Adaptive Decision-Making for Equity Trading Using Deep Reinforcement Learning

Ashok Jana
Data Science
Anderson College of Business and Computing
Regis University, Denver, CO, USA
ajana@regis.edu

Abstract

Financial markets don't sit still. Volatility spikes, trends reverse, and what worked last month stops working today. This project builds an adaptive trading agent using deep reinforcement learning to handle these shifts. Instead of fixed rules or static predictions, the agent learns trading policies through simulated market interaction. I'll use deep neural networks to approximate action-value functions for sequential decisions. Raw historical data gets processed into state representations capturing price movements, volume, volatility signals, and macro indicators. Several baseline strategies get compared against different RL configurations to test what actually improves risk-adjusted returns. The core question: can agents learn to recognize and adapt to different market regimes? This project tests whether regime-aware representations help models generalize across bull markets, bear markets, and choppy sideways action.

I. Introduction/Background

Equity trading involves making decisions under constantly changing uncertainty. A strategy optimized for low-volatility periods often crashes when volatility jumps. Fixed models trained on history struggle when dynamics shift.

Traditional approaches use rule-based strategies (moving averages, momentum) or supervised models that predict direction. Both break down. Rules can't adapt when market structure changes. Supervised models assume the future looks like the past.

Reinforcement learning works differently. Instead of predicting what will happen, an RL agent learns what to do. It interacts with an environment, gets rewards based on performance, and adjusts its policy to maximize long-term returns.

This project tackles the full data science pipeline for time-series decision-making. I'll handle data from multiple sources, create state representations, implement RL algorithms, design realistic simulations with transaction costs, and evaluate across market regimes. The scope includes model selection, hyperparameter tuning, risk management, and interpreting learned policies.

The project combines advanced machine learning techniques with domain-specific financial modeling, requires handling streaming data and sequential decision processes, and rigorous experimental evaluation to separate genuine learning from overfitting to historical patterns.

II. Problem Statement

How can an automated trading agent learn to adapt its strategy across different market conditions without human intervention?

Most trading algorithms fail because they're built on assumptions that are not consistent. A momentum strategy works great during trending markets but loses money when prices oscillate. Mean reversion strategies profit from range-bound markets but get crushed during breakouts. The problem is not finding one good strategy, it's having a system that recognizes when to apply different approaches.

This project frames trading as a sequential decision-making problem where an agent must learn optimal actions (buy, sell, hold) based on current market state. The agent needs to balance exploration (trying new strategies) with exploitation (using proven approaches), manage risk by considering position sizing and drawdown constraints, and learn from delayed rewards since trading decisions show their value over multiple time steps.

Table I shows how different market conditions require different approaches. A single fixed strategy can't handle this variation effectively.

TABLE I
MARKET REGIME CHARACTERISTICS AND STRATEGY PERFORMANCE

Market Regime	Trend Strategy	Mean Reversion
Low Volatility	Moderate	Strong
Normal Market	Strong	Moderate
High Volatility	Weak	Weak
Crisis Period	Failed	Failed

The project covers the full data lifecycle: collection (pulling from Yahoo Finance), preparation (handling missing values, adjusting for splits, normalizing features, creating rolling indicators), exploration (analyzing correlations, identifying regime changes, examining reward distributions), modeling (implementing DQN or policy gradient algorithms), visualization (learning curves, cumulative returns, decision patterns), and reporting (documenting architectures, hyperparameter sensitivity, out-of-sample performance).

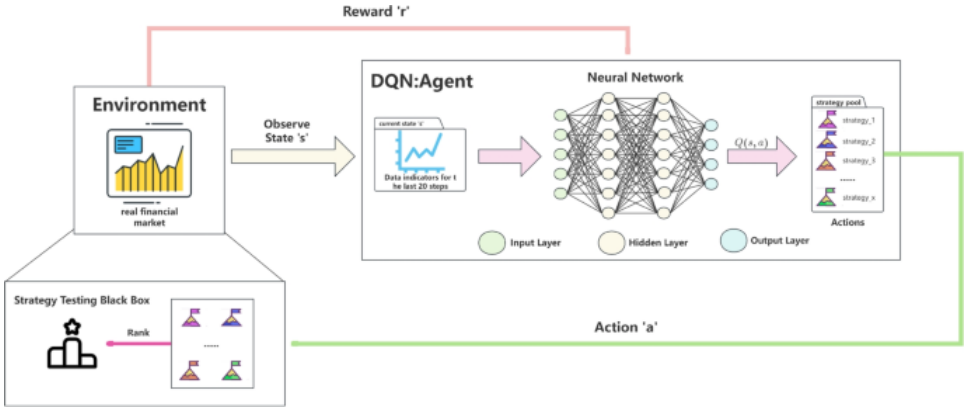

Fig. 1. Reinforcement learning framework for adaptive trading. The agent interacts with the market environment through a cycle of observation, action, and reward feedback. Source: Zhong *et al.* [1].

The potential impact: if the agent achieves a Sharpe ratio improvement of 0.3 over buy-and-hold on new data while maintaining lower maximum drawdown, It represents a great risk-adjusted outperformance.

III. EXPECTED OUTCOMES

This project is expected to produce a reproducible reinforcement learning framework for adaptive equity trading, along with a clear empirical comparison of multiple trading agents under changing market conditions.

The outcomes include several trained agents: value-based DQN variants (DQN, Double DQN, and Dueling DQN), sequence-aware agents using BiLSTM architectures, attention- and transformer-based agents, and hybrid models that combine convolutional and temporal components. These will be evaluated against rule-based strategies and a supervised learning baseline using consistent risk-adjusted metrics.

A custom trading environment that models transaction costs, position limits, and portfolio dynamics will support regime-specific evaluation across different volatility and trend conditions. Rather than aiming

for constant outperformance, the expected result is a realistic assessment of when adaptive agents improve stability and risk-adjusted returns, and when they fail to generalize.

The project is limited to historical data and simulated execution, and does not account for real-world trading frictions such as market impact or latency. Success is defined by delivering a transparent, well-documented analysis that clarifies the practical strengths and limits of reinforcement learning for adaptive trading.

IV. Related Work

Deep reinforcement learning has become a common tool for trading problems, largely because trading fits naturally into a sequential decision-making setup. The early success of deep Q-networks in control tasks [2] encouraged researchers to explore similar ideas in financial markets, even though markets are noisier and far less forgiving.

Several studies apply DRL directly to stock trading. Li et al. [3] and Bao [4] report that DRL agents can outperform traditional strategies when trained on price-based indicators, but their experiments mostly assume relatively stable market conditions. Actor–critic models with memory, such as the LSTM-DDPG approach by Jia et al. [5], handle variable positions more naturally, though they stop short of explicitly dealing with regime shifts.

On the tooling side, FinRL [6] has made it easier to compare DRL methods under a shared framework. That said, many benchmark results still gloss over how strategies behave during high-volatility or crisis periods. More recent work has tried to address instability and risk. Imitative learning [7], risk-aware objectives [8], and the use of macroeconomic signals [9] all point in this direction, with similar ideas appearing in cryptocurrency markets [10].

What stands out to me is that regime awareness and realistic trading frictions are still treated inconsistently. This work focuses on that gap by examining how DRL agents behave across different market regimes while accounting for costs and risk in a controlled setting.

V. Methodology

The research design adopted in this project is the type of experiment research to determine whether or not the trading strategies based on reinforcement learning are more effective than the old school method of investment decisions. We are training ML agents with actual market information to make decisions to buy, hold or sell. To gather the following data we are using Python yfinance package to retrieve daily stock market data via Yahoo Finance API. It discusses investments such as SPY, AAPL, MSFT, JPM, XOM and JNJ between 2015 and 2024. It has everything the standard stuff possesses, and it includes open, high, low, close, adjusted close and volume, which is stored in Parquet format so it also loads quickly. In order to prevent data leakage, we divide it chronologically: the training 2015-2019, validation in 2020, and 2021-2024 is the test set. We then add a handful of technical pointers on that raw data: RSI, MACD, moving averages, Bollinger Band width, ATR, OBV and returns. These assist in drawing in momentum, trends, volatility and the trading activity. We also handle possible data issues such as missing data, outliers, winsorizing extremes and StandardScaler normalization features to ensure that the models receive quality and clean input.

In this project we are largely working in Python, based on the data science stack in general. Pandas and NumPy are used to crunch the data whereas Matplotlib and Plotly are used to visualize them. On the reinforcement learning end, I have been able to create my own trading environment in Gymnasium which will begin with an initial capital of 10,000 and introduce some real-world transaction costs to help the simulation feel like a real-world investment. In training, I am training a variety of Deep Q-Network (DQN) models using PyTorch: a basic one, a version with dropout, a residual architecture, a dueling network, an LSTM-based DQN and an attention-based DQN. These models will produce Q-values on which the decision to trade will be made depending on the appearance of the market and that which is

held in the portfolio. I tweak the hyperparameters such as learning rate and exploration decay to improve performance of the models and combinations of models.

To ensure that everything is solid, This project operates a multitude of tricks in verification and evaluation. To ensure the results remain free of bias, I tested the models on validation and test sets that were not used during training. The three measures that are monitored are the eventual value of the portfolio, performance relative to a simple buy and hold and the dispersion of findings across a number of runs. I do as well add baseline tactics such as random trading and the traditional technical rules only to be able to tell our position. To maintain quality control I seed randomly, repairing the seed set to ensure that it is reproducible, maintaining the dataset splits at good quality and tracking all the models configs I tested and all the log of experiment data in a systematized manner. Finally I developed a interactive Streamlit dashboard which presents the stock trends, return histograms and model forecasts.

VI. Data Description

In this project I scraped historical financial market data publicly of a sample of big tickers and an index ETF, which is SPY, AAPL, MSFT, JPM, XOM, JNJ, 2015 - 2024. I scraped the data with Yahoo Finance via yfinance in Python and tracked the standard financial variables: open, high, low, close, adjusted close and volume. Then I gave those raw figures engineered features to ensure the analysis got interesting technical indicators such as RSI, MACD, moving averages, Bollinger Bands, ATR, OBV and various indicators based on returns. These assist us to capture both market momentum, market trend, volatility and trading activity all of which is what we require in the predictive models. These were all stored in Parquet format and divided into folders with separate training, validation and test splits to prevent data leakage.

The project does not entail any personal or sensitive information or otherwise, there is no sight of it even in terms of governance or ethics and thus, GDPR or HIPAA do not apply anymore. It follows a good data science playbook: performing good data validation, missing data, by aligning and cleaning up data, and dealing with extreme outliers with winsorization and normalization. I understand the biases of financial time-series, such as market-specific effects or overfitting to the past, and I will exploit many assets, across many industries, divide the data with time and rely on the ensemble forecasts to ensure it remains strong.

VII. Conclusion

The issue that this practicum project addresses is the difficulty in constructing information-based trading policies using reinforcement learning to help me gain a clearer understanding of how turbulent financial markets operate. Trading traditional methods tend to be adhered to the fixed rules or the extensive use of manual analysis that can become clogged in cases where the market alters rapidly. To address this, I will compute a structured data science pipeline: I will scrape and clean the history of financial data then construct features using technical indicators and lastly, construct a trading environment with reinforcement learning. I will also make multiple Deep Q -networks (DQN) and optimize them with hyperparameter experiments and ensemble tricks to allow the agent to decide when to purchase, retain, or dispose of the raw market data. I will use extensive validation and head-to-head testing against benchmarks such as buy-and-hold to determine whether such smart agents can really out-perform simple strategies.

The project contributes to the data science community as well as financial analytics since it demonstrates that contemporary machine learning (particularly reinforcement learning) can be applied in real world, money making decisions. As a learning outcome, the practicum summarizes the skills we have acquired during the data science major, which include cleaning data, training the models, putting the reinforcement learning into practice, evaluating the models, and visualizing the outcomes. It helps me know that I can build an entire end-to-end data science solution and prepares me to work in data science, quantitative analysis, and machine learning engineering.

VIII. Timeline

Weeks 1–2: Collect equity data along with market indicators such as volatility indices and interest rate series. Check for timestamp mismatches, and inconsistencies across sources, then create time-based train, validation, and test splits. Implement the trading environment as an MDP (for transaction costs, position limits, and portfolio state updates). Run simple random and rule-based agents to confirm that state transitions and rewards.

Week 3: Develop basic benchmark strategies, including buy-and-hold, moving average crossover, and momentum-based rules. Train a supervised learning model to act as a non-reinforcement learning reference point. Use these baselines to establish initial performance measures.

Week 4: Review baseline results to identify weakness. Adjust feature construction, normalization, and evaluation procedures to ensure they support both simple feedforward agents and models that rely on sequential information.

Week 5: Train a standard value-based reinforcement learning agent to serve as the primary RL baseline. Extend this to agents with recurrent networks that can capture temporal dependencies in price movements, and compare their training and performance against the baseline agent.

Week 6: Experiment with more expressive agent architectures, including attention-based and transformer-inspired models, as well as hybrid designs that combine convolutional feature extraction with temporal modeling. Perform hyperparameter tuning and compare learning behavior, convergence speed, and sensitivity to market conditions.

Week 7: Evaluate all trained agents on held-out test periods that were not used during development. Conduct ablation studies to understand the impact of selected features and architectures. Analyze agent behavior across different market situations, generate performance plots and summaries, document for reproducibility, and prepare the final report and presentation.

References

[1] X. Zhong, J. Wei, S. Li, and Q. Xu, "Deep reinforcement learning for dynamic strategy interchange in financial markets," *Applied Intelligence*, vol. 55, no. 30, 2025, published online: 26 November 2024.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] Y. Li, P. Liu, and Z. Wang, "Stock trading strategies based on deep reinforcement learning," *Scientific Programming*, vol. 2022, pp. 1–15, 2022.

[4] M. Bao, "Deep reinforcement learning based optimization and risk control of trading strategies," *Journal of Engineering Science*, vol. 20, no. 5s, pp. 241–252, 2024.

[5] Z. Jia, Q. Gao, and X. Peng, "Lstm-ddpg for trading with variable positions," *Sensors*, vol. 21, no. 19, p. 6571, 2021.

[6] X.-Y. Liu, H. Yang, Q. Chen, R. Zhang, L. Yang, B. Xiao, and C. Wang, "Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance," *arXiv preprint arXiv:2011.09607*, 2020.

[7] Y. Liu, Q. Liu, H. Zhao, P. Zhen, and C. Liu, "Adaptive quantitative trading: An imitative deep reinforcement learning approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, 2020, pp. 2128–2135.

[8] E. Lwele, "Risk-aware deep reinforcement learning for dynamic portfolio optimization," *arXiv preprint arXiv:2511.11481*, 2025.

[9] J. Park, J. Kim, and J. Huh, "Deep reinforcement learning robots for algorithmic trading: Considering stock market conditions and u.s. interest rates," *IEEE Access*, vol. 12, pp. 20 705–20 725, 2024.

[10] K. Paykan, "Cryptocurrency portfolio management with reinforcement learning: Soft actor–critic and deep deterministic policy gradient algorithms," *arXiv preprint arXiv:2511.20678*, 2025.