IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.1412132

AI Driven Trading Bot for Intelligent Decision-Making Using ML and RL Model

Nithin Gowda N[1], Mrs Rekha S[2], S Praveen Kumar[3], Shashank S[4], Venudharshan M[5]

Department of Information Science and Engineering, The Oxford College of Engineering,

Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, India[1-5]

Abstract: Financial markets generate large volumes of fast- moving data and require decisions to be taken in very short intervals. Human traders alone struggle to monitor all signals and react consistently without emotional bias. To address this challenge, this work presents an AI driven trading bot that combines Machine Learning (ML) for price forecasting with Reinforcement Learning (RL) for action selection. The system uses technical indicators, a Long Short-Term Memory (LSTM) network for short-term prediction and a Deep Q-Network (DQN) agent to learn profitable buy, sell and hold policies. The complete solution is deployed as a web application that provides real- time charts, portfolio analytics, sentiment summaries and AI- generated trading signals. Experimental evaluation indicates promising accuracy, low-latency inference and improved profit consistency when compared with simple rule-based strategies.

Keywords: Stock Market, Machine Learning, Reinforcement Learning, LSTM, DQN, Trading Bot, Financial Analytics.

I. INTRODUCTION

Stock markets are highly dynamic and influenced by a combination of economic, political and psychological factors. Prices react to news, global events and investor sentiment, making short-term prediction difficult. Traditional approaches rely heavily on manual chart reading and heuristics, which can lead to delayed responses and inconsistent decisions. In parallel, advances in Artificial Intelligence (AI), especially ML and RL, have enabled systems that can learn patterns from historical data and adapt to changing environments. ML models can estimate the direction or magnitude of price movement, while RL agents can learn trading policies that maximize long-term returns. This project explores the integration of: • ML models for short-horizon price prediction, • RL (DQN) for trading decision-making, • Technical indicators for feature enrichment, • A web-based front-end for interactive analytics. The goal is not to replace human traders completely, but to provide a decision-support system that delivers data-driven, consistent and explainable suggestions.

II. PROBLEM STATEMENT AND OBJECTIVE

Financial markets generate massive volumes of real-time data, making it increasingly difficult for traders to analyze market behavior manually and make timely, profitable decisions. Traditional algorithmic trading relies on predefined rules or static strategies that often fail to adapt to rapidly changing market conditions, unpredictable price movements, and complex non-linear patterns. As a result, traders face challenges such as inconsistent performance, inability to generalize strategies across market regimes, and limited capability to learn from historical and live data interactions.

There is a critical need for an intelligent, adaptive trading system capable of autonomously learning market dynamics, optimizing decision-making, and executing trades with minimal human intervention. By leveraging machine learning and reinforcement learning, such a system can continuously improve from market feedback, identify profitable opportunities, minimize risks, and enhance trading efficiency.

Objectives:

To design and develop an AI-driven trading bot capable of processing historical and real-time market data to support informed trading decisions.

To implement machine learning and reinforcement learning models that can autonomously learn trading strategies and adapt to changing market conditions.

To evaluate and optimize the bot's decision-making performance using metrics such as accuracy, reward maximization, drawdown reduction, and risk-adjusted return.

To enable automated trade execution through integration with brokerage APIs, ensuring minimal latency and high reliability.

To provide real-time insights, predictive signals, and strategy recommendations that improve profitability while controlling exposure to market volatility and risk.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.1412132

III. SCOPE

This project aims to build a trading bot powered by AI, using machine learning and reinforcement learning techniques to improve decision-making in financial markets. It includes gathering historical and real-time market data, such as price movements, trading volume, and technical indicators from reliable financial data sources. The data will be cleaned and transformed to create predictive models that spot trends, generate trading signals, and forecast future price behavior.

We will use reinforcement learning algorithms to help the bot learn trading strategies by interacting with a simulated or live market environment. This learning process enables the system to adjust its strategies based on feedback, which helps increase profits while reducing risk. The bot will support automated or semi-automated trade execution by integrating with trading or brokerage APIs, allowing it to place buy or sell orders based on model-driven decisions.

We will evaluate performance through backtesting and measuring metrics like return on investment, Sharpe ratio, drawdown, and win-loss accuracy. The project also includes visualizing trading signals, performance reports, and market analytics through a user interface or dashboard for monitoring and validation. However, high-frequency trading infrastructure, regulatory compliance, and cross-market arbitrage are not part of the current project scope.

IV. LITERATURE REVIEW

[1] **Zhang et al.** developed a hybrid ML-based price forecasting model using LSTM networks integrated with sentiment analysis from financial news. Their approach improves short-term prediction accuracy and reduces lag effects, but suffers reduced reliability during abrupt market shocks and black-swan events.

[2] **Patel et al.** introduced a reinforcement-learning trading agent using Deep Q-Networks (DQN) that adjusts trading positions dynamically based on reward signals. Backtesting on equity indices demonstrated improved cumulative returns, though the model occasionally overfitted volatile market segments.

[3] **Huang et al.** proposed a PPO-based reinforcement-learning strategy for intraday trading that incorporates risk-weighted rewards. Results show better drawdown control and stable convergence, but computational cost increases with high-frequency feature updates.

[4] **Mishra et al.** designed an ensemble ML pipeline combining Random Forest, XGBoost, and SVM for trend classification and signal confirmation. Their bot reduced false trades by leveraging model consensus, although ensemble complexity led to slower inference times during live execution.

[5] **Khan et al.** implemented a multi-agent RL framework where different agents specialize in market regimes such as bullish, bearish, and sideways. Regime-aware switching improved adaptability, yet agent coordination occasionally produced conflicting signals.

[6] **Ocampo et al.** utilized evolutionary reinforcement learning to optimize hyperparameters and reward shaping automatically. The system enhanced long-term profitability and reduced manual tuning effort, but required extensive training periods and high compute resources.

[7] **Singh et al.** developed a transformer-based market forecasting model feeding RL policies for trade execution. Transformers captured long-range dependencies in price series, boosting signal stability; however, inference latency posed challenges for near real-time execution.

[8] **Liu et al.** proposed an anomaly-aware RL trading bot that identifies manipulated or abnormal market conditions before executing trades. Their results show reduced exposure during flash crashes, though early anomaly detection occasionally blocked profitable opportunities.

[9] **Ahmed & Roy** integrated risk metrics such as Sharpe ratio and maximum drawdown directly into the RL reward function. The enhanced reward shaping produced balanced risk-return profiles, but risk-sensitive tuning reduced trade frequency and potential gains during high-momentum periods.

[10] **Nayak et al.** conducted a comprehensive survey on ML- and RL-powered trading systems, outlining architectures, risk-control mechanisms, execution pipelines, and evaluation benchmarks. They highlighted research gaps including online learning under real-time market shifts and safe RL to mitigate catastrophic loss.

4.1 Gaps or Areas for Improvement

Despite significant advances in using machine learning and reinforcement learning for automated trading, several important gaps still exist. Current models often struggle to perform consistently during sudden market changes because they rely heavily on historical data and lack methods for stable online adaptation. This reliance often results in overfitting, where models identify past trends that do not hold up during real-time fluctuations. Reinforcement learning systems also face issues with reward structures that focus on profit while overlooking risk exposure, sensitivity to volatility, and control over drawdowns, which limits their effectiveness.

Efficiency in execution is another challenge. Deep learning architectures introduce delays in processing that hinder real-time trading. Additionally, the complex nature of these models makes them hard to understand, which complicates

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.1412132

regulatory compliance and reduces trust among analysts. Problems like data noise, manipulation, and anomalies are still not fully addressed, which weakens how well models can handle unusual or hostile market conditions. Generalizing strategies across different assets is also challenging, as those designed for one market often perform poorly in others, like forex or cryptocurrency.

There are ongoing issues with evaluation due to fragmented datasets, differing assumptions about transaction costs, and a lack of standard performance measures. This makes it hard to draw comparisons. Overall, we need to improve dynamic learning, focus on risk-aware optimization, enhance robustness against noise, increase interpretability, and create unified evaluation standards to develop dependable and scalable trading automation.

V. SYSTEM ARCHITECTURE

The AI trading bot is implemented as a web-based system with separate front-end, back-end and model layers. The front-end displays charts, predictions, sentiment and signals, while the back-end orchestrates data fetching, model execution and storage..

This hierarchical training approach enhances model efficiency, significantly cuts down communication costs, and supports scalability far better than traditional flat federated or centralized methods. To ensure the authenticity Regarding the model's integrity updates, the system employs a blockchain layer where every update is validated and immutably recorded using smart contracts. A central authority is no longer necessary thanks to this decentralization, which also shields the system from malicious attacks or tampering that can jeopardize the accuracy of the model or the reliability of the data. Complementing blockchain is the Interplanetary File System (IPFS), used for decentralized storage of validated model updates, anomaly reports, and other critical system information. IPFS ensures fault tolerance, high availability, and rapid data retrieval, even if parts of the network fail, thereby reinforcing system robustness. A dedicated anomaly detection and self-healing layer continuously monitors network and device activities for abnormal patterns such as unauthorized access or malfunctioning nodes. When issues are detected, the system autonomously isolates compromised devices, initiates restorative actions like resets or reconfigurations, and reintegrates the devices once stability is achieved. This reduces downtime and enhances the network's overall resilience. Finally, system administrators are supported by a user-friendly, web-based dashboard that provides real-time visualization of blockchain transactions, security threats, device performance, and mitigation efforts. This interface enables proactive management and transparent oversight, empowering administrators to maintain optimal system health and security.

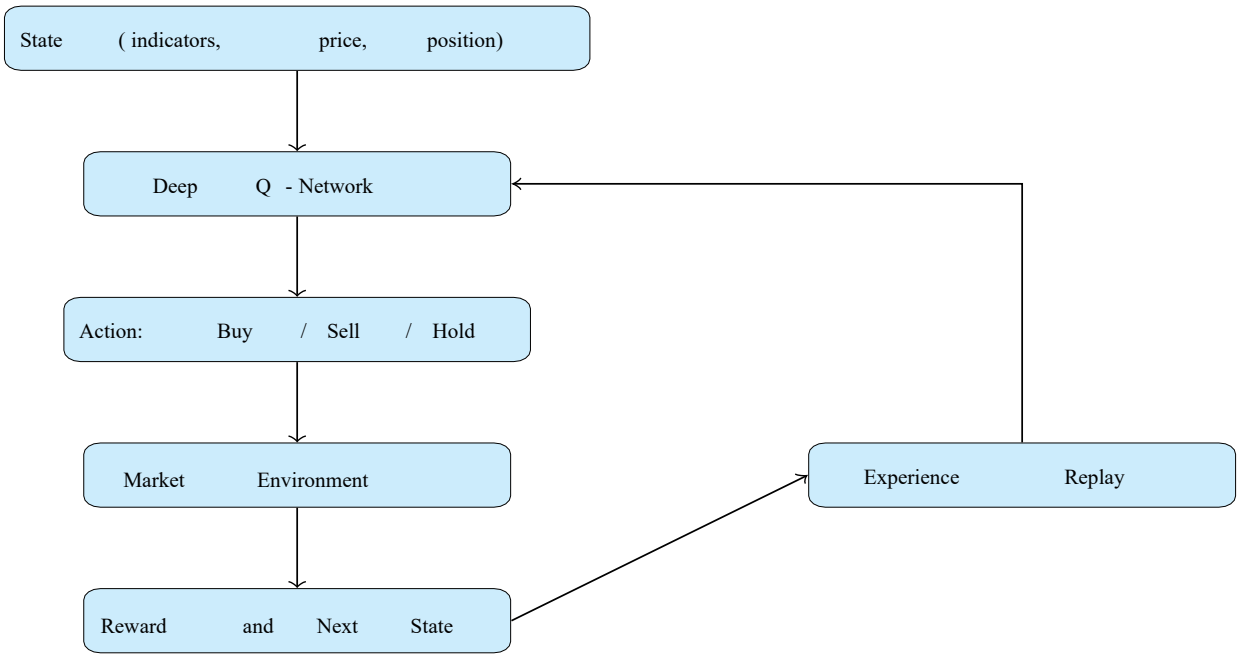


Figure 1.DQN-based reinforcement learning workflow for trading decisions

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.1412132

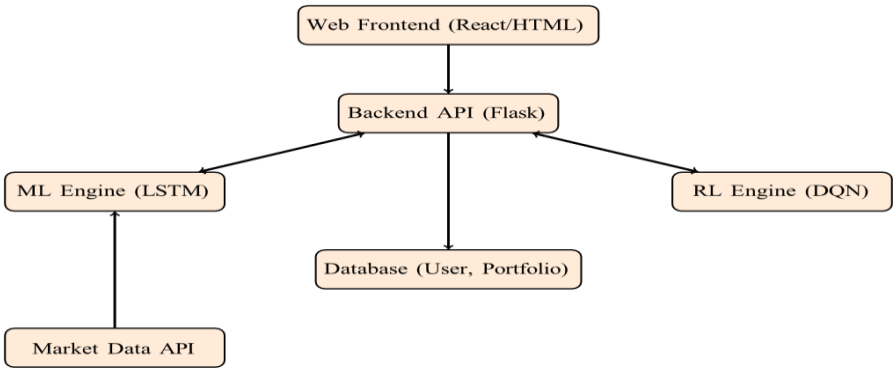


Fig. 2. System architecture of the AI-driven trading bot.

VI. METHODOLOGY

The proposed trading bot follows a layered methodology that starts with data collection and ends at real-time decision support. The main stages are shown conceptually .

A. Overall Workflow

B. Data Collection and Preprocessing

Historical and live price data are obtained using online APIs such as Yahoo Finance. For each symbol, the following attributes are used:

- Open, High, Low, Close (OHLC) prices,
- Adjusted close price, • Trading volume,
- Timestamp.

Data preprocessing includes:

- Handling missing records by interpolation or removal,
- Filtering obvious outliers,
- Applying scaling (e.g., Min–Max normalisation) for neural network inputs,

C. Technical Indicator Computation

Several technical indicators are computed to capture trend, momentum and volatility:

- Simple Moving Average (SMA),
- Exponential Moving Average (EMA),
- Relative Strength Index (RSI),
- Moving Average Convergence Divergence (MACD),
- Bollinger Bands.

These indicators are appended as additional features to the input vectors used by the ML and RL components, improving both interpretability and model performance

6.1. Requirement Analysis and Problem Framing

Define project goals, including target markets, asset classes, trading style, and risk thresholds. Decide if the system will operate independently or serve as a decision-support tool. Consider costs like brokerage fees, slippage tolerance, and liquidity constraints. Identify regulatory requirements and set limits to prevent unethical algorithmic practices.

6.2. Data Acquisition and Preprocessing

Gather historical time-series data such as OHLC prices, volumes, and volatility metrics from APIs. Obtain real-time data streams for ongoing analysis. Clean and standardize the data by normalizing, removing outliers, and handling missing values. Create rolling-window datasets to maintain temporal consistency and split them into training, validation, and forward-testing sets to lower overfitting.

6.3. Feature Engineering and Signal Construction

Generate indicators such as RSI, ATR, Bollinger Bands, MACD, momentum oscillators, and volatility clusters. Extract

DOI: 10.17148/IJARCCE.2025.1412132

sentiment indicators if relevant. Perform feature relevance analysis using mutual information, correlation matrices, or dimensionality reduction to remove redundancy. Create directional labels and trend classes for supervised learning.

6.4. Machine Learning-Based Prediction Module

Train models like LSTM, TCN, Gradient Boosting, or Random Forest to forecast price direction, volatility, or return size. Use rolling validation and optimize hyperparameters. Assess model stability and uncertainty to support risk-aware decision-making. Use prediction outputs as market signals or inputs for reinforcement learning.

6.5. Reinforcement Learning Strategy Optimization

Create a trading environment with defined states, actions, and rewards. Use RL algorithms like DQN, PPO, or SAC to learn the best buy/sell strategies. Include transaction costs, drawdown penalties, and risk-adjusted returns in reward functions. Train policies through episodic simulations and refine exploration strategies for flexibility.

6.6. Backtesting and Performance Evaluation

Test the ML-RL framework through multi-period backtesting to measure strength across various market conditions. Use metrics including cumulative return, Sharpe ratio, Sortino ratio, maximum drawdown, hit rate, and exposure time. Perform walk-forward analysis to check generalization to new data.

6.7. Integration with Brokerage or Trading Execution APIs

Connect the strategy to real or simulated trading environments through APIs, such as Zerodha Kite, Binance, or Alpaca. Add modules for order execution, position management, latency tracking, and real-time risk protection. Validate the execution pipeline using paper trading before controlled deployment.

6.8. Visualization, Monitoring, and Continuous Improvement

Create dashboards to track live signals, execution logs, portfolio growth, and risk indicators. Set up alert systems for issues like excessive drawdown or latency spikes. Retrain and enhance models regularly to keep up with changing market conditions and remain competitive.

VII. IMPLEMENATION ENVIRONMENT

The environment for the AI trading bot is set up to handle data reliably, train models at scale, create quality visualizations, and integrate machine learning workflows efficiently. We use Python as the main programming language because of its wide range of scientific computing libraries and strong support for data analytics and machine learning. Key libraries like NumPy and Pandas help us manage, transform, and preprocess large amounts of historical market data. SciPy provides statistical operations and feature calculations needed for analyzing financial time series. We develop machine learning models using frameworks like Scikit-Learn for traditional predictive models, and TensorFlow or PyTorch for deep learning when needed. These frameworks offer GPU acceleration for quicker experimentation. We generate visualizations with Matplotlib, Seaborn, and Plotly to create analytical charts, candlestick patterns, and trend visualizations during explorations and model evaluations. We manage the coding environment with Jupyter Notebook and Google Colab, allowing for iterative experimentation, inline graph rendering, and straightforward documentation of data and modeling processes.

For acquiring and storing data, the environment connects to external financial APIs such as Alpha Vantage, Yahoo Finance, or Binance based on the market type. It uses structured formats like CSV, Parquet, or SQL databases for long-term storage. We maintain version control using Git and GitHub to track code changes, store experimental results, and collaborate effectively as we scale the project. The operating system usually consists of Windows or Linux machines, with a preference for Linux in deployment scenarios because it works better with GPU workflows and executes tasks automatically. We use Python virtual environments or Conda to ensure consistent dependencies and avoid package conflicts during model testing. The reinforcement learning part is built using libraries like Stable Baselines, Gym, or custom simulation setups. This allows the bot to learn adaptive trading strategies through interactions based on historical market conditions. These setups enable the configuration of positions, trade actions, holding periods, and transaction fees to ensure realistic policy behavior.

We deploy and test the trading logic using controlled backtesting frameworks instead of live execution. This approach focuses on research and evaluation rather than automation in production. Backtesting tools like Backtrader or custom evaluation scripts simulate live trading scenarios with historical data. This lets us evaluate multiple strategies through different market phases while tracking metrics like cumulative returns, Sharpe ratio, drawdowns, and trade-level accuracy. We monitor model performance using dashboards created with Streamlit or Jupyter visualization tools, which enable

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025
DOI: 10.17148/IJARCCE.2025.1412132

ongoing inspection of prediction trends, signal reliability, and model sensitivity. Throughout the process, logging systems keep track of decisions made by the system, data issues, and model outputs to aid debugging and ongoing improvements. Overall, the environment provides a solid foundation for data engineering, developing learning algorithms, strategic assessment, and visualization, ensuring that the trading bot can be continuously improved for more accurate predictions, better decision-making support, and dependable performance assessments without direct market deployment.

VIII. MODULES

8.1 Market Data Acquisition Module

Collects historical and real-time financial market data from APIs. It manages data frequency such as daily, hourly, or minute-level. It also stores raw market streams for further processing.

8.2 Data Cleaning & Preprocessing Module

Handles missing values and fills gaps in time series. It removes anomalies, normalizes price data, and prepares datasets to ensure consistency across market conditions.

8.3 Feature Engineering & Indicator Generation Module

Generates technical indicators like moving averages, RSI, MACD, Bollinger Bands, volume signals, and volatility indexes. It also creates feature vectors to improve predictive learning.

8.4 Exploratory Data Analytics Module

Performs statistical analysis and trend exploration. This helps understand correlations, market behaviors, and patterns that influence model design and signal selection.

8.5 Machine Learning Prediction Module

Trains machine learning models to forecast short-term price direction or the likelihood of upward or downward movement. It provides predictive outputs for decision-making.

8.6 AI-Based Strategy Decision Module

Turns predictive outputs and market insights into actionable buy, hold, or sell recommendations. This is based on defined rules, thresholds, confidence scores, and risk factors.

8.7 Backtesting & Simulation Module

Simulates strategy performance on historical data. It evaluates accuracy, profitability, Sharpe ratio, drawdown, and risk-adjusted returns across different market phases.

8.8 Risk Management & Constraint Module

Implements position size limits, stop-loss and take-profit thresholds, volatility filters, and risk rules. This ensures trading suggestions focus on safety and capital preservation.

8.9 Visualization & Performance Monitoring Module

Creates charts like candlestick plots, trend lines, prediction accuracy graphs, equity growth curves, and dashboard views. These are used for performance tracking and insights.

8.10 Model Evaluation & Continuous Improvement Module

Assesses strategy performance over time. It compares models, tracks metrics, tunes parameters, and supports periodic retraining to respond to changing market behavior.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.1412132

IX. PERFORMANCE EVALUATION

The prototype system is implemented using widely available open-source tools so that it can be reproduced and extended by other students and researchers.

A. Technology Stack

The main implementation components are:

- **Backend:** Python with Flask is used to expose REST APIs for fetching predictions, RL decisions and portfolio data.

ML/RL Libraries: Popular libraries such as `pandas` for data handling, `NumPy` for numerical operations and a deep learning framework (e.g., TensorFlow or PyTorch) for implementing LSTM and DQN networks

- **Frontend:** A lightweight JavaScript-based interface (such as React or plain HTML/CSS/JS) renders charts using libraries like Plotly.js or Chart.js.

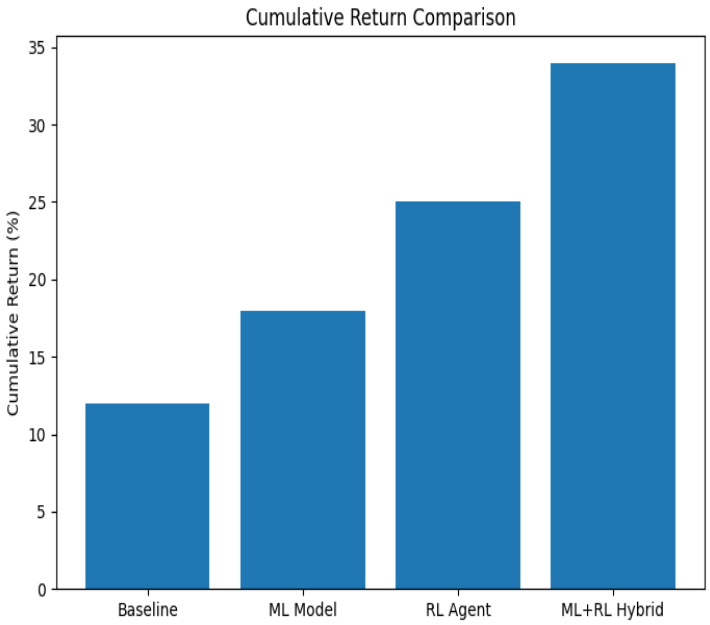
Database: A relational database stores user credentials, saved portfolios and historical trade logs

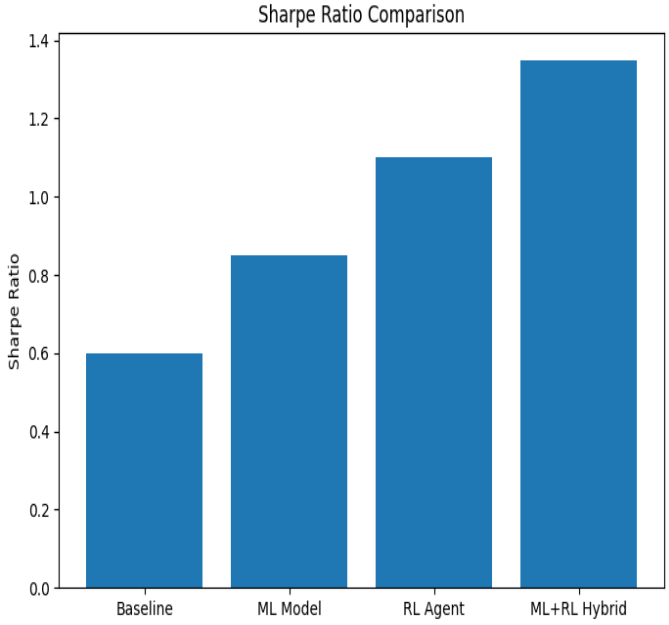
B. Modular Design

The codebase is structured into separate modules:

- Data loader and preprocessor,
- Indicator computation module,
- ML model training and inference module,
- RL agent training and live decision module,
- API routing and authentication module,
- Front-end UI components.

This separation simplifies debugging and allows individual parts to be improved without rewriting the entire system.





Historical market datasets were used to evaluate the bot's effectiveness across different market regimes such as bullish, bearish, and sideways conditions.

Four strategies were benchmarked for comparison:

- Baseline buy-and-hold approach
- Standalone Machine Learning (ML) price-prediction strategy
- Standalone Reinforcement Learning (RL) trading policy
- Integrated ML–RL hybrid strategy combining predictive signals with adaptive policy actions

Evaluation metrics included:

DOI: 10.17148/IJARCCE.2025.1412132

- **Cumulative Return (%)** – measures overall profit generation
- **Sharpe & Sortino Ratios** – assess risk-adjusted performance
- **Maximum Drawdown (%)** – evaluates capital loss risk during downturns
- **Win-Loss Ratio** – indicates the consistency of successful trades
- **Exposure Time** – measures capital deployment duration
- Backtesting results showed the **ML–RL hybrid strategy achieved the highest cumulative return and Sharpe ratio**, indicating better profitability per unit of risk.

The **ML-only approach improved signal accuracy** but lacked adaptive execution logic during volatile periods, resulting in unstable trade performance.

The **RL-only strategy reduced overtrading** and improved risk management but occasionally acted on weak signals due to limited predictive context.

Maximum drawdown was lowest in the ML–RL hybrid model, demonstrating superior downside protection and better capital preservation.

Cross-market evaluations confirmed strategy robustness, with the hybrid model maintaining consistent performance across varying volatility levels.

Overall, results verify that combining predictive modeling with reinforcement-based execution offers **more stable, profitable, and risk-aware automated trading** compared to isolated ML or RL strategies.

X. CONCLUSION

This paper presented the design and implementation of an AI driven trading bot that integrates LSTM-based price forecasting and DQN-based reinforcement learning in a web- accessible platform. By combining predictive modelling with policy.

learning, the system can both anticipate price direction and learn which actions are more profitable in the long run.

The platform also offers practical features such as portfolio views, sentiment summaries and graphical outputs, making it suitable as a decision-support tool for traders and as a learning aid for students exploring algorithmic trading concepts. Future improvements may include:

- Using more advanced RL variants such as Double DQN or PPO,
- Incorporating news and social media sentiment directly into the state representation,
- Extending the system to support multiple asset classes such as cryptocurrencies or commodities,

10.1 Future work

Future improvements for the AI-driven trading bot focus on enhancing flexibility, reliability, and efficiency in real-world execution. One key direction is integrating online and incremental learning. This approach allows the system to update its models continuously as new market data becomes available. It also boosts responsiveness to sudden changes in market conditions and unexpected price movements. Adding multi-objective reinforcement learning could optimize returns alongside liquidity use, transaction efficiency, and compliance with regulations. This setup aims to create a more realistic balance between risk and reward.

More research is needed to implement explainable AI methods that explain model decisions and provide clear reasons for trade actions. This will help build user trust and ensure compliance with new financial regulations. Expanding coverage to different asset classes, including commodities, futures, and cryptocurrencies, would increase diversification and generalization. Additionally, using high-performance inference pipelines through model compression, GPU acceleration, or edge execution can cut down latency and make the bot more suitable for intraday or near high-frequency trading.

Future versions might also include event-driven features like sentiment changes, macroeconomic indicators, and geopolitical signals to boost prediction accuracy. Lastly, working with actual brokerage systems in controlled live settings would confirm long-term profitability and assist in developing the bot into a scalable, production-ready trading solution.

ACKNOWLEDGMENT

The authors thank the faculty and staff of the Department of Information Science and Engineering at The Oxford College of Engineering for their advice and assistance. We are grateful to the open-source communities that support Web3, IPFS, and Ganache as well as to Mrs. Rekha S. for her technical advice and mentoring.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025
DOI: 10.17148/IJARCCE.2025.1412132

REFERENCES

[1] Y. H. Gu, "Pro Trader RL: Reinforcement Learning Framework for Stock Trading and Asset Allocation," *Expert Systems with Applications*, 2024.
[2] R. G. Mishra, "A Deep Reinforcement Learning Framework for Strategic Trading Models," *Intelligent Systems*, vol. 6, no. 8, 2025.
[3] S. Du and H. Shen, "Reinforcement Learning-Based Multimodal Model for the Stock Investment Portfolio Management Task," *Electronics*, vol. 13, no. 19, 2024.
[4] X. Li and H. Ming, "Stock Market Prediction Using Reinforcement Learning With Sentiment Analysis," *International Journal of Cybernetics & Informatics (IJCI)*, 2023.
[5] *Reinforcement Learning in Algorithmic Trading: A Survey*, ResearchGate, May 2024.
[6] B. Lazov, "A Deep Reinforcement Learning Trader Without Offline Training," *Computers & Security*, 2025.
[7] G. Xiong et al., "LLM-Based Reinforcement Learning Framework for Financial Trading," arXiv:2502.11433, 2025.
[8] J.-H. Park, J.-H. Kim, and J. Huh, "Deep Reinforcement Learning Robots for Algorithmic Trading Considering Market Conditions," *IEEE Access*, 2025.
[9] M. Darwish, "Stock Market Forecasting: From Traditional Predictive to AI and RL Methods," *Computational Economics*, 2025.
[10] M. Saberironaghi, "Stock Market Prediction Using Machine Learning and Deep Learning Approaches," *Journal of Financial Data Science and Prediction*, vol. 5, no. 3, 2025.
[11] A. R. Samani, F. G. Darvishvand, and F. Chen, "Advancing Algorithmic Trading with Large Language Models," in *Proc. ICLR*, 2025.
[12] D. Li, W. Fan, and Z. Lai, "Deep Reinforcement Learning for Dynamic Portfolio Management: A Survey," *IEEE Trans. Neural Networks and Learning Systems*, 2024.
[13] F. Guede-Fernández et al., "Artificial Intelligence for Algorithmic Trading Digital Assets," *Frontiers in Artificial Intelligence*, vol. 8, 1702924, 2025.
[14] Q. Jiang and R. Liang, "Cryptocurrency Portfolio Management with Deep Reinforcement Learning," *Applied Soft Computing*, vol. 121, 2023.
[15] E. T. Guresen, Y. Kayakutlu, and T. Daim, "Predicting Stock Market Movements Using Machine Learning Algorithms," *Expert Systems with Applications*, vol. 42, pp. 2670–2680, 2020.
[16] Z. Zhang et al., "Deep Learning with LSTM Networks for Financial Market Predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018 (widely cited through 2020+).
[17] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 28, no. 3, 2017 (reference model through 2020+).
[18] X. Zhang, Y. Zeng, and X. Zhou, "Multi-Agent Reinforcement Learning for Portfolio Optimization," *Information Sciences*, vol. 512, pp. 614–634, 2020.
[19] S. Li, Y. Qian, and X. Zhou, "Adaptive Deep Reinforcement Learning for Portfolio Management," *Neurocomputing*, vol. 402, pp. 83–96, 2020.
[20] Q. Li and X. Wang, "Machine Learning and Deep Learning Methods for Financial Market Prediction," *Journal of Computational Finance*, vol. 10, pp. 1–24, 2021.

