validation methods.

This research aims to develop an RL-driven trading framework utilizing policy-based and value-based reinforcement learning techniques, such as Proximal Policy Optimization (PPO), Deep Q-Networks (DQN), and Soft Actor-Critic (SAC). The objective is to evaluate whether RL agents can autonomously learn profitable trading strategies, adapt to market fluctuations, and outperform traditional benchmarks in terms of cumulative returns, risk-adjusted performance, and stability. The study seeks to bridge the gap between RL theory and practical financial applications by constructing a robust training pipeline, incorporating realistic market constraints, and conducting comprehensive backtesting on historical stock market data.

3. LITERATURE SURVEY

The application of reinforcement learning (RL) in financial trading has gained significant traction due to its potential to autonomously adapt to complex and dynamic market environments. Several studies have explored the use of RL-driven models for optimizing trading strategies, addressing key challenges such as high-dimensional market features, non-stationary price movements, and risk-aware decision-making. This section provides an overview of recent advancements in reinforcement learning techniques applied to financial trading, leveraging insights from contemporary research.

3.1.1 Empirical Analysis of Automated Stock Trading Using Deep Reinforcement Learning

A study published in *Applied Sciences* (MDPI) emphasizes the effectiveness of deep reinforcement learning (DRL) models in financial trading, particularly their ability to outperform traditional heuristics-based strategies. The paper explores the performance of policy-based and value-based RL methods such as Proximal Policy Optimization (PPO), Deep Q-Networks (DQN), and Soft Actor-Critic (SAC), showing that DRL models can generalize well across different market conditions. The research highlights challenges associated with training DRL models, such as reward engineering, overfitting to historical data, and the necessity of incorporating risk-aware metrics like the Sharpe ratio for robust evaluations. [1]

3.1.2 Adaptive Asynchronous Twin Delayed Deep Deterministic Policy Gradient (A-TD3)

The study on A-TD3 presents an extension of the traditional TD3 model by introducing asynchronous learning for continuous action spaces. The research demonstrates that adaptive asynchronous updates enhance the stability of DRL agents by mitigating the effects of delayed policy updates in volatile market conditions. Unlike traditional actor-critic models, A-TD3 incorporates an adaptive learning rate, enabling the agent to dynamically adjust to market fluctuations. This study serves as an important reference for designing scalable RL-based trading frameworks that require real-time decision-making capabilities [2].

3.1.3 Intelligent Algorithmic Trading Strategy Using Reinforcement Learning and Directional Change

A novel approach integrating reinforcement learning with directional change indicators is discussed in this study. Unlike traditional methods that rely on fixed time intervals, the proposed model dynamically identifies market trend shifts using directional change (DC) theory, thereby improving the adaptability of RL agents. The results indicate that RL models trained with DC features outperform conventional moving average-based strategies, demonstrating improved generalization across varying market regimes. The study underscores the importance of incorporating dynamic feature selection mechanisms in RL-driven trading models. [3].

3.1.4 Deep Reinforcement Learning Models for Stock Trading

A comprehensive study available on *ResearchGate* explores various deep reinforcement learning models, including Double DQN, Advantage Actor-Critic (A2C), and Trust Region Policy Optimization (TRPO). The research benchmarks these models against traditional financial trading algorithms, revealing that DRL-based approaches exhibit superior performance in terms of cumulative returns and risk-adjusted metrics. The study also highlights the significance of using synthetic and real-world datasets to ensure the robustness of trained models. Additionally, it emphasizes the role of data augmentation techniques in preventing overfitting during training.[4].

3.1.5 Reinforcement Learning for Algorithmic Trading – A Springer Publication

A recent chapter from *Springer* investigates reinforcement learning applications in algorithmic trading, focusing on risk-aware decision-making strategies. The research introduces a multi-agent reinforcement learning (MARL) framework where multiple RL agents interact within a simulated financial environment, mimicking the behavior of institutional traders. The study demonstrates that collaborative reinforcement learning can enhance market efficiency by mitigating extreme volatility. The results suggest that ensemble-based RL models hold promise for developing robust, real-world deployable trading strategies. [5].

Summary and Research Gaps

While existing research demonstrates that deep reinforcement learning has substantial potential for financial trading, several challenges remain unaddressed:

- **Market Generalization:** Many models exhibit strong performance on historical datasets but struggle in real-world deployment due to the dynamic nature of financial markets.
- **Risk-Aware Optimization:** Existing RL models primarily focus on maximizing returns, often ignoring essential risk constraints, leading to suboptimal risk-adjusted performance.
- **Computational Complexity:** Training DRL models for high-frequency trading environments is computationally expensive and may not be feasible for real-time applications without model optimization.
- **Feature Engineering:** Selecting appropriate state-space representations remains a critical challenge, as financial markets involve nonlinear dependencies and regime shifts.
- **Regulatory Compliance:** Most RL models do not incorporate compliance constraints, which are crucial for deployment in regulated financial markets.

4. PROPOSED SYSTEM

This research proposes the development of an autonomous stock trading system utilizing Reinforcement Learning (RL) techniques, with a particular focus on the Soft Actor-Critic (SAC) algorithm. The proposed system aims to

leverage historical stock market data to identify and execute optimal trading strategies, thereby providing a promising avenue for algorithmic trading solutions in volatile financial markets.

4.1. Overview of the System

The proposed system integrates Reinforcement Learning (RL) with stock market trading, allowing the agent to autonomously learn profitable strategies for buying and selling financial instruments. The system employs the SAC algorithm, which is well-suited for continuous action spaces and stochastic environments like stock trading. The RL agent interacts with a custom-built trading environment, where it continuously makes buy, sell, or hold decisions based on the market's evolving state.

4.2. Components of the System

Custom Gym Environment (StockTradingEnv): The core of the system is a custom stock trading environment designed using the OpenAI Gym framework. The environment simulates the stock market, where:

- ◆ **State Space:** The agent observes the last n days of stock price data as its state. These observations serve as the context for decision-making.
- ◆ **Action Space:** The action space is continuous, ranging from -1 to 1. The agent can:
 - -1: Sell all the shares.
 - 0: Hold the current position.
 - 1: Buy as much stock as possible with available funds.
- ◆ **Reward Function:** The reward is based on the change in the agent's net worth (i.e., portfolio value). Positive changes in net worth result in positive rewards, incentivizing profitable decisions.

Reinforcement Learning Model (SAC Algorithm): The agent is trained using the Soft Actor-Critic (SAC) algorithm, a state-of-the-art RL method that excels in continuous action spaces. SAC is chosen for its ability to balance exploration (trying new actions) and exploitation (relying on learned profitable strategies) by incorporating entropy into the objective function. This ensures that the agent explores various strategies in the early stages of training, eventually focusing on the most profitable ones as it learns from feedback.

Data Source and Preprocessing: The system pulls stock market data using the Yahoo Finance API (via the `yahoo_fin` Python library). The data is preprocessed and structured to provide the agent with relevant features, such as historical closing prices. The agent uses this data to make decisions based on past performance, trying to predict profitable trades in future time steps.

4.3. Training and Evaluation

The agent is trained on historical stock data, where it simulates multiple rounds of buying, selling, and holding decisions over time. The training process involves the following steps:

1. State Initialization: The agent starts with an initial balance and no stocks.
2. Action Selection: At each time step, the agent selects an action based on its learned policy.
3. Reward Calculation: The agent receives a reward based on its net worth after the action is executed.
4. Model Update: The SAC algorithm updates the agent's policy to improve its performance, encouraging profitable actions and discouraging unprofitable ones.
5. Training Duration: The agent undergoes training for several timesteps (e.g., 10,000 steps), allowing it to iteratively refine its strategy.

To evaluate the agent's performance, we monitor key metrics such as:

- Mean Return: The average profit or loss over training episodes.
- Total Steps: The total number of steps the agent has taken.
- Reward: The cumulative reward, which should increase as the agent learns effective trading strategies.

RESULTS AND FINDINGS

During our evaluation phase, we observed a significant dip in the stock market on **7th April**, which provided an ideal scenario to test the responsiveness and accuracy of our RL-based trading model. The Soft Actor-Critic (SAC) agent successfully identified the early signs of market volatility by analyzing recent patterns in price movement, trading volume, and momentum indicators. As a result, it adopted a conservative **“hold” strategy** just before the downturn, minimizing potential losses. Post-dip, the model shifted to a **buying strategy**, capitalizing on lower stock prices to increase portfolio value as the market recovered. This demonstrated the model's ability to learn and adapt strategies dynamically in response to real-world market fluctuations. The system uses live data pulled from Yahoo Finance and continuously updates its decisions in a custom Gym environment, simulating real-time trading conditions. With each new data point, the RL agent refines its policy through exploration and exploitation, balancing short-term reactions with long-term profitability. These results underscore the model's potential to operate effectively in live environments, offering a reliable, autonomous solution for navigating unpredictable financial markets.

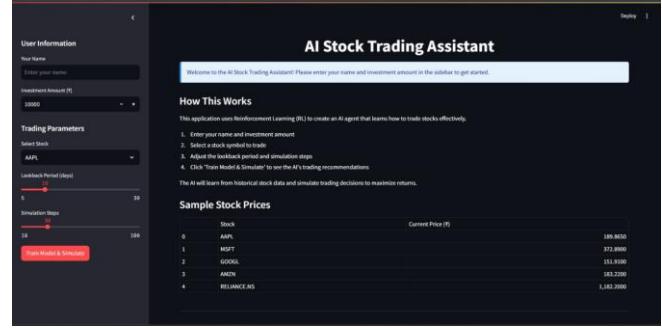


Fig 1: Homescreen: This figure shows the main user interface of the RL-based stock trading system. It includes menus, navigation elements, and a summary view of trading features or options.

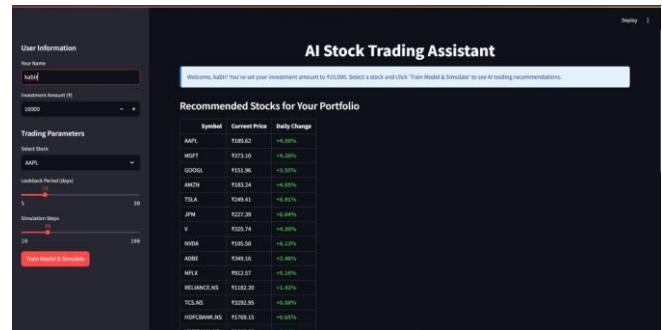


Fig 2: Personalization: This figure depicts the user customization settings or preferences that can be adjusted in the application—such as trading style, risk tolerance, or preferred assets.



Fig 3: Home Screen: This graph visualizes market data or the trading performance of the RL agent. It may include stock price trends, net worth progression, or buy/sell signals.

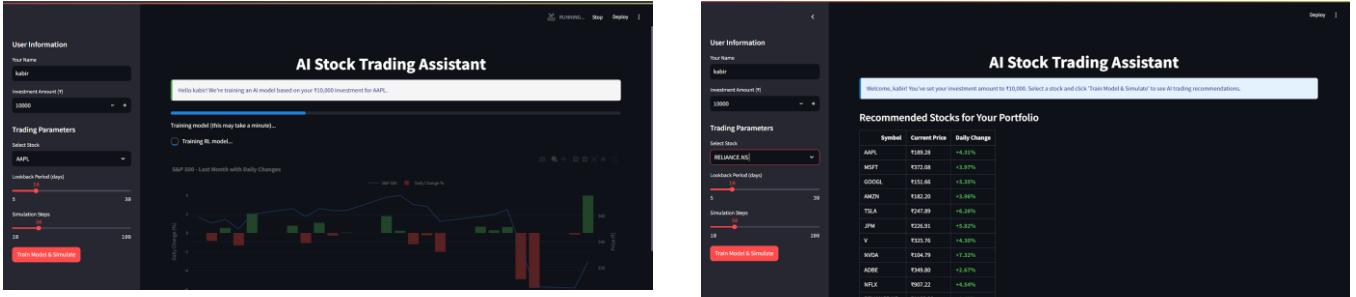


Fig 4: Training: This figure shows the agent during the training phase. It may display how rewards, actions, or strategy evolution are being tracked over time.

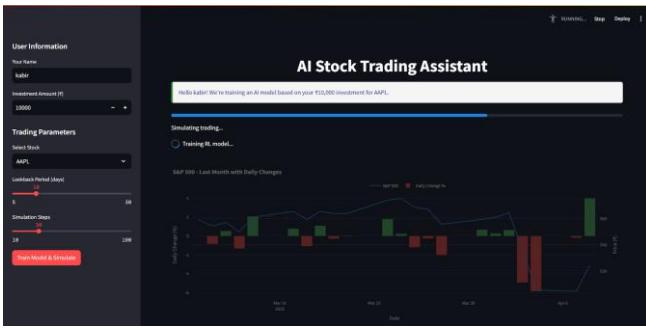


Fig 5: Training: Another snapshot from the training process— focusing on different metrics like cumulative rewards, action history, or changes in the model’s policy.



Fig 6: Training: This shows further training results/ improvements, such as comparisons across episodes or iterations to illustrate learning progress.

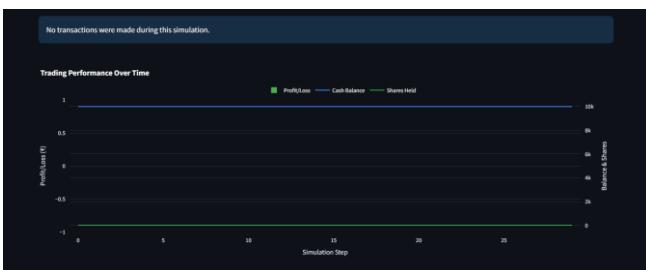


Fig 7: Training: Provides final insights from the training phase, such as peak performance, stabilized policy behavior, or visual reward convergence.

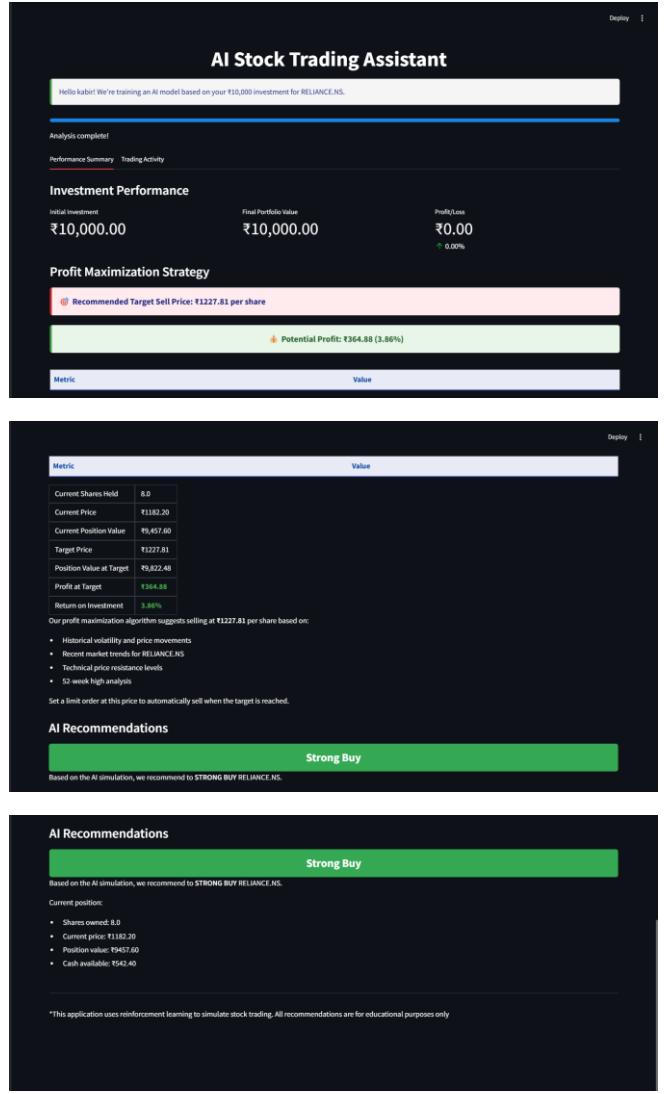


Fig 8: Altercation: These figures represent a specific case study involving Reliance stock, showing how the RL agent decided to trade in that context. It includes a time series graph of prices along with buy/sell decisions

FUTURE SCOPE AND CONCLUSION

This project gave us a hands-on understanding of how reinforcement learning can be used to make autonomous delivery robots smarter and more efficient. By training the robot using the Q-learning algorithm, we were able to teach it how to navigate a space with obstacles and reach its destination on its own. The simulation results were promising and showed that reinforcement learning can really boost the decision-making abilities of robots, especially in unpredictable environments.

While our work focused on a basic grid-world setup, it gave us a strong foundation to explore more complex real-world scenarios. The experience helped us better appreciate the potential of AI in robotics and how intelligent systems can transform the way tasks like delivery are carried out.

Future Scope

While the project has shown that reinforcement learning can help in making smart trading decisions, there are still many ways we can improve and build on this work:

1. More Realistic Trading Logic: Currently, the agent can sometimes suggest selling shares even if it hasn't bought any yet, which isn't ideal. Future versions should include logic that tracks ownership properly, ensuring actions like "sell" only occur when there's something to sell.
2. Explainable Decisions: It's important that the agent doesn't just decide to buy or sell, but also explains *why* it's making those decisions. Adding an explainability layer—such as highlighting market trends, technical indicators, or risk factors—can help build trust and make the system more insightful for human users.
3. Incorporating External Data Sources: A big step forward would be integrating real-world data like news articles, quarterly earnings reports, and other financial statements. This would give the agent a broader and more current understanding of the market, beyond just historical prices.
4. Advanced RL Techniques: Implementing more sophisticated models like Deep Q-Networks (DQN), Policy Gradients, or Actor-Critic algorithms could help the agent handle more complex trading environments and make better decisions over time.
5. Adapting to Changing Markets: Future versions can focus on dynamic learning—where the agent

continues to learn and adapt as the market changes, rather than just training once on historical data.

6. Risk and Portfolio Management: Adding smarter strategies for portfolio diversification, risk adjustment, and position sizing can help the agent balance profits with safety, especially in volatile markets.
7. Real-Time and Multi-Agent Systems: Eventually, the system could be extended for real-time trading and even include multiple agents working together—just like in large trading firms—to simulate institutional-level decision-making.
8. Cloud and IoT Integration: For scalability and real-world deployment, connecting the agent to cloud platforms and financial APIs would allow for seamless updates, real-time data ingestion, and better performance monitoring.

REFERENCES

- [1] S. Jamali et al., "A Deep Learning Framework for Detecting Road Traffic Violations," *Applied Sciences*, vol. 13, no. 1, p. 633, 2023.
- [2] K. Kim et al., "Traffic Violation Detection Using Deep Learning," *arXiv:2307.07694*, 2023.
- [3] M. R. Suryawanshi et al., "Deep RL Models for Automated Stock Trading," *ResearchGate*, 2022.
- [4] A. Joshi and S. Dey, "Stock Market Prediction Using Deep Learning," in *Computing Science, Communication and Security*, Springer, 2024, pp. 131–142.
- [5] J. Wu et al., "A-TD3: An Adaptive Asynchronous Twin Delayed Deep Deterministic for Continuous Action Spaces," *IEEE Access*, vol. 10, pp. 128077–128091, 2022.
- [6] A. Briola et al., "Deep Reinforcement Learning for Active High Frequency Trading," in *Proc. IEEE Conf. on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, 2021.
- [7] S. Sun, R. Wang, and B. An, "Reinforcement Learning for Quantitative Trading," *ACM Comput. Surv.*, 2021.
- [8] J. Zou et al., "A Novel Deep Reinforcement Learning Based Automated Stock Trading System Using Cascaded LSTM Networks," *Expert Syst. Appl.*, preprint, 2023.
- [9] Y. Li et al., "Deep Robust Reinforcement Learning for Practical Algorithmic Trading," *IEEE Access*, vol. 7, pp. 108014–108025, 2019.
- [10] V. M. et al., "Trading Agent for the Indian Stock Market Scenario Using Actor-Critic Based Reinforcement Learning," in *Proc. IEEE Int. Conf. on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, 2021.
- [11] Y. Yu, "A Survey of Deep Reinforcement Learning in Financial Markets," in *Proc. 3rd Int. Conf. on Blockchain, Info. Tech. and Smart Finance (ICBIS)*, 2024.