National College of Ireland

A Comparative Study of Reinforcement Learning Algorithms and Custom Technical Indicators for Short-Term Stock Trading

MSc Research Project
Data Analytics

Roopak Mallik
Student ID: 23433639

School of Computing
National College of Ireland

Supervisor: Dr. David Hamill

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Roopak Mallik
Student ID:	23433639
Programme:	MSc Data Analytics **Year:** 2025-2026
Module:	Research Practicum/ Master Thesis
Supervisor:	Dr. David Hamill
Submission Due Date:	11th December 2025
Project Title:	A Comparative Study of Reinforcement Learning Algorithms and Custom Technical Indicators for Short-Term Stock Trading
Word Count:	11130 **Page Count** 39

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Roopak Mallik

Date: 8th December 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

National College of Ireland

Project Submission Sheet

Student Name:	Roopak Mallik		
Student ID:	23433639		
Programme:	MSc Data Analytics	**Year:**	2025-2026
Module:	Research Practicum/ Master Thesis		
Lecturer:	Dr. David Hamill		
Submission Due Date:	11th December 2025		
Project Title:	A Comparative Study of Reinforcement Learning Algorithms and Custom Technical Indicators for Short-Term Stock Trading		
Word Count:	11130		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	Roopak Mallik
Date:	8th December 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**
6. **Please check that you read AI and Academic Integrity Acknowledgement Supplments in this document**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

AI Acknowledgement Supplement

Research Practicum/ Master Thesis

A Comparative Study of Reinforcement Learning Algorithms and Custom Technical Indicators for Short-Term Stock Trading

Roopak Mallik/23433639	MSc Data Analytics	8th December 2025
NA	NA	NA

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click here.

AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool
NA	NA	NA

Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

[Insert Tool Name]	
[Insert Description of use]	
[Insert Sample prompt]	[Insert Sample response]

Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

Additional Evidence:

[Place evidence here]

Additional Evidence:

[Place evidence here]

A Comparative Study of Reinforcement Learning Algorithms and Custom Technical Indicators for Short-Term Stock Trading

Roopak Mallik
x23433639
MSc Data Analytics
National College of Ireland
Supervisor: Dr. David Hamill

Abstract—Reinforcement learning (RL) has emerged as a promising alternative to traditional algorithmic trading approaches, yet significant gaps remain in understanding how different RL algorithms compare and whether adding technical indicators improves trading performance. This dissertation systematically compares three prominent RL algorithms—Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C) for short-term stock trading using historical daily data from three GPU technology companies (NVIDIA, AMD, and Intel) over eight months from January to August 2025.

The research addresses two key objectives: first, comparing DQN, PPO and A2C performance using standard market data (Open, High, Low, Close, Volume); second, determining whether adding technical indicators (RSI, SMA, OBV) improves trading outcomes. Using OpenAI Gymnasium, agents were trained on 20,000 timesteps of historical data and evaluated on separate test data under identical conditions to ensure fair comparison. Performance was measured using factors such as total profit.

Results show that A2C outperforms the other algorithms, occasionally achieving profits above break-even (1.0385 on AMD with standard data), though with significant variability between runs. DQN generally remains near break-even but experiences substantial losses on Intel, while PPO consistently underperforms, never reaching profitability. Adding technical indicators produces relatively moderate improvements across all algorithms, with A2C benefiting most (reaching 1.0652 on AMD and 1.0749 on Intel). However, no algorithm achieves consistent profitability, highlighting fundamental challenges in applying RL to volatile financial markets with noisy data.

This research advances the field by establishing a standardized experimental framework that enables fair algorithm comparison and demonstrates that while algorithmic choices and feature engineering influence performance, significant challenges remain in developing stable and consistently profitable automated trading systems. The findings provide realistic expectations for both researchers and practitioners, emphasizing the need for rigorous evaluation, careful feature engineering and acknowledgment of limitations when deploying RL-based trading strategies in real-world financial markets.

1. Introduction

1.1 Background and Motivation

The financial markets represent one of the most complex and dynamic environments in which decision-making algorithms must operate. Traditional approaches to algorithmic trading have largely relied on rule-based systems, statistical models or supervised machine learning techniques that attempt to predict future price movements. However, these methods often struggle to adapt to the rapidly changing market conditions and the non-linear, non-stationary nature of financial time series data.

Reinforcement Learning (RL) offers a fundamentally different paradigm for approaching financial decision-making. Rather than attempting to predict future prices, RL agents learn optimal trading strategies through direct interaction with the market environment, receiving feedback in the form of rewards based on their trading performance. This approach aligns naturally with the sequential decision-making nature of trading, where each action influences future market positions and potential returns.

The emergence of deep reinforcement learning has opened new possibilities for sophisticated trading strategies that can potentially capture complex market patterns and adapt to changing conditions. RL algorithms such as Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C) have demonstrated success across various domains, yet their comparative performance in financial markets, particularly for short-term trading strategies, remain unexplored under controlled experimental conditions.

1.2 Problem Statement

Despite growing interest in applying reinforcement learning to financial markets, several critical gaps exist in the current literature. First, most studies focus on individual RL algorithms without providing systematic comparisons under identical market conditions and evaluation frameworks. This makes it difficult to determine which approaches are most suitable for specific trading scenarios. Second, the role of feature engineering, particularly the inclusion of traditional technical indicators, in RL-based trading systems remains poorly understood. While technical indicators such as Relative Strength Index (RSI), Simple Moving Average (SMA), and On-Balance Volume (OBV) are widely used in traditional technical analysis, their contribution to the performance of RL agents has not been thoroughly investigated or explored.

The lack of standardized evaluation frameworks and inconsistent experimental setups across studies further complicates the assessment of RL algorithms' effectiveness in trading applications. Many existing studies use different datasets, evaluation periods and performance metrics, making it challenging to draw meaningful conclusions about the relative merits of different approaches.

1.3 Research Objectives

This dissertation aims to address these gaps by conducting a comprehensive comparative study of reinforcement learning algorithms for short-term stock trading. The primary objectives are:

Algorithm Comparison: To systematically evaluate and compare the performance of three prominent reinforcement learning algorithms—Deep Q-Network (DQN), Proximal Policy

Optimization (PPO), and Advantage Actor-Critic (A2C) in the context of short-term stock trading using daily OHLCV data.

Feature Engineering Analysis: To investigate the impact of incorporating traditional technical indicators (RSI, SMA, OBV) on the performance of RL trading agents compared to models trained exclusively on raw OHLCV dataset.

Assessment: To evaluate the consistency of findings across different stocks in the GPU technology sector, specifically focusing on NVIDIA (NVDA), AMD, and Intel (INTC) to understand the proposed approaches.

1.4 Research Questions

This study seeks to answer two fundamental research questions:

Q1: Which reinforcement learning algorithm (DQN, PPO, or A2C) demonstrates superior performance for short-term stock trading when trained on daily OHLCV data?

Q2: How does the inclusion of custom technical indicators (RSI, SMA, OBV) affect the performance of reinforcement learning trading agents compared to using raw OHLCV data exclusively?

1.5 Contribution

This dissertation contributes to the field of algorithmic trading by providing a systematic comparison of three reinforcement learning algorithms—Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C) under identical experimental conditions, offering a fair assessment of their effectiveness in short-term stock trading. It further explores the impact of incorporating traditional technical indicators such as RSI, SMA and OBV into reinforcement learning frameworks highlighting the role of feature engineering in enhancing trading performance. By focusing on stocks from the GPU technology sector (NVIDIA, AMD, Intel), the study ensures sector-specific relevance while maintaining generalizability across assets. Additionally, the research establishes a standardized experimental framework using consistent datasets, timeframes and evaluation metrics, thereby improving reproducibility and comparability for future studies. Collectively, these contributions provide both theoretical insights into the application of reinforcement learning in financial markets and practical guidance for designing more adaptive and effective trading strategies.

1.6 Potential Limitations

Reinforcement Learning models are very sensitive to choices about hyperparameters, how the reward function is designed and training stability. This means that small changes can greatly change how well the model performs. Using technical indicators like RSI, SMA and OBV gives useful information for feature engineering, but the range of indicators looked at is limited. More advanced or specific features might lead to different results. The study assumes perfect market conditions by not including factors like transaction costs, liquidity issues and slippage which could impact profits in the real world.

1.7 About the Dataset

This study utilizes daily stock market data spanning the period from January 2025 to August 2025, acquired through the Yahoo Finance (yfinance) API. The selection of this eight-month timeframe serves two primary methodological purposes. First, it aligns with the short-term trading context of this research. Second, the use of contemporary data is critical for capturing current market dynamics, volatility patterns and behavioural characteristics that may differ substantially from historical periods due to evolving market structures, regulatory changes and macroeconomic conditions.

The yfinance API was selected as the data acquisition tool due to several practical and methodological advantages. As an open-source library, yfinance provides unrestricted access for academic and personal research purposes without requiring licenses or institutional subscriptions. Furthermore, the API can be tuned to get customised data as per the project requirements in the future.

2. Literature Review

2.1 Markov Decision Process

Markov Decision Process is a mathematical concept for modelling decision-making with uncertainty (Bellman, 1957). It is about modelling a scenario in which outcomes are partly under the control of a decision-maker. The agent interacts with the environment, takes actions toward goals, receives rewards and transitions to different states based on these actions.

2.2 Reinforcement Learning in Financial Markets and Stock Trading

The applications of Reinforcement Learning in financial markets and stock trading has emerged as a prominent research area offering novel approaches to address the complexities of modern financial systems. This literature review combines recent advances, methodologies and challenges in applying RL techniques to quantitative trading, drawing from comprehensive surveys and framework developments in the field.

This work on the application of RL to trading and execution problems set the stage for later research in Deep Reinforcement Learning. The original formulation of DRL can be found in the work by Moody and Saffell who coined the name and considered an agent that optimizes financial performance objectives. Another early field of RL in finance was trade execution. Here the question of optimal execution is formulated as a sequential decision problem taking price impact and uncertainty into account. Early literature already tackled state encoding, financial reward design (return, drawdown, risk), transaction costs and liquidity, as well as dealing with non-stationarity of the financial time series.

With the advent of deep RL particularly thanks to breakthroughs like the Deep Q-Network (DQN) in games, financial researchers began to adopt architectures that combine representation learning (e.g., convolutional or recurrent neural networks) with RL objectives. Deng et al. (2017) proposed a recurrent deep neural network model that integrates deep learning for feature extraction with RL for decision-making in trading. They demonstrated robustness on both stock

and commodity futures markets. Jiang et al. (2017) explored how DRL agents can optimize stock trading strategies, and Zhang et al. (2019) systematically compared modern DRL methods in multi-asset trading setups.

These studies demonstrate that RL can learn trading policies end-to-end from raw price/time-series input but also highlight practical issues such as the explosion of state dimensions, continuous action spaces, and the need for robust backtesting.

2.3 RL Frameworks and Financial Modelling

Financial trading is often modelled as a Markov Decision Process (MDP), a framework that provides a theoretical basis for achieving goals through interactive learning. In an MDP, a trading agent observes the market's current state, takes an action (buy, sell, or hold), and receives a reward based on the outcome with the goal of maximizing long-term returns.

The core components of the MDP in a financial context are typically defined as

State: Represents the information an agent observes at a given time. This can include the agent's own status such as account balance and current stock holdings as well as market data like historical prices, trading volume and various other technical indicators.

Action: Describes the set of possible moves the agent can make. In trading, this usually involves buying, selling, or holding a certain number of shares of one or more stocks. Actions can be modelled in a discrete space (e.g., buy, sell) or a continuous space (e.g., what percentage of a portfolio to allocate to an asset).

Reward: An incentive that guides the agent's learning process. The reward function is often defined as the change in portfolio value, the portfolio's log return or a risk-adjusted metric like the Sharpe ratio.

A variety of RL algorithms are employed to solve this MDP, broadly categorized as model-free and model-based. Model-free algorithms learn directly from interactions without building an explicit model of the market's dynamics, while model-based approaches learn a simulator of the environment to plan actions.

The Deep Q-Network (DQN) algorithm, originally developed for Atari games, has been adapted in finance for trading tasks where the action space is discrete. Théate and Ernst (2020) proposed “Trading DQN” (TDQN) derived from DQN and applied it to algorithmic trading, generating promising results. A further advantage of DQN is its relative simplicity in implementation and sample efficiency when actions are discrete. However, DQN does not natively handle continuous portfolio weights, suffers from value-estimation bias, and its replay-buffer mechanism may violate temporal dependencies in non-stationary financial data.

Actor-critic methods such as A2C (Advantage Actor-Critic) and A3C (Asynchronous Advantage Actor-Critic) combine a policy (actor) that selects actions and a value-function (critic) that estimates expected return, thereby reducing variance of the policy-gradient updates. In financial trading, these methods are increasingly used for continuous action spaces (e.g.,

adjusting portfolio weights, dynamic risk control) and for multi-asset allocations. Empirical studies such as Yang et al. (2020) employ actor-critic architectures inside ensembles of trading agents, showing that actor-critic methods can handle position sizing and risk-control better than discrete value-based approaches.

Proximal Policy Optimization (PPO) is a policy-gradient method developed to improve stability of training by restricting how much the policy can change at each update. In finance, PPO has gained popularity because trading environments are noisy and unstable. Papers such as Briola et al. (2021) apply PPO in high-frequency trading environments, showing that the algorithm learns stable trading strategies from order-book data.

Empirical studies have shown that RL-based agents outperform baseline strategies, accounting for transaction costs and non-stationarity. In two studies, Deng et al. (2017) and Liu et al. (2018), the DRL-based agents realized a higher Sharpe ratio and a higher cumulative return. Zhang et al. (2019) and Yang et al. (2020) show that overfitting can be avoided using appropriate reward design, transaction cost modelling, and walk-forward validation strategies.

2.4 Recent Progress in Reinforcement Learning in Financial Markets

Reinforcement learning (RL) has emerged as a transformative approach in quantitative finance, offering model-free alternatives to traditional stochastic control methods. Recent years have witnessed substantial advances across multiple financial applications driven by the integration of deep learning architectures and the availability of high-frequency market data.

2.5 Algorithmic Trading and Portfolio Optimization

Deep reinforcement learning (DRL) has demonstrated remarkable capabilities in developing automated trading systems. Hambly et al. (2023) provide a comprehensive survey showing that approximately 39% of recent research has focused on RL-based approaches, with particular emphasis on policy gradient methods and Q-learning variants. The Actor-Critic framework has proven especially effective, with studies demonstrating significant improvements over traditional benchmarks like Time-Weighted Average Price (TWAP) strategies (Hambly et al., 2023). Recent implementations have achieved Sharpe ratios exceeding 127% in quantitative trading strategies using Gated Recurrent Unit (GRU) networks optimized through RL algorithms (Sahu et al., 2023).

Portfolio optimization has benefited from RL's ability to handle sequential decision-making under uncertainty. Wang and Zhou's (2020) continuous-time mean-variance framework with entropy regularization demonstrated that RL agents can learn Gaussian policies with decaying variance, outperforming classical maximum likelihood approaches (Hambly et al., 2023).

Multi-asset portfolio management using Deep Deterministic Policy Gradient (DDPG) has shown superior performance compared to conventional methods like Constantly Rebalanced Portfolio (CRP) strategies (Sahu et al., 2023).

2.6 Methodological Innovations

Some methodological improvements, such as nested RL methods, have included several DRL methods, such as Advantage Actor-Critic, DDPG and SAC, and agent training strategies that allow for dynamic agent selection for a trading strategy specific to a market (Hambly et al., 2023). For model-based RL methods, improved sample complexity through more efficient exploration strategies has been reported to be a solution to limited data sets in finance.

Transfer learning has emerged as a promising direction, allowing knowledge from established markets to accelerate learning in newly listed assets or emerging markets (Sahu et al., 2023). Additionally, adversarial training methods have improved robustness against model uncertainty and changing market regimes (Hambly et al., 2023).

2.7 Performance and Evaluation

Recent studies have established rigorous evaluation frameworks encompassing both predictive accuracy and portfolio performance metrics. Beyond traditional measures like mean squared error, researchers now emphasize risk-adjusted returns through metrics such as Sharpe ratio, maximum drawdown, and Sortino ratio (Sahu et al., 2023).

2.8 Challenges and Future Directions

Robustness: Models trained on historical data may not perform well under different future market conditions, a problem often attributed to low signal-to-noise ratios in financial data.

Explainability: The "black box" nature of deep neural networks makes it difficult to understand why an agent makes a particular trading decision. This lack of interpretability is a major hurdle for adoption by asset managers who have a fiduciary duty to explain risks to clients.

MDP Modelling: Defining an appropriate state, action, and reward structure is non-trivial. An incomplete state representation can violate the Markov property, while a poorly designed reward function can lead the agent to learn suboptimal behaviours.

Strategy Degradation: A profitable strategy discovered by an RL agent may lose its effectiveness over time as other market participants discover and exploit the same patterns.

While many studies demonstrate that RL agents can outperform traditional baselines in simulated environments, the literature lacks substantial evidence of their efficacy in live trading. Future research directions aim to address these limitations through areas like multi-agent RL to model competitive environments, model-based RL to create better market simulators and risk-sensitive RL that explicitly incorporates risk into the learning algorithm itself. The development of standardized open-source libraries and benchmarks, such as FinRL, is also crucial for facilitating reproducible research and fair comparisons across different models.

Author / Year	Dataset used	Models used	Core findings	Limitations / Gaps
Yang et al., 2020 – "Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy"	Daily OHLCV data for Dow Jones 30 constituent stocks (as of 01/01/2016), from 01/01/2009–05/08/2020, downloaded from Compustat via WRDS.	PPO, A2C and DDPG.	Ensemble strategy beats individual agents, DJIA, and minimum variance benchmarks across Sharpe ratio, annual returns, and drawdown from 2016-2020. Adding a turbulence index helps agents reduce risk during the 2020 crash without sacrificing performance.	Results rely entirely on historical backtests of one large-cap US market with no live trading or cross-market testing. Transaction costs and liquidity are simplified, and survivorship bias may exist (DJIA constituents frozen at 2016). The ensemble uses hand-crafted Sharpe-based switching instead of learned strategies, and rewards prioritize risk-adjusted returns while ignoring broader constraints like market impact and regulation.
Sahu et al., 2023 – "An Overview of Machine Learning, Deep Learning, and Reinforcement Learning-Based Techniques in Quantitative Finance: Recent Progress and Challenges"	Survey of >70 papers across many indices and markets, including S&P 500, DJIA, NASDAQ, NYSE, NIFTY50, BSE30, CSI, DAX, FTSE 100, Nikkei 225, Nigerian and African indices. Uses both daily OHLCV and intraday/tick data plus some sentiment/text sources, but no single unified dataset.	Summarises a broad range of ML/DL/RL models: traditional ML (SVM, RF), DL (CNN, RNN/LSTM, GRU), and RL (Q-learning, DQN, actor–critic, DDPG, hybrid models).	Offers a taxonomy covering data types, learning tasks (prediction, classification, trading), and evaluation metrics (RMSE/MAE, accuracy, Sharpe, drawdown, portfolio metrics). Shows that RL and DRL frequently outperform traditional baselines in backtests, with increasing focus on multi-asset portfolio management and risk-aware RL.	As a survey, it lacks a standardized experimental benchmark—datasets, timeframes, costs, and baselines differ significantly across studies. Identifies gaps including inconsistent metrics, limited coverage of certain regions and asset classes, and inadequate treatment of transaction costs, liquidity, and risk constraints in many trading studies.
Hambly, Xu & Yang, 2023 – "Recent Advances in Reinforcement Learning in Finance"	Survey of RL applications using diverse financial datasets: order-book data for optimal execution and market making, historical equity and futures prices for portfolio optimisation, and derivatives data for option pricing/hedging. No single dataset; instead, it synthesises results across many empirical studies.	Covers a spectrum of RL algorithms: value-based (Q-learning, DQN), policy-gradient and actor–critic (PPO, A2C/A3C, DDPG, SAC), and model-based RL.	Presents a unified RL framework for financial decision-making and shows RL can match or surpass classical strategies in optimal execution and portfolio management, particularly in complex, high-dimensional markets. Emphasizes that deep RL reduces dependence on restrictive model assumptions and better leverages rich market data.	Identifies key challenges: sample inefficiency, non-stationary markets, difficulty rigorously incorporating risk constraints and transaction costs, and absence of standard benchmarks and theory for realistic market models. Notes that many reported gains come from context-specific backtests, leaving robustness and out-of-sample generalization as unresolved issues.

Bai et al., 2025 – "A Review of Reinforcement Learning in Financial Applications"	Meta-review of RL across market making, portfolio management, and optimal execution, aggregating results from many published datasets (equities, FX, derivatives, high-frequency order-book data). No single dataset; instead it compiles and normalises performance metrics such as Sharpe ratios from multiple studies.	Analyses a wide range of model-free and model-based RL: actor-only (PG, PPO), critic-only (Q-learning, DQN), actor–critic (A2C/A3C, DDPG, NAC), and emerging model-based, multi-agent and offline RL methods.	Demonstrates that RL often achieves positive Sharpe-ratio gains over strong non-RL baselines in portfolio and execution tasks, particularly with carefully designed states and rewards. Identifies common patterns in state variables (prices, inventory, technical/fundamental indicators) and reward functions (PnL, risk-adjusted return).	Highlights major limitations: lack of explainability, ad-hoc MDP and reward design, limited robustness across markets and time periods, small datasets, and no unified benchmarks. Emphasizes that many RL studies overlook non-stationarity, long-range dependence, and realistic constraints, calling for standardized benchmarking, contextual RL, multi-agent approaches, and offline RL.
Yasin & Gill, 2024 – "Reinforcement Learning Framework for Quantitative Trading"	Two years of daily data for AAPL/APPL from 2020-01-01 to 2022-01-01.	DQN, A2C and PPO.	Shows that normalization and indicator selection have minimal impact on performance compared to algorithm choice; DQN generally performs best, while A2C and PPO are more hyperparameter-sensitive. Demonstrates that rich indicator sets can be processed but don't guarantee stable profits, reinforcing the need for careful reward and environment design.	Experiments cover only one stock over a short period with daily data; limited to backtests without live deployment and minimal hyperparameter exploration. Acknowledges strategy degradation and overfitting risks, with simplified transaction costs and market impact. The framework lacks systematic comparison across different assets or market regimes.
Dempster & Leemans, 2006 – "An Automated FX Trading System Using Adaptive Reinforcement Learning"	High-frequency Euro–Dollar FX data (tick/1-minute level) from Jan 2000–Jan 2002.	A layered adaptive reinforcement learning architecture based on recurrent reinforcement learning (RRL). Layer 1 produces trading signals; Layer 2 performs risk and performance management (e.g., trailing stop-loss); Layer 3 optimises risk-aversion and other parameters. The system learns to adjust position sizes dynamically.	Shows that an adaptive RL-based FX system achieves substantial positive PnL (5104 pips; 26% annually) versus buy-and-hold losses (−1636 pips; −8% annually) over 2000–2002. The layered design demonstrates how RL can integrate explicit risk overlays (e.g., stop-loss rules) to mitigate human behavioural biases.	Focuses on a single FX pair in one historical period, leaving robustness across other pairs and timeframes unclear. Market frictions, evolving microstructure, and post-2002 regime shifts aren't explored. The RRL framework predates modern deep RL and doesn't leverage richer state representations like deep features or multiple assets.

Table 1. Summary of Key Studies in Reinforcement Learning for Stock Trading

The next chapter will discuss about the methodology which was implemented for building the project and will dive deep into the technical aspects in detail.

3. METHODOLOGY

3.1 Research Design

The study utilised a simulation-based experimental methodology where reinforcement learning agents were trained and evaluated in a controlled trading environment. This approach allowed for careful testing of algorithmic performance without the risks and ethical concerns associated with live market trading. The experimental design was structured to address two primary research questions:

- Which reinforcement learning algorithm demonstrates superior performance in automated stock trading?
- Does the inclusion of custom technical indicators enhance trading outcomes compared to using standard OHLCV data alone?

3.2 Experimental Framework

Phase 1: Baseline Performance Evaluation

- RL algorithms were trained and tested on standard OHLCV (Open, High, Low, Close, Volume) datasets.
- This established baseline performance metrics for each algorithm.
- Results provided a foundation for comparing algorithmic effectiveness using stock market data.

Phase 2: Enhanced Feature Evaluation

- The same RL algorithms were retrained and tested on augmented datasets incorporating custom technical indicators (RSI, SMA, OBV).
- This phase assessed whether additional market signals improved trading decision-making.
- Direct comparison with Phase 1 results revealed the impact of feature engineering on model performance.

3.3 Control Variables and Consistency

To ensure fair comparison and reliable results, several control measures were implemented:

- **Consistent Training Parameters:** All algorithms used identical timesteps (20,000), learning rates, and environmental configurations.
- **Uniform Data Split:** Training data (days 50-130) and testing data (days 131-165) remained consistent across all experiments.
- **Standardised Evaluation Metrics:** Performance was assessed using the same metrics—total reward, total profit, and win rate for all algorithms and datasets.
- **Identical Market Context:** All three companies (NVIDIA, AMD, Intel) operated within the same time period (January-August 2025), experiencing similar macroeconomic conditions.

3.4 Sampling Techniques

This study used a **Purposive sampling technique** to select the dataset, focusing on three major GPU technology companies—NVIDIA, AMD, and Intel chosen intentionally for their comparable market capitalisation, technological relevance and active trading volumes within the semiconductor sector, ensuring data consistency and representativeness of the targeted domain. The sampling period from January 2025 to August 2025, provided recent data encompassing diverse market conditions that could challenge reinforcement learning (RL) models to adapt and generalise effectively. A total of 165 daily trading records per company formed a robust sample size for evaluating model performance without data redundancy or overfitting concerns. The dataset sourced from the publicly accessible Yahoo Finance (YFinance) API, involved no human participants and therefore contained no personal or sensitive information. Utilising this historical stock data, the project developed and evaluated reinforcement learning models for automated trading within a simulated virtual environment where agents executed trades and received rewards based on profitability. Custom technical indicators such as the Relative Strength Index (RSI), Simple Moving Average (SMA) and On-Balance Volume (OBV) were incorporated to enhance the models' decision-making capabilities. Three RL algorithms—Deep Q-Network (DQN), Proximal Policy Optimisation (PPO) and Advantage Actor-Critic (A2C) were trained and tested on the sampled company data and subsequently compared to evaluate their performance in achieving profitable trading strategies and assessing the impact of the custom indicators on improving the trading outcomes.

3.5 Description of Tools and Technologies

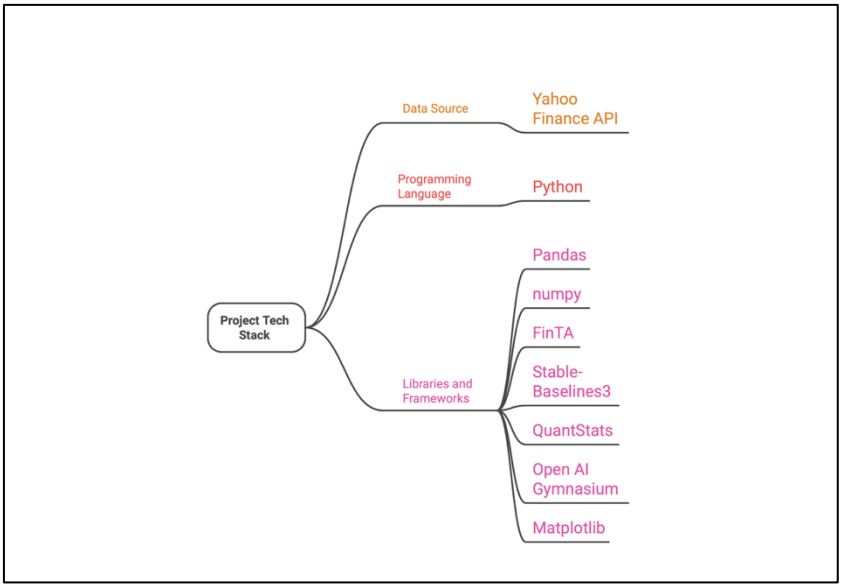


Figure 1. Representation of the Project Tech Stack

- **Yahoo Finance (YFinance) API:** Used to fetch real-time and historical market data for financial analysis and model training.
- **Python:** The core language for data processing, analysis, visualisation and model development.
- **Pandas & NumPy:** For efficient data manipulation, cleaning and numerical computations.

- **FinTA:** Provides a collection of technical analysis indicators for financial data.
- **Stable-Baselines3:** Reinforcement learning framework used to train trading or decision-making agents.
- **QuantStats:** For generating detailed performance reports and metrics for financial strategies.
- **OpenAI Gymnasium:** Offers an environment framework to simulate and evaluate reinforcement learning algorithms.
- **Matplotlib:** Used for creating visualisations of trends, indicators and performance metrics.

3.6 Dataset

The dataset utilised for this study was obtained through the YFinance API, a publicly available Python library that provides access to historical financial market data. Stock price data was collected for three leading GPU technology companies—NVIDIA, AMD and Intel, covering the period from January 2025 to August 2025, having 165 records. The primary dataset consisted of standard financial indicators, Open, High, Low, Close and Volume (OHLCV) values. In addition to the base dataset, a secondary dataset was constructed by adding custom technical indicators, including the Relative Strength Index (RSI), Simple Moving Average (SMA) and On-Balance Volume (OBV).

A Python script **'fetch_data'** was created to download the historical stock data for Nvidia, AMD and Intel, the datasets were stored as CSV files in the 'Datasets' folder.

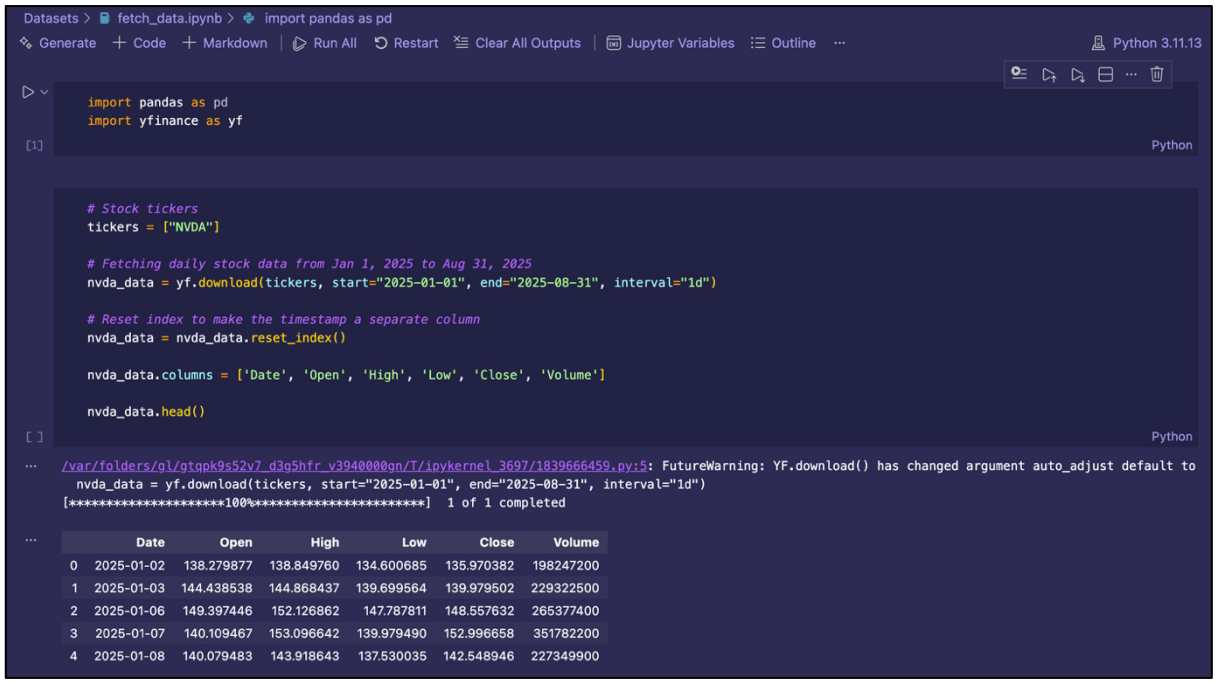


Figure 2. Script for gathering the historical stock data using the Yfinance API

3.7 General Analysis to understand the Datasets

This general analysis examines the short-term market performance of three major semiconductor companies—AMD, Intel and Nvidia using daily historical stock data from January to August 2025. After importing the datasets into Pandas dataframes, descriptive

statistics were computed to summarize central tendencies and variability in closing, high, low, and volume values. Time-series visualizations were then created using Matplotlib and Plotly to compare price movements and trading volumes over time. The methodology focuses on visually identifying trends, volatility patterns and volume-price relationships across the three equities, enabling comparative interpretation of market behaviour within the same industry over the selected time window.

3.8 Reinforcement Learning Algorithms

DQN, A2C, and PPO excel in short-term stock prediction because they effectively combine deep function approximation, stability under market noise and adaptive learning, making them robust in fast-changing environments with subtle signals. DQN efficiently handles discrete trading actions, A2C balances exploration and exploitation through its actor critic design and PPO ensures stable policy updates via clipping. In comparison, simpler methods like Policy Gradient, REINFORCE, Q-learning and SARSA struggle with instability or continuous state spaces, while DDPG and TD3 are complex for discrete trading scenarios.

Deep Q-Learning: A model-free algorithm that learns an optimal policy by approximating a Q-function that estimates how good it is to perform a given action in a given state. It attempts to find a policy that maximises cumulative rewards.

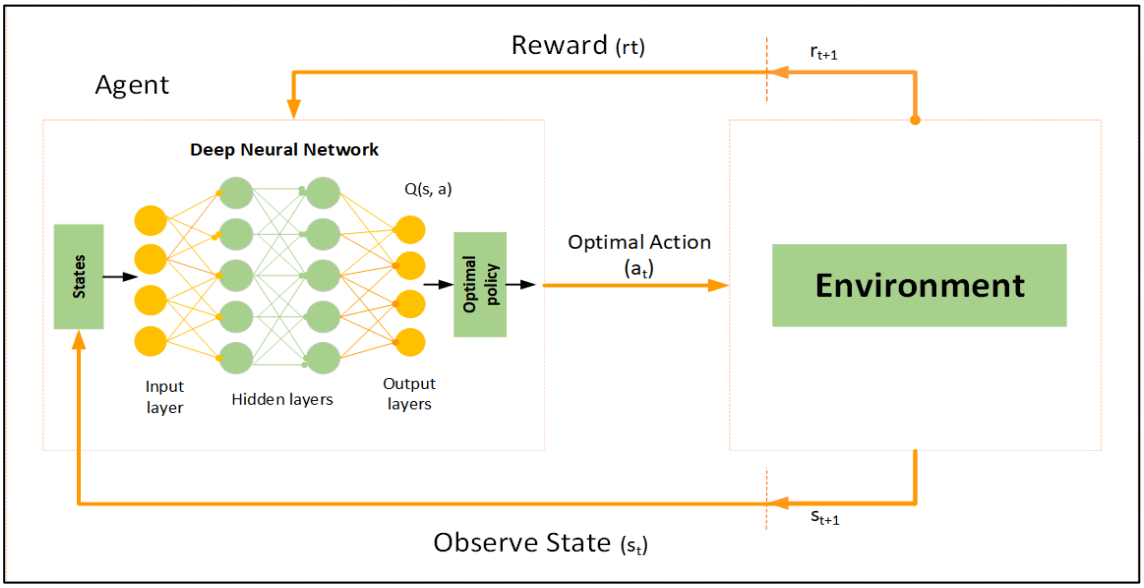


Figure 3. Architecture of Deep Q-Networks

The Deep Q-Network (DQN) architecture combines traditional Q-learning with deep neural networks to enable effective decision-making in high-dimensional state spaces. At its core, the DQN uses a deep convolutional neural network (or fully connected network, depending on the input type) to approximate the Q-value function Q(s, a; θ), where (s) represents the state, (a) the action and θ the learnable parameters. The network takes the current state as input and outputs a vector of Q-values corresponding to all possible actions. To stabilize training and avoid divergence issues inherent in using neural networks for Q-learning, DQN employs two key mechanisms: experience replay and a target network. Experience replay stores past transitions (s, a, r, s') in a memory buffer, allowing the agent to learn from randomised mini-batches that break the correlation between continuous experiences. The target network, a

periodically updated copy of the primary Q-network, provides stable target values during training. Together, these components enable the DQN to learn robust policies that approximate optimal action-value functions, allowing agents to perform complex sequential decision-making tasks in dynamic environments.

Proximal Policy Optimisation: Proximal Policy Optimisation (PPO) is a reinforcement learning algorithm that allows the agents to improve their actions while keeping learning steady. It directly updates the policy like other policy gradient methods but limits the magnitude of large destabilising changes. This balance between maximising rewards and keeping updates minimal makes PPO simpler, more reliable and suitable for various applications.

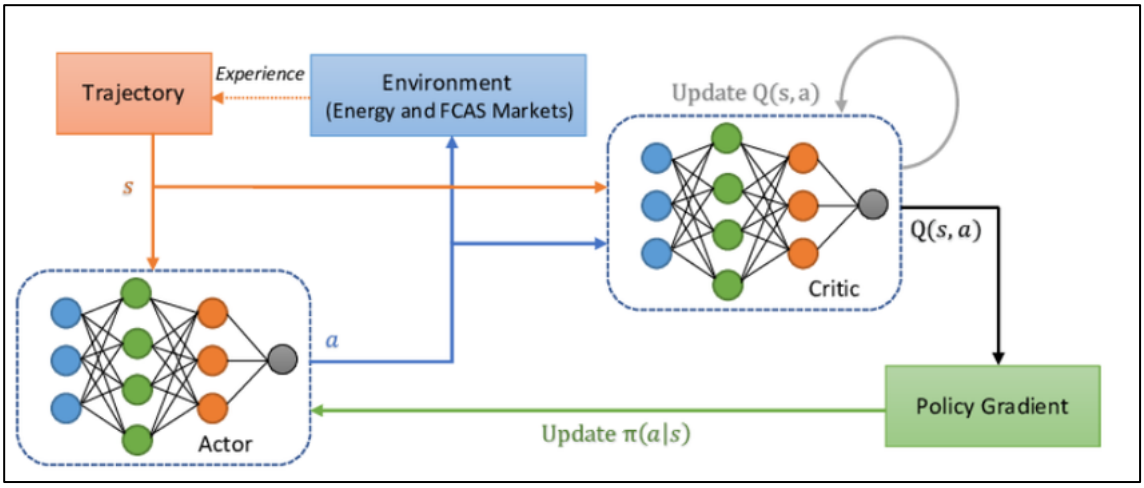


Figure 4. Architecture of Proximal Policy Optimisation

PPO employs an actor-critic structure, where two neural networks operate in tandem: the actor network (policy network) and the critic network (value function estimator). The actor network takes the current state as input and outputs a probability distribution over possible actions, parameterised by $\pi\theta(a|s)$. The critic network, on the other hand, estimates the state-value function $V\phi(s)$, providing a baseline that helps reduce the variance of policy gradient updates. PPO introduces a clipped objective function to constrain policy updates within a trust region, preventing large deviations that can destabilise learning. This is achieved by clipping the probability ratio between the new and old policies, ensuring that the policy does not change too abruptly in a single update. Both networks are typically implemented using deep neural architectures, often multilayer perceptron or convolutional networks, depending on the complexity of the input space. Through this design, PPO achieves a balance between exploration and stability, enabling reliable convergence and high performance across a wide range of continuous and discrete control tasks.

Actor-Critic Algorithm: It is a reinforcement learning algorithm that is a combination of two parts i.e. the Actor, which selects actions, and the Critic, which offers an evaluation. It provides better learning to the agent as it balances decision-making and critique. In the actor-critic algorithm, the actor learns to make decisions, and the critic verifies how good the decisions are. This two-fold utility helps the agent learn to perform new actions and capitalise on learned experiences, and optimise the learning process of balancing.

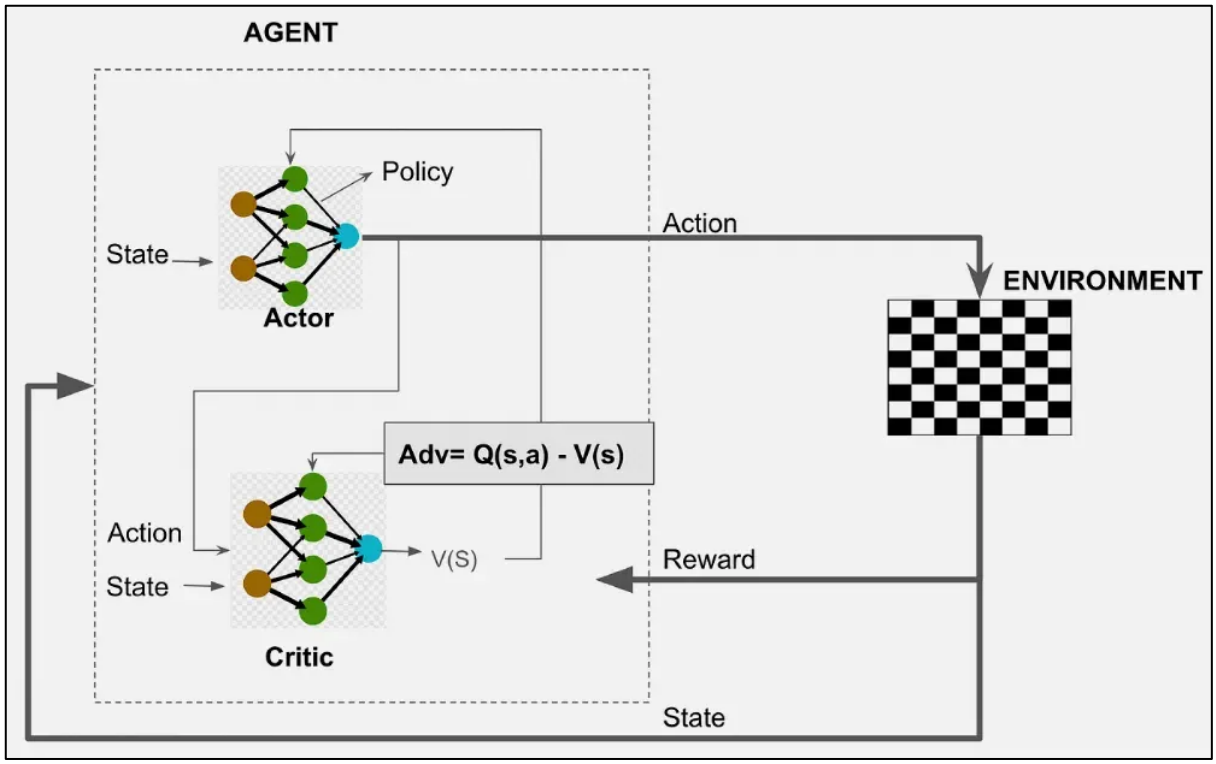


Figure 5. Architecture of Actor-Critic Algorithm

The Advantage Actor-Critic (A2C) architecture is a synchronous and more stable variant of the actor-critic reinforcement learning framework, designed to efficiently learn both the policy and value functions. It consists of two primary components: the actor network and the critic network, which may share some initial layers to extract common feature representations from the input state. The actor network outputs a parameterized policy $\pi\theta(a|s)$, representing the probability distribution over possible actions given the current state, while the critic network estimates the state-value function $V\phi(s)$, used to evaluate how good a particular state is under the current policy. A2C introduces the concept of an advantage function, defined as $A(s,a)=Q(s,a)-V(s)$, to measure how much better an action is compared to the average action at a given state, thereby reducing variance in the policy gradient estimation. Unlike asynchronous methods such as A3C, A2C synchronously aggregates gradients from multiple parallel environments before updating the model parameters, leading to more stable and efficient learning. This architecture, often implemented with deep neural networks, allows the agent to balance exploration and exploitation while maintaining stable convergence in both discrete and continuous action spaces.

3.9 Implementation of RL Algorithms on Standard OHLCV Datasets

This section of the methodology chapter discusses the procedural framework and implementation details regarding the application of three reinforcement learning algorithms to standard Open-High-Low-Close-Volume (OHLCV) datasets derived from three corporate entities Nvidia, AMD and Intel. The goal is to train a model that can decide when to buy or sell a stock based on historical price data and technical indicators. The same process and parameters were used for all three companies so that their results can be compared fairly.

3.9.1 Data Import and Pre-Processing

- Historical stock data for AMD, Intel, and NVIDIA were downloaded from Yahoo Finance. The data included the Open, High, Low, Close, and Volume values for each trading day (total 165 trading days).
- Columns which were not relevant were dropped, the 'Date' column was converted to 'datetime' and 'Volume' column was converted to 'float'.

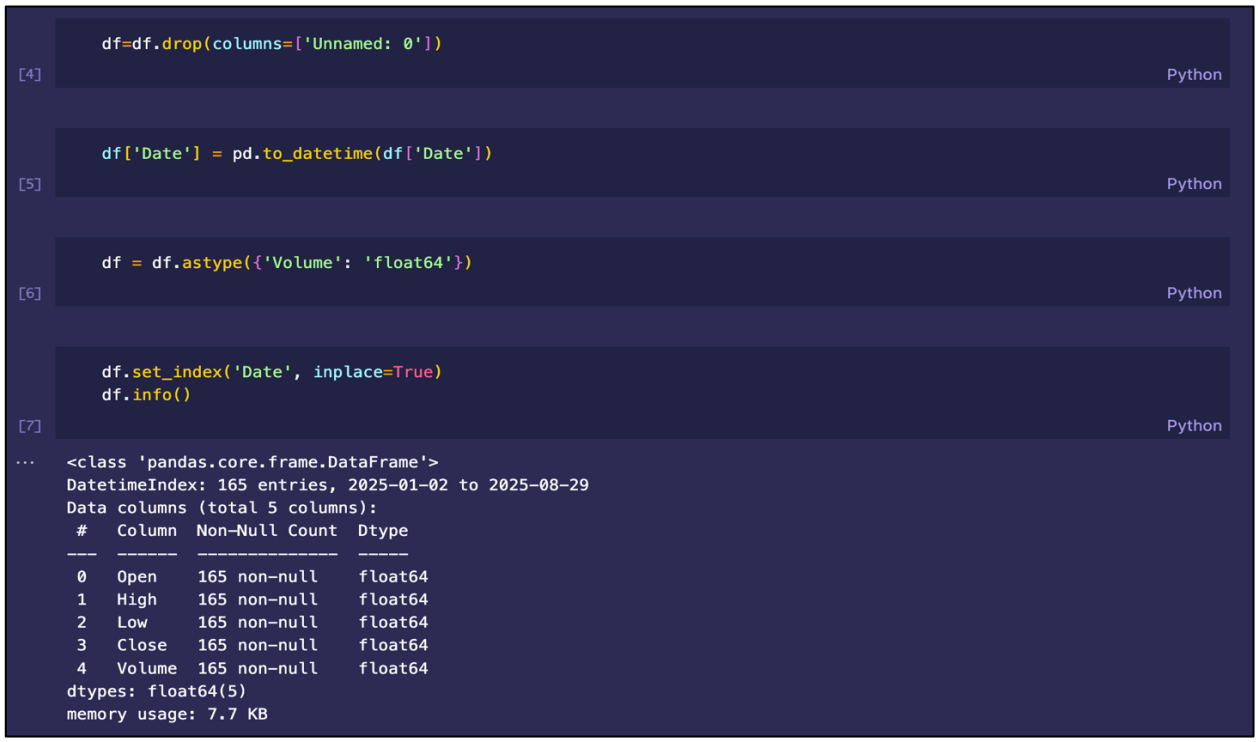


Figure 6. Data Pre-Processing

3.9.2 Building the Trading Environment

- A trading environment was created for each stock using the OpenAI Gymnasium.
- The agent interacts with this environment by choosing one of two possible actions: Buy – purchase the stock, or Sell – sell the stock.
- After each action, the environment updates the portfolio and gives a reward based on profit or loss.
- The model gets a reward based on whether the trade is good or bad.

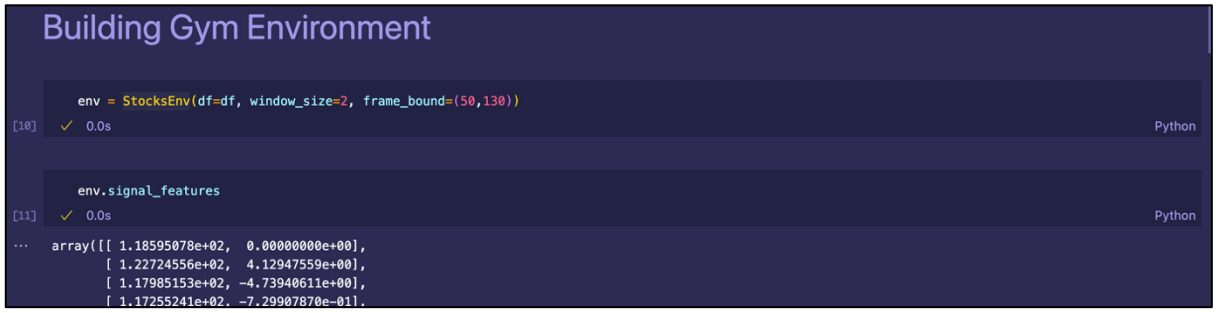


Figure 7. Creating Trading Environment

```
env.action_space
```
[12] Python

```
Discrete(2)
```

```
state, info = env.reset()

while True:
    action = env.action_space.sample()
    n_state, reward, terminated, truncated, info = env.step(action)
    done = terminated or truncated
    if done:
        print("info:", info)
        break

plt.figure(figsize=(15,6))
plt.cla()
env.render_all()
plt.show()
```
[13] Python

```
info: {'total_reward': np.float32(10.158737), 'total_profit': np.float32(0.9608755), 'position': <Positions.Short: 0
```

Figure 8. Action Spaces available and Testing the Environment

Figure 7 demonstrates the setup of a trading environment built with OpenAI Gymnasium using the 'StocksEnv' environment type. The window size is configured to 2, meaning the agent analyses the two most recent price data points before making each decision. The frame bound is set to 50-130, which defines the data range used during training—specifically, the agent processes timesteps from the 50th to the 130th data point.

Figure 8 illustrates the available action spaces in the trading environment. The agent can perform two discrete actions: buy or sell. A testing environment is then established to validate the agent's trading execution using random samples from the dataset.

3.9.3 Model Selection, Implementation and Evaluation

- Wrap the environment using DummyVecEnv for compatibility with stable-baselines3.
- Instantiate the RL model (A2C, DQN, or PPO) with the environment.
- Train the model using model.learn(total_timesteps=20000).
- Reset the environment for evaluation.
- Use the trained model to predict actions and interact with the environment.
- Render and visualize the results.

Training

```
env_maker = lambda: StocksEnv(df=df, window_size=2, frame_bound=(50,130))
env = DummyVecEnv([env_maker])
```
[14] ✓ 0.0s Python

```
model=DQN('MlpPolicy', env, verbose=1)
model.learn(total_timesteps=20000)
```
[15] ✓ 4.0s Python

Figure 9. DQN Implementation

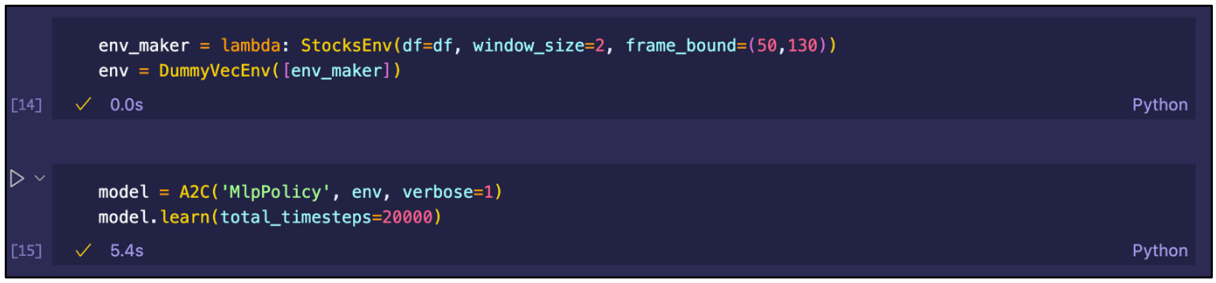


Figure 10. A2C Implementation

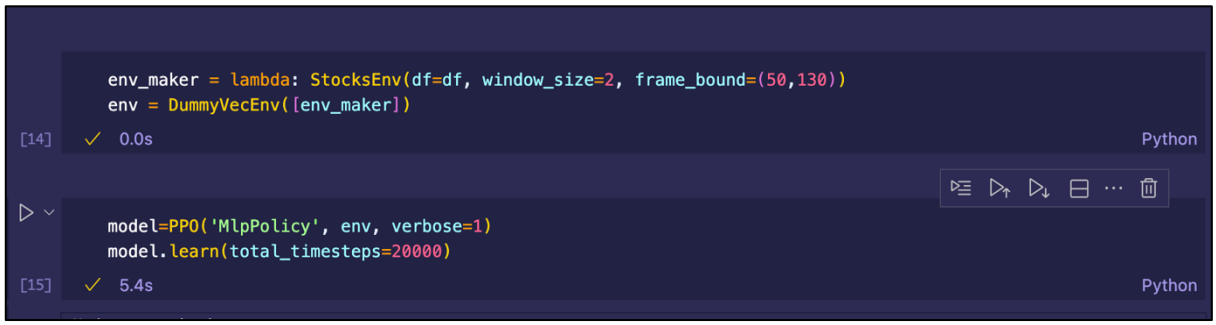


Figure 11. PPO Implementation

Figure 12. Model Evaluation

As shown in Figure 9-11, three reinforcement learning models—DQN (Deep Q-Network), A2C (Advantage Actor-Critic), and PPO (Proximal Policy Optimisation) were selected, implemented, and evaluated to analyse their performance on Nvidia, AMD and Intel Stock Datasets. The model selection phase focused on choosing these algorithms due to their distinct learning strategies: A2C's balance between policy and value optimisation, DQN's efficiency in discrete action spaces, and PPO's robustness in handling policy updates.

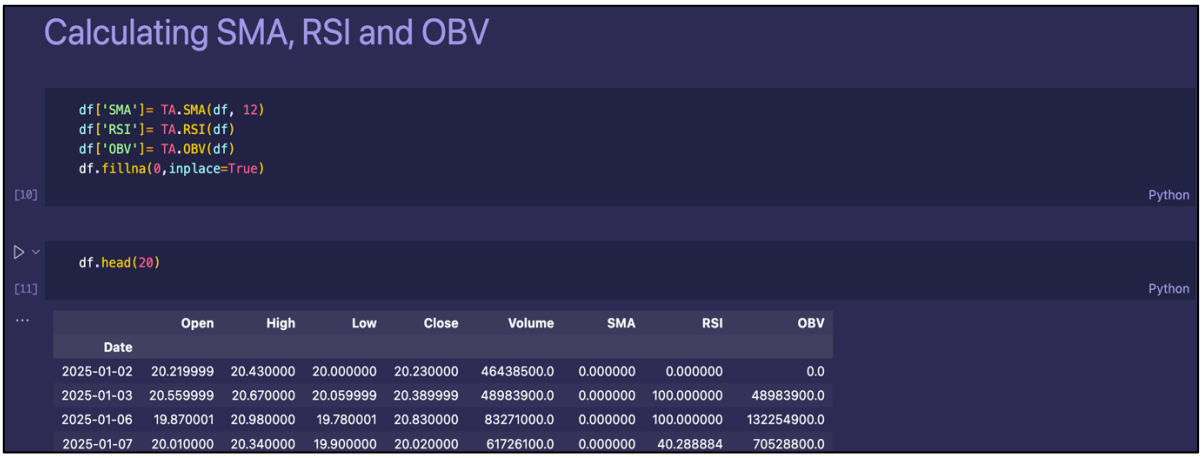
For model evaluation, as illustrated in Figure 12, the frame boundary was updated and set to (131, 165), indicating that the model processes data from the 131st to the 165th timestep. Each model is subsequently utilised for prediction, with the agent receiving rewards based on performance outcomes. The agent's stock buying and selling actions are then visualised through a time-series graph generated using the Matplotlib library.

3.10 Implementation of RL Algorithms on Datasets with Custom Technical Indicators

The datasets were enhanced with custom technical indicators to capture trend, momentum, and volume-based market dynamics. Specifically, three indicators were utilised: the Relative Strength Index (RSI), the Simple Moving Average (SMA), and the On-Balance Volume (OBV). RSI was employed to measure the speed and magnitude of recent price changes, helping identify potential overbought or oversold conditions. SMA was calculated to smooth short-term fluctuations and reveal underlying price trends over a defined window. OBV, a cumulative indicator that relates volume to price movement, was incorporated to gauge the strength of buying and selling pressure. Together, these indicators provided a comprehensive technical representation of each stock's behaviour, serving as informative features for model training.

3.10.1 Data Import and Pre-processing

- Historical data for NVIDIA, AMD and Intel was loaded, containing fields Open, High, Low, Close and Volume containing 165 trading days records.
- Columns which were not relevant were dropped, the 'Date' column was converted to 'datetime' and the 'Volume' column was converted to 'float'.
- To enhance the dataset and provide the trading agent with better contextual information, several custom technical indicators were computed. These included commonly used market indicators like Simple Moving Averages, Relative Strength Index and On-Balance Volume derived through mathematical transformations of price and volume data.



Date	Open	High	Low	Close	Volume	SMA	RSI	OBV
2025-01-02	20.219999	20.430000	20.000000	20.230000	46438500.0	0.000000	0.000000	0.0
2025-01-03	20.559999	20.670000	20.059999	20.389999	48983900.0	0.000000	100.000000	48983900.0
2025-01-06	19.870001	20.980000	19.780001	20.830000	83271000.0	0.000000	100.000000	132254900.0
2025-01-07	20.010000	20.340000	19.900000	20.020000	61726100.0	0.000000	40.288884	70528800.0

Figure 13. Incorporating Custom Technical Indicators

3.10.2 Building the Trading Environment

A custom OpenAI Gymnasium environment was created to simulate the trading process. This environment defined the agent's interactions with the market through the following components:

- A trading environment was created for each stock using the OpenAI Gymnasium.
- The agent interacts with this environment by choosing one of two possible actions: Buy - purchase the stock or Sell - sell the stock.
- After each action, the environment updates the portfolio and gives a reward based on profit or loss.
- Train the model using model.learn(total_timesteps=20000).

```python
def add_signal(env):
    start=env.frame_bound[0] - env.window_size
    end=env.frame_bound[1]
    prices=env.df.loc[:,'Low'].to_numpy()[start:end]
    signal_features=env.df.loc[:,['Low','Volume','SMA','RSI','OBV']].to_numpy()[start:end]
    return prices, signal_features
```

[15] Python

```python
class MyCustomEnv(StocksEnv):
    _process_data = add_signal

env2 = MyCustomEnv(df=df, window_size=2, frame_bound=(50,130))
```

[16] Python

Figure 14. Custom Environment

Building Environment and Training

```python
env_maker = lambda:env2
env = DummyVecEnv([env_maker])
```

[17] ✓ 0.0s Python

```python
model = A2C('MlpPolicy', env, verbose=1)
model.learn(total_timesteps=20000)
```

[18] ✓ 5.9s Python

```
Using cpu device
-----------------------------------
| time/                 |         |
|    fps                | 3200    |
|    iterations         | 100     |
|    time_elapsed       | 0       |
|    total_timesteps    | 500     |
| train/                |         |
|    entropy_loss       | -0.673  |
|    explained_variance | -0.019  |
|    learning_rate      | 0.0007  |
|    n_updates          | 99      |
|    policy_loss        | 5.1     |
|    value_loss         | 93.8    |
-----------------------------------
```

Figure 15. Building the Trading Environment

As shown in Figure 15, a new trading environment was created by updating the variable to 'env2'.

3.10.3 Model Selection, Implementation and Evaluation

- Three reinforcement learning algorithms—A2C, DQN, and PPO were applied on the custom dataset to develop and evaluate trading strategies.
- The trading environment was initialised using DummyVecEnv for Stable Baselines3 compatibility, with key parameters like learning rate defined. Each model was trained over multiple timesteps to maximise cumulative rewards, then evaluated on test data using metrics such as total reward, total profit and win rate to assess performance and trading efficiency.

Figure 16. Evaluation of Algorithms on Custom Dataset

According to Figure 14-16, the custom trading environment extends the StocksEnv class from the gym_anytrading library to incorporate multiple technical indicators for enhanced decision-making. While the base environment typically uses only closing prices, this implementation adds five features through a modified _process_data method: Low price, Volume, Simple Moving Average (SMA with 12-period window), Relative Strength Index (RSI with 14-period default), and On-Balance Volume (OBV). These indicators are calculated using the FinTA library and provide the reinforcement learning agent with richer market signals encompassing trend direction (SMA), momentum (RSI), volume dynamics (OBV), and price extremes (Low). The environment operates with a 2-day window size and evaluates agent performance on a test period (days 131-165) separate from the training period (days 50-130) using Nvidia, AMD and Intel stock data from January to August 2025. This multi-indicator approach allows the agent to learn more nuanced trading strategies based on comprehensive market conditions rather than price action alone.

3.11 Methodology Conclusion

The Methodology chapter investigated the application of Reinforcement Learning algorithms to automated stock trading by implementing and comparing three prominent RL methods—Deep Q-Network (DQN), Proximal Policy Optimisation (PPO), and Advantage Actor-Critic (A2C)—on historical stock data from three leading GPU technology companies: NVIDIA, AMD, and Intel. The research employed a systematic experimental design that evaluated these algorithms across two distinct dataset configurations: a standard dataset containing only OHLCV (Open, High, Low, Close, Volume) features, and an enhanced dataset incorporating custom technical indicators, including Relative Strength Index (RSI), Simple Moving Average (SMA), and On-Balance Volume (OBV).

The methodology involved creating a simulated trading environment using OpenAI Gymnasium, where RL agents could learn and execute trading strategies through iterative interaction. Each algorithm was trained over 20,000 timesteps on data spanning days 50-130 and subsequently evaluated on out-of-sample data from days 131-165, covering the period from January to August 2025. The study addressed two primary research objectives: (1) comparing the relative performance of different RL algorithms in achieving profitable trading strategies,

and (2) assessing whether the inclusion of custom technical indicators enhances trading outcomes.

3.12 Ethical Considerations

From an ethical perspective, the study adhered to standards of transparency, data integrity, and responsible AI experimentation. All datasets used were publicly available, ensuring full compliance with open data usage policies and eliminating any risk of privacy violation. The study maintained data authenticity by avoiding any manual manipulation of market records, ensuring that model training and evaluation reflected genuine market behaviours. Additionally, all reinforcement learning algorithms were developed and executed in a simulated trading environment using OpenAI Gymnasium, preventing real-world financial impact or speculative use of the models. The results and models were tailored with academic integrity, avoiding exaggeration of performance metrics and clearly stating experimental limitations. These steps made sure that the research was ethical, easy to repeat and followed standard rules for data science, finance and AI research.

4. Findings and Results

This chapter presents the findings from the empirical implementation of three reinforcement learning algorithms—Advantage Actor-Critic (A2C), Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) applied to historical stock market data. The investigation focuses on three major GPU companies: NVIDIA Corporation, Advanced Micro Devices (AMD) and Intel Corporation, utilizing historical stock datasets to establish a comprehensive comparative analysis of algorithmic performance.

This research pursues two primary objectives. **First, it evaluates the relative effectiveness of the three reinforcement learning algorithms in generating profitable trading strategies based solely on raw price and volume data. Second, it investigates whether the incorporation of technical indicators specifically the Relative Strength Index (RSI), Simple Moving Average (SMA) and On-Balance Volume (OBV) into the feature space enhances the decision-making capabilities of these algorithms compared to baseline implementations using only OHLCV data.**

By systematically comparing algorithm performance across both standard and custom datasets, this chapter provides empirical evidence regarding the value proposition of feature engineering in reinforcement learning-based trading systems. The findings contribute to the broader understanding of how different reinforcement learning architectures respond to varying levels of market information complexity, with implications for the practical deployment of automated trading strategies in financial markets.

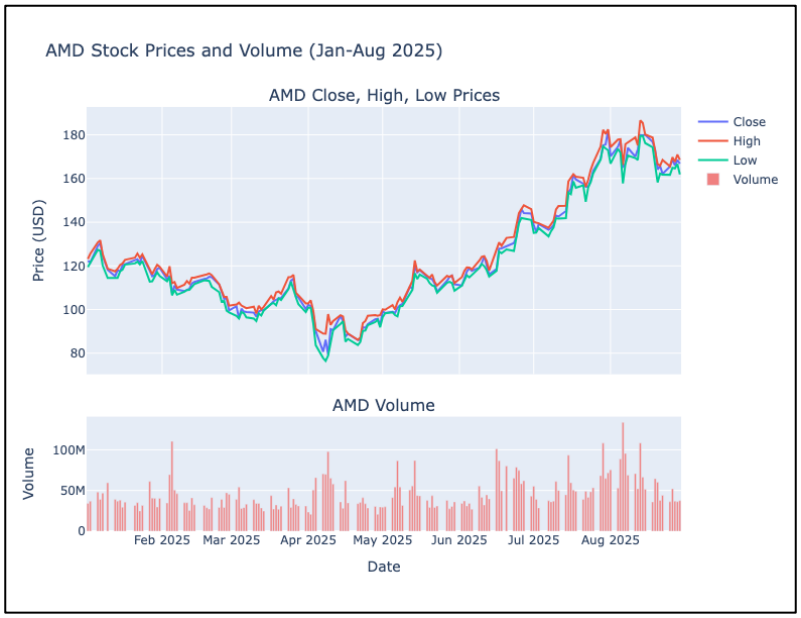


Figure 17. AMD Close, High, Low Prices and Volume over Time

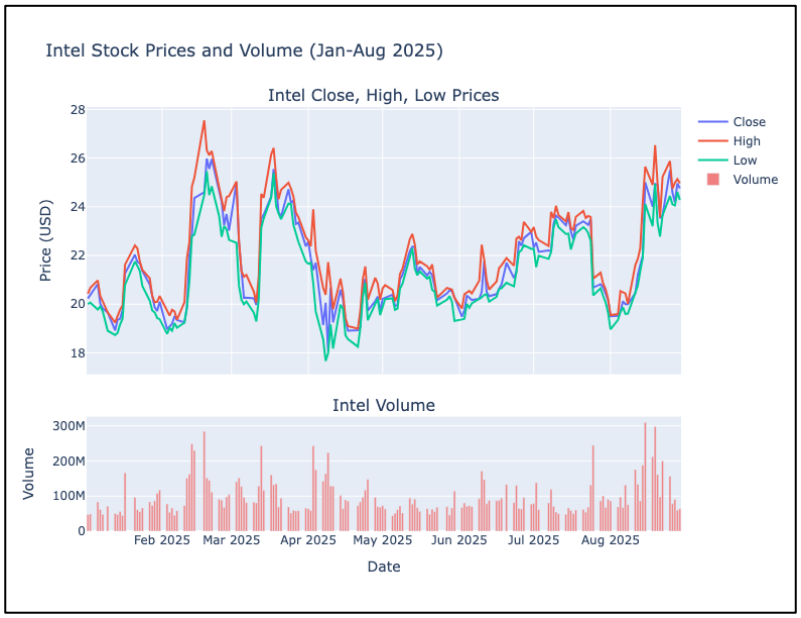


Figure 18. Intel Close, High, Low Prices and Volume over Time

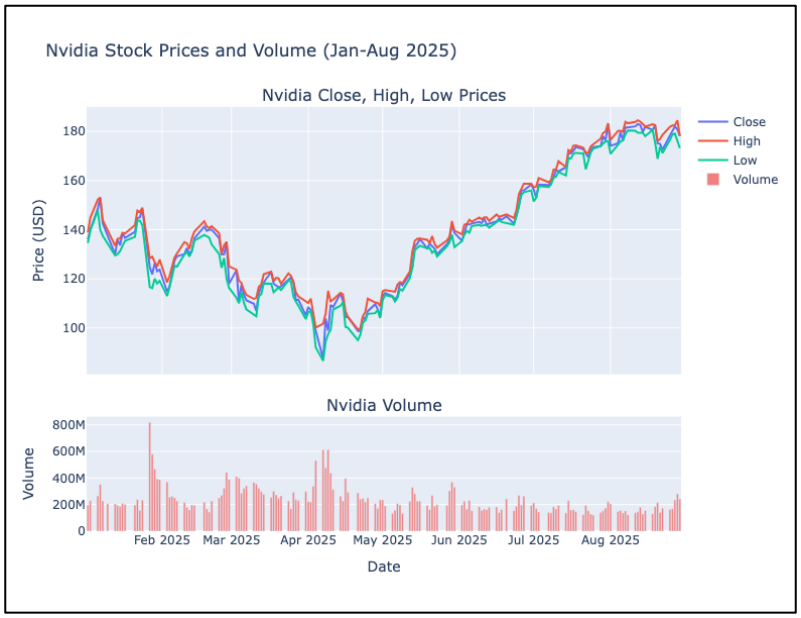


Figure 19. Nvidia Close, High, Low Prices and Volume over Time

The plotted trends in Figure 17, 18 and 19 show that Nvidia exhibits the highest price levels and the strongest upward momentum, consistent with its dominant industry position and continued demand for AI-related hardware. AMD displays moderate growth with noticeable short-term fluctuations, indicating more volatile market behaviour but still following an overall positive trajectory across the period. Intel, appears comparatively stable with smaller price swings, reflecting slower but steadier performance during the observed months. Volume patterns across all three stocks generally reinforce price movements— Nvidia experiences substantial volume spikes aligned with price surges, AMD shows variability consistent with its volatility and Intel maintains more uniform trading activity. Collectively, these results highlight distinct market profiles: NVIDIA as a strong growth leader, AMD as a competitive but more volatile performer, and Intel as a steady, lower-volatility counterpart.

4.1 Variability in Results

A fundamental characteristic of this **experimental framework is that each independent execution of the code yields a different total profit result, even when hyperparameters and datasets remain identical.** This variability in trading performance can be attributed to the high sensitivity of Reinforcement Learning models to initial conditions and the stochastic nature of their optimization trajectory. In the domain of financial time-series, which is characterized by a low signal-to-noise ratio and non-stationary dynamics, these stochastic components specifically probabilistic exploration, random weight initialization and experience replay sampling create varying learning paths. Consequently, distinct execution runs may result in agents converging on different local optima rather than a single global strategy. **This phenomenon highlights that the agent is not merely memorizing historical data but is actively navigating a complex probability space where minor changes in the early stages of training can propagate into significant disparities in final accumulated profit and risk-adjusted returns.**

- **Sensitivity to Initial Conditions:** This refers to the "butterfly effect" in neural networks. Because the financial market is complex, starting the learning process with

slightly different random neural weights can lead the "brain" of the model to interpret the same data in fundamentally different ways.

- **Low Signal-to-Noise Ratio:** Financial data is volatile (messy). The random sampling in RL algorithms can sometimes cause the agent to mistake random market noise for a real pattern (overfitting) in one run, while correctly ignoring it in another, leading to different results.
- **Local vs. Global Optimum:** In optimization theory, a "local optimum" is a solution that looks like the best possible strategy within a small range, but isn't the best overall (Global Optimum). The randomness in RL means one run might get stuck in a "good enough" local strategy, while another run finds a "superior" global strategy.
- **Path Dependence:** This emphasizes that the order in which the agent experiences trades matters. If the agent makes a lucky random guess (exploration) early in Run 1 but an unlucky one in Run 2, their future learning paths will diverge significantly.

The analysis focuses on two key performance metrics extracted from the final evaluation episodes:

- **Total Reward:** Cumulative reward obtained during the test episode.
- **Total Profit:** Final portfolio value relative to the initial value (e.g., 0.90 = –10% return, 1.00 = break-even, 1.10 = +10% return).

4.2 Comparative Performance Across Algorithms on Standard OHLCV Datasets

In the figures (20 to 37) presented below regarding the agent trading stocks using Reinforcement Learning, the horizontal axis denotes the temporal progression measured in discrete timesteps, while the vertical axis indicates the monetary valuation of the respective equity securities.

4.2.1 DQN Results

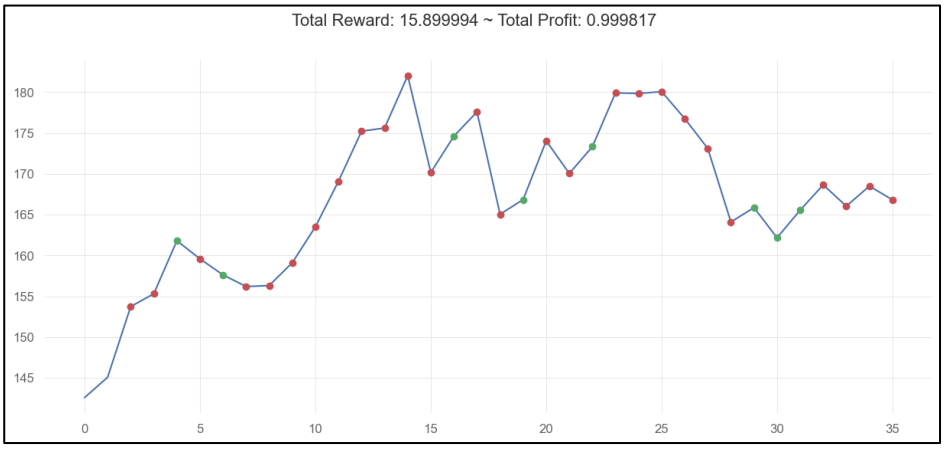


Figure 20. DQN-Based Trading Agent Performance on AMD Stock

Figure 21. DQN-Based Trading Agent Performance on Intel Stock

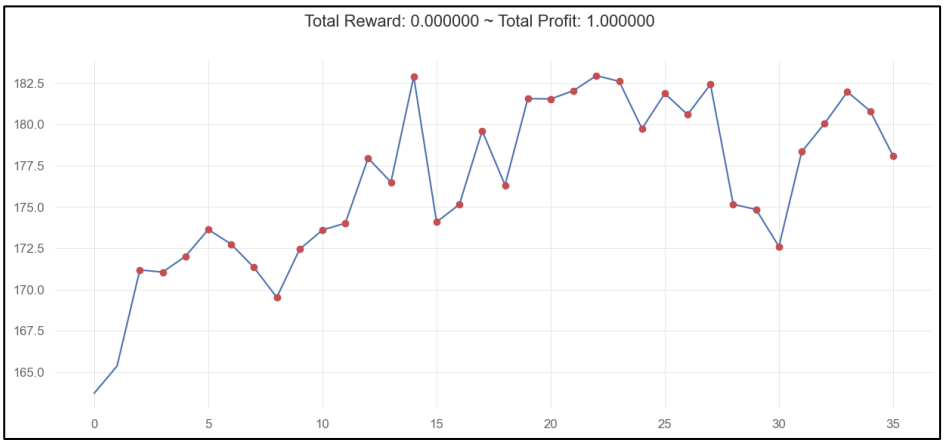


Figure 22. DQN-Based Trading Agent Performance on Nvidia Stock

Stock	Total Profit (Range)
AMD	0.9020 – 0.9998
Intel	0.6852 – 0.9812
Nvidia	0.9062 – 1.00

Table 2. DQN Algorithm Results

Overall, DQN tends to hover near break-even on AMD and Nvidia but can suffer substantial drawdowns on Intel. This reflects its sensitivity to noisy Q-value estimates and the possibility of converging to sub-optimal trading policies due to replay-buffer sampling and local optima.

4.2.2 PPO Results

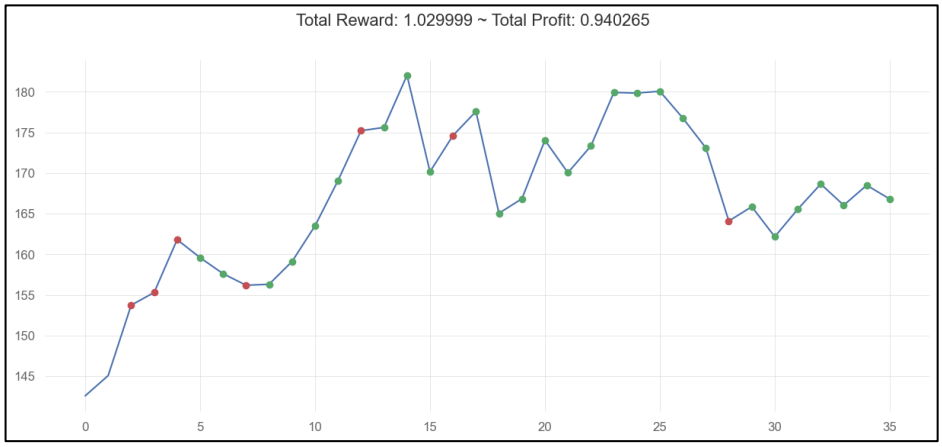


Figure 23. PPO-Based Trading Agent Performance on AMD Stock

Figure 24. PPO-Based Trading Agent Performance on Intel Stock

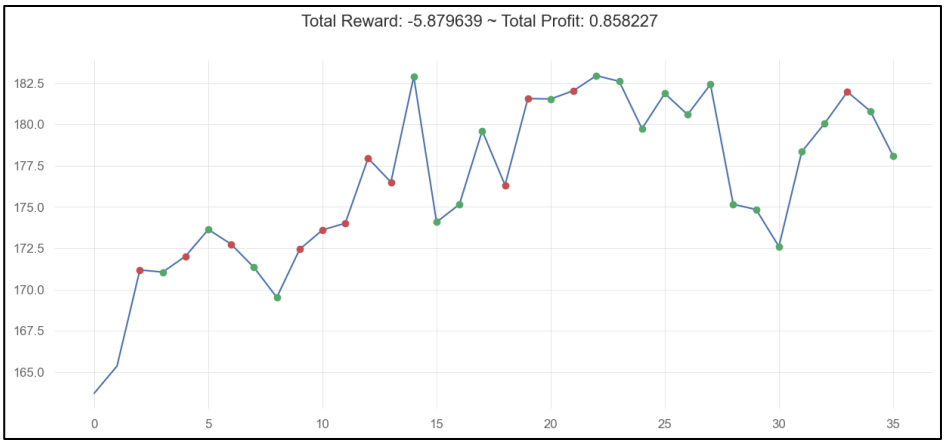


Figure 25. PPO-Based Trading Agent Performance on Nvidia Stock

Stock	Total Profit (Range)
AMD	0.9020 – 0.9402
Intel	0.6852 – 0.8236
Nvidia	0.9062 – 0.8582

Table 3. PPO Algorithm Results

Across all three assets, PPO underperforms both DQN and A2C in terms of best-case profit, never approaching or surpassing break-even in these runs. This suggests that, under the chosen hyperparameters and data horizon, the conservative nature of PPO's clipped updates may limit its ability to discover aggressive but profitable trading strategies.

4.2.3 A2C Results

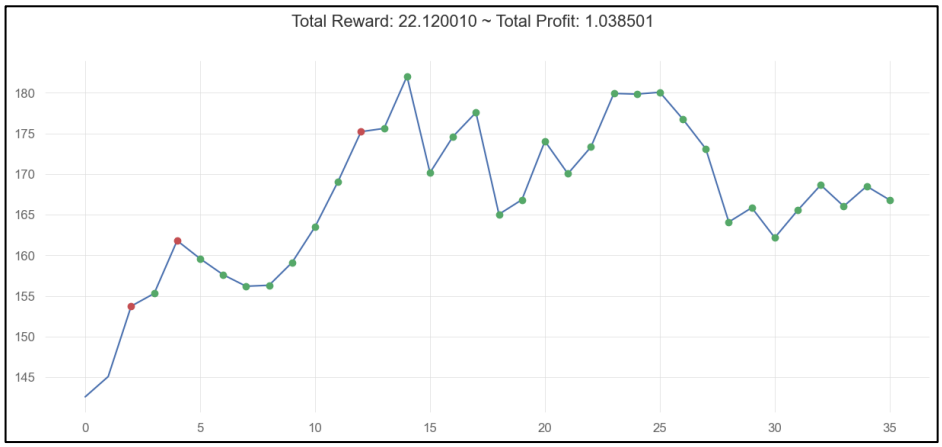


Figure 26. A2C-Based Trading Agent Performance on AMD Stock

Figure 27. A2C-Based Trading Agent Performance on Intel Stock

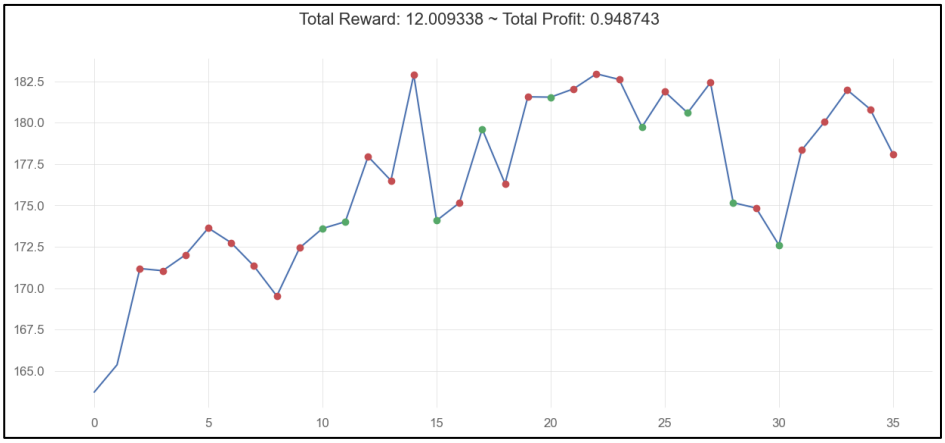


Figure 28. A2C-Based Trading Agent Performance on Nvidia Stock

Stock	Total Profit (Range)
AMD	0.9020 –1.0385
Intel	0.6852 – 0.9081
Nvidia	0.9062 – 0.9487

Table 4. A2C Algorithm Results

Notably, A2C is the only algorithm that achieves a run with total profit above 1.0 on AMD, indicating that its actor–critic architecture can, in some instances, extract more effective trading signals from the same OHLCV data. However, its lower bounds, particularly on Intel, still reflect meaningful downside risk and confirm that the algorithm is far from consistently profitable.

Overall, the comparative analysis on the standard OHLCV datasets shows that the three reinforcement learning algorithms exhibit distinct but consistently fragile performance profiles across AMD, Intel and Nvidia stocks. DQN generally remains close to break-even on AMD and Nvidia but is prone to significant drawdowns on Intel, reflecting its sensitivity to noisy Q-value estimates and replay-buffer sampling. PPO underperforms both DQN and A2C across all the stocks, not approaching profitability in the evaluated runs, which suggests that its conservatively clipped policy updates may constrain the discovery of more profitable trading behaviours under the given data horizon and hyperparameter settings. A2C delivers the strongest best-case outcome, being the only algorithm to achieve a total profit above 1.0 on AMD; however, its performance still spans a wide range and includes substantial losses, particularly on Intel, indicating that this advantage is not robust. Taken together, these findings demonstrate that while architectural choices influence relative performance, none of the algorithms achieves stable, consistently profitable trading on raw OHLCV data alone, underscoring both the difficulty of the financial control problem and the need for better feature representations and careful model design in practical deployments.

4.3 Comparative Performance Across Algorithms on Custom Technical Datasets

4.3.1 DQN Results on Custom Dataset

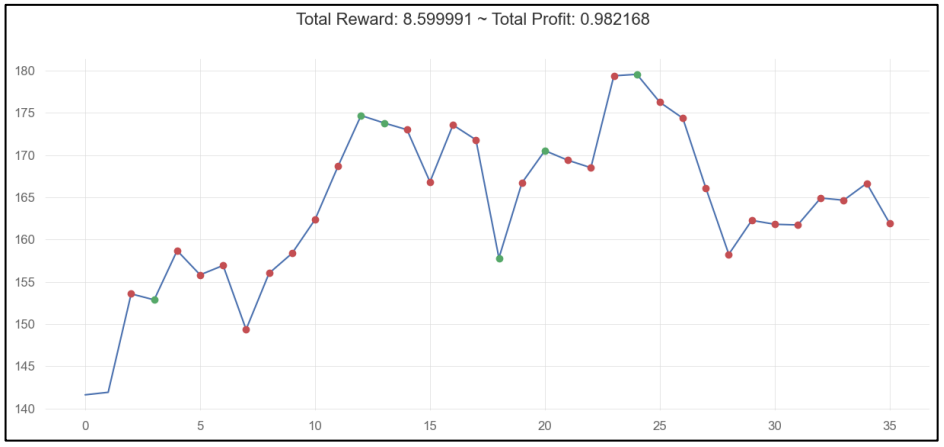


Figure 29. DQN-Based Trading Agent Performance on AMD Stock with Custom Indicators

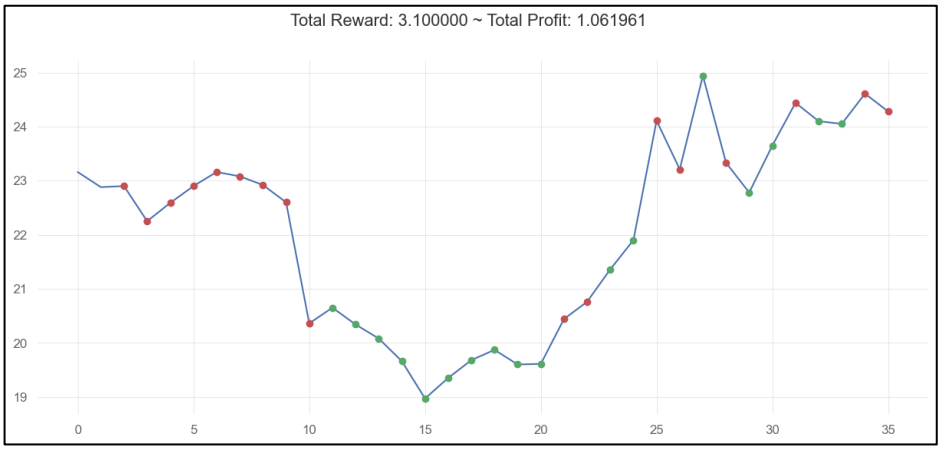


Figure 30. DQN-Based Trading Agent Performance on Intel Stock with Custom Indicators

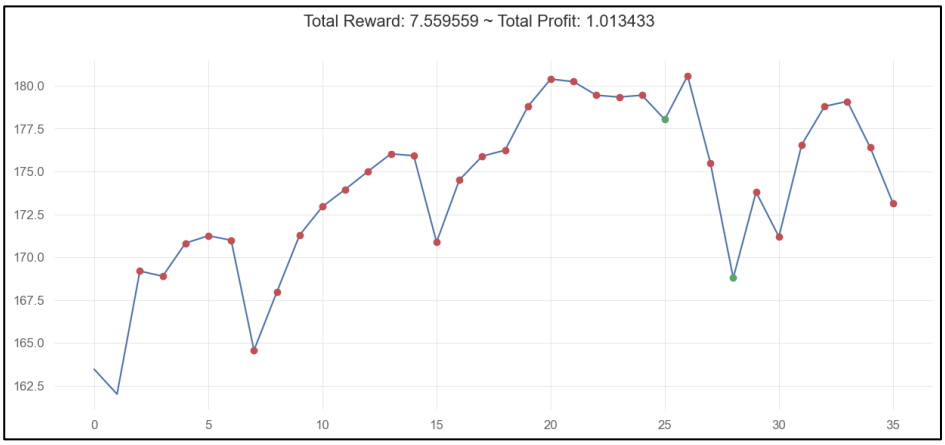


Figure 31. DQN-Based Trading Agent Performance on Nvidia Stock with Custom Indicators

Stock	Total Profit (Range)
AMD	0.9020 – 0.9821
Intel	0.6852 – 1.0619
Nvidia	0.9062 – 1.0134

Table 5. DQN Algorithm Results on Custom Dataset

On the Custom Indicator datasets, DQN exhibits a modest but asset-specific improvement in performance. For AMD, the total profit range remains below break-even (0.9020–0.9821), indicating that the inclusion of RSI, SMA and OBV does not meaningfully change its tendency to cluster around losses. Whereas, Intel and Nvidia both show best-case profits above 1.0 (Intel: 0.6852–1.0619; Nvidia: 0.9062–1.0134), suggesting that the enriched feature space occasionally allows DQN to identify more profitable trading regimes on these stocks compared to the baseline OHLCV runs. However, the determination of low lower bounds especially on Intel confirms that this uplift is not consistent and that DQN remains vulnerable to sub-optimal policies and noisy value estimates.

4.3.2 PPO Results on Custom Dataset

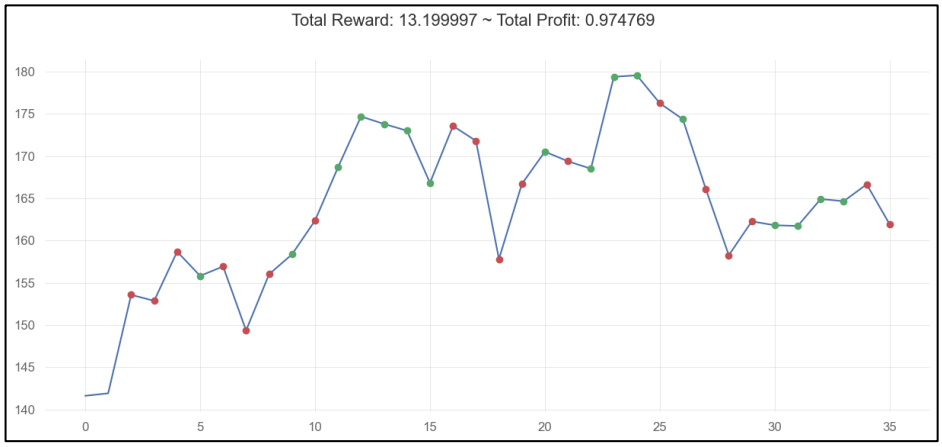


Figure 32. PPO-Based Trading Agent Performance on AMD Stock with Custom Indicators

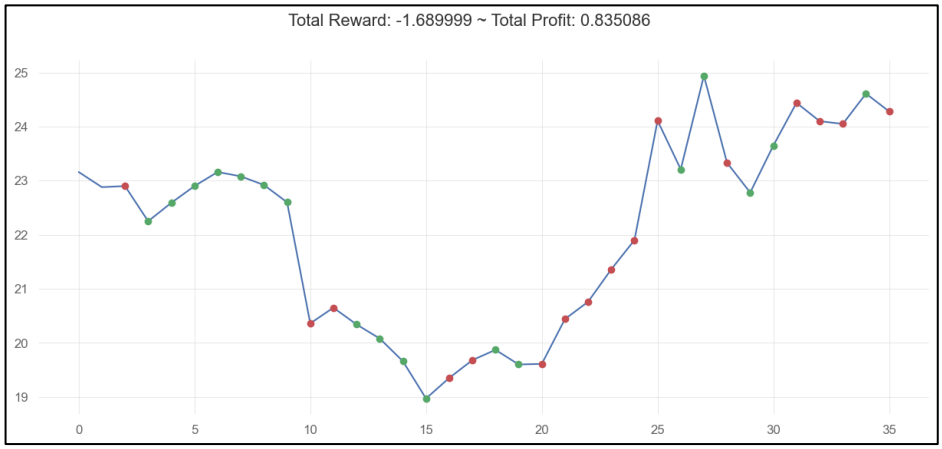


Figure 33. PPO-Based Trading Agent Performance on Intel Stock with Custom Indicators

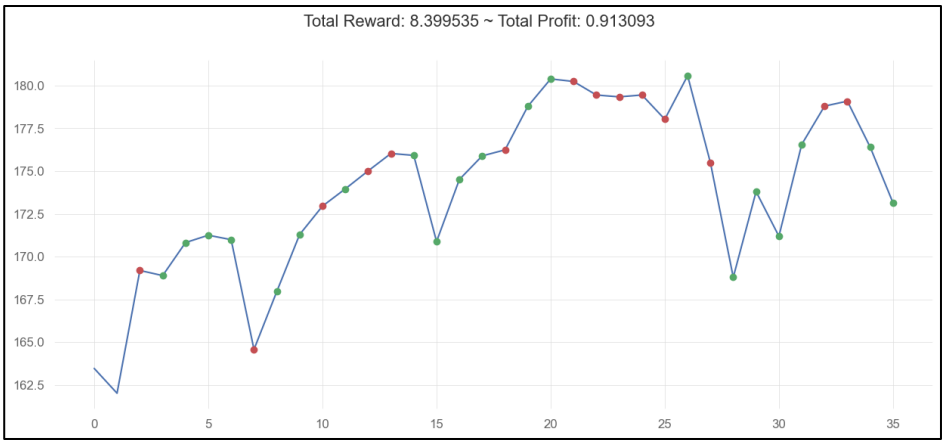


Figure 34. PPO-Based Trading Agent Performance on Nvidia Stock with Custom Indicators

Stock	Total Profit (Range)
AMD	0.9020 – 0.9747
Intel	0.6852 – 0.8350
Nvidia	0.9062 – 0.9130

Table 6. PPO Algorithm Results on Custom Dataset

PPO algorithm continues to underperform compared to DQN and A2C even when trained on the custom technical indicator datasets. Although there is a slight upward shift in best-case profit across all three assets (AMD: 0.9020–0.9747; Intel: 0.6852–0.8350; Nvidia: 0.9062–0.9130), the algorithm still fails to reach or exceed break-even in any of the evaluated runs. This pattern implies that under the current hyperparameter settings and training horizon, PPO's clipped policy updates remain overly conservative: the agent benefits somewhat from the additional features provided by RSI, SMA and OBV, but still does not explore sufficiently aggressive policies to achieve consistently profitable outcomes.

4.3.3 A2C Results on Custom Dataset

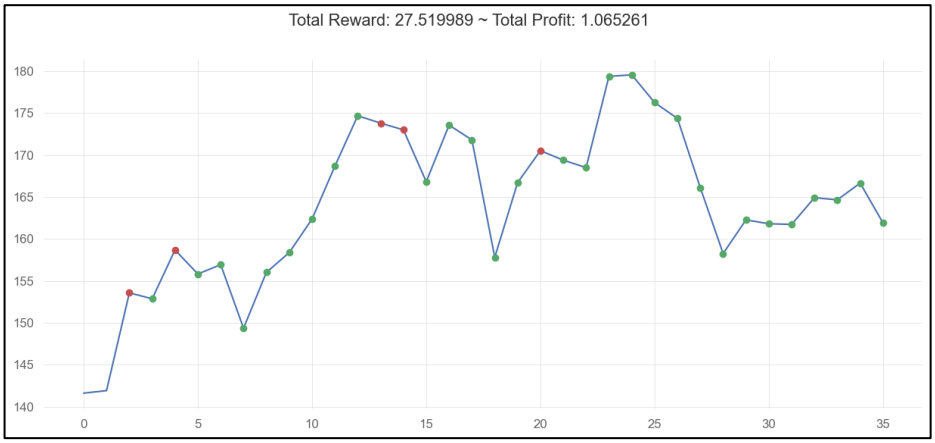


Figure 35. A2C-Based Trading Agent Performance on AMD Stock with Custom Indicators

Figure 36. A2C-Based Trading Agent Performance on Intel Stock with Custom Indicators

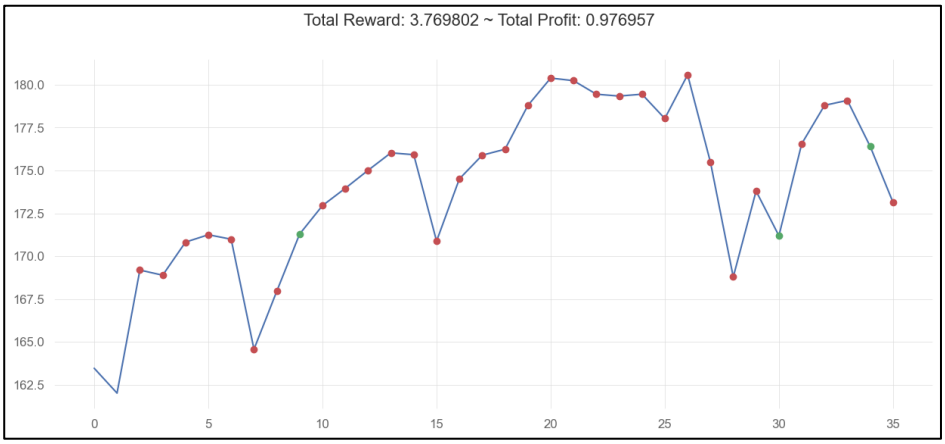


Figure 37. A2C-Based Trading Agent Performance on Nvidia Stock with Custom Indicators

Stock	Total Profit (Range)
AMD	0.9020 – 1.0652
Intel	0.6852 – 1.0749
Nvidia	0.9062 – 0.9769

Table 7. A2C Algorithm Results on Custom Dataset

A2C benefits the most from the inclusion of custom technical indicators, showing a clear improvement in best-case profitability across all three stocks. On AMD and Intel, A2C now achieves total profits well above 1.0 (AMD: 0.9020–1.0652; Intel: 0.6852–1.0749), outperforming its own OHLCV-only runs and indicating that the actor–critic framework is able to leverage the richer feature space to extract more informative trading signals. Nvidia also shows an uplift in best-case profit (0.9062–0.9769), though it remains just below break-even. Despite these gains, the lower bounds remain significantly below 1.0 for all assets, highlighting that A2C's improvement comes with considerable downside risk and variability across runs.

When applied to the standard dataset , custom technical indicators RSI, SMA and OBV showed marginally but improved performance for all three architectures and assets. For the DQN networks, there is a meaningful improvement in performance for Intel and Nvidia, and sometimes even over zero. However, it still shows large drawdowns, especially on Intel. PPO profits were barely improved and all experiments suffered a net loss, suggesting that its cautious update rule, as configured in our experiments, was not able to benefit from the added feature expressiveness. A2C, however, was the most responsive to the indicators and made the most profit in ideal conditions. A2C also turns out to be reliably break-even-plus on AMD and Intel, in runs with both sets of engineered features. However, none of the algorithms becomes strongly or consistently profitable. This suggests that engineered features can be useful in terms of learning and improving the upside potential of this approach, but are unlikely to solve the instability and risk associated with reinforcement learning-based trading in volatile non-stationary financial markets.

4.4 Discussion of Research Questions

Q1. Which reinforcement learning algorithm (DQN, PPO, or A2C) demonstrates superior performance for short-term stock trading when trained on daily OHLCV data?

On daily OHLCV data, A2C performs the best overall:

- A2C is the only algorithm that sometimes makes a total profit above 1.0 for at least one stock.
- DQN usually stays close to break-even.
- PPO performs the worst and generally loses money.
- However, all three algorithms are unstable as their results change a lot from run to run, and they can still make large losses. So even though A2C is the "best" of the three, none of the algorithms is consistently profitable for short-term trading.

Q2. How does the inclusion of custom technical indicators (RSI, SMA, OBV) affect the performance of reinforcement learning trading agents compared to using raw OHLCV data exclusively?

Adding the custom indicators RSI, SMA and OBV on top of OHLCV data gives a small but better improvement:

- All three algorithms do slightly better with indicators than with raw OHLCV alone.
- DQN sometimes finds more profitable trades (especially on Intel and Nvidia) but can still lose a lot.
- PPO improves only a little and still mostly loses money.
- A2C gains the most, it reaches higher best profits and can be profit-making in some runs on AMD and Intel.
- Even so, the agents are still not reliably profitable. The indicators help the models "see" better patterns, but they do not remove the high risk or instability of reinforcement-learning-based trading in real stock data.

5. Conclusion

5.1 Problem Statement and Research Context

This dissertation addressed critical gaps in the application of reinforcement learning to financial markets, specifically the lack of systematic algorithmic comparisons under identical conditions and the poorly understood role of technical indicators in RL-based trading systems. Despite growing interest in applying RL algorithms such as Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C) to stock trading, most existing studies evaluate individual algorithms without standardized frameworks, making it difficult to determine which approaches are most suitable for specific trading scenarios. Furthermore, while traditional technical indicators are widely used in technical analysis, their contribution to RL agent performance had not been thoroughly investigated. This research systematically evaluated these three prominent RL algorithms across standard OHLCV datasets and

augmented datasets incorporating RSI, SMA, and OBV indicators, using daily stock data from NVIDIA, AMD, and Intel spanning January to August 2025.

5.2 Key Findings

Several main findings from the empirical study of the work are relevant for forthcoming work in the context of RL algorithm development for use in financial trading applications. First, when the three RL algorithms to be compared purely focus on standard OHLCV features, the A2C algorithm is the most successful. A2C was also the only algorithm that got total returns above 1 (though 1.0385 on AMD) over its runs. DQN usually performed marginally above break-even for AMD and Nvidia, but got very underwhelming results for Intel (maximum drawdown of 0.6852) and was the algorithm with the lowest returns. Its performance is likely due to the sensitivity of the policy update to the noisy Q-value estimates, leading to sub-optimal convergence. Likewise, the PPO agent performed no better than the DQN and A2C agents and did not get close to breaking even for all three stocks probably due to the clipped policy updates.

Second, adding technical indicators (RSI, SMA, and OBV) for the three trading algorithms into the data incrementally improved all three algorithms and assets. The A2C algorithm was the most impacted by additional features, consistently achieving best-case profits above the breakeven point for both AMD (1.0652) and Intel (1.0749). These results show that the actor-critic framework is able to extract better trading signals with richer representations. DQN improved on both Intel and Nvidia, and was occasionally profitable (1.0619 and 1.0134), but still showed large drawdowns. PPO also improved slightly, but all runs lost, indicating that the added features on their own were not enough to help the algorithm overcome its conservatism for that particular configuration.

Third, and most critically, a fundamental characteristic emerged across all experimental conditions, significant run-to-run variability in trading performance, even when hyperparameters and datasets remained identical. This variability reflects RL models' high sensitivity to initial conditions, stochastic exploration strategies, random weight initialization and experience replay sampling in the context of financial time series characterized by low signal-to-noise ratios and non-stationary dynamics. Consequently, none of the algorithms achieved stable, consistently profitable trading across multiple runs, with lower-bound performance remaining substantially below break-even for all three stocks and algorithms. This finding underscores that while architectural choices and feature engineering influence relative performance and upside potential, fundamental challenges remain in applying RL to volatile financial markets.

5.3 Theoretical and Practical Implications

There are vital implications in the findings in terms of reinforcement learning applied in the financial environment. Theoretically, the research contributes in the field of financial reinforcement learning in the manner that there is a distinct and direct comparison provided. Moreover, it reveals the effect of different reinforcement-learning methods in the same environment, the same data, and the same measures, concluding that A2C performs better

compared to DQN and PPO, implying that actor-critic models could be rather useful in financial dilemmas, where the ability to create an optimal policy and an optimal value approximation are both essential. Equally, the distinct improvement provided through the application of the conventional technical tools provides insight into the effectiveness and application of domain knowledge in the technical aspects applied in reinforcement learning systems.

In terms of practical implications, the results provide critical guidance on the use of automated RL trading systems. There is a of run-to-run variability, and in all the algorithms, there are number of losses. This goes on to highlight the risks and realities involved in the application of the RL approach in the live trading environment. The implications highlight the fact that despite the ability of the agents in the RL approach to identify profitable times of trading, they are generally not better than the break-even strategies and in some cases may experience large losses. This provides a reality check on the expectations and hopes traders have about the applicability and effectiveness of the RL approach. In addition, the experimental framework developed in this work provides a means through which other researchers can compare solutions.

5.4 Limitations

There are some restrictions to the interpretation of these results. First, the sample period from January to August 2025 provides a relatively limited scope of market dynamics and may not be fully representative of the various market regimes, risk states and macroeconomic environments that investors may encounter over different time horizons. Second, there were only three firms using GPU technology in the data tested in the study, limiting generalizability to other firms across industries, asset classes or segments with different volatility and trading characteristics. Third, the results focused on an experiment without transaction costs or liquidity constraints and did not consider slippage in trade execution. In real-world trading, these issues would have a meaningful impact on the algorithm's profitability. Fourth, three indicators (RSI, SMA, OBV) were chosen. Results may vary from this set with expert feature engineering, different combinations of indicators, or other preprocessing. Fifth, RL models depend highly on hyperparameter settings, reward function definitions, and training stability. Other setups and hyperparameters could yield better models, but hyperparameter tuning was beyond the scope of this study. Further, due to the randomness nature of the algorithms and run-to-run variability, the absolute performance levels should be considered with caution, and the results should be interpreted comparatively rather than absolutely.

5.5 Future Research Directions

There are several clear paths for future investigation that arise from the present investigation. Firstly, using a longer time horizon, encompassing numerous years with a range of market conditions, such as those present in growing markets, declining markets, high volatility, and periods of financial crisis would provide clearer insight into the efficacy of the algorithms and their ability to generalize. Secondly, using the investigation across a range of assets, rather than the GPU technology industry, including different industry sizes and other types of asset such

as commodities, forex and cryptocurrencies, would provide insight into whether the trends observed in the present investigation are generalizable across different market structures. Thirdly, the incorporation of a range of real trading costs, such as those outlined, would provide a clearer indication of their effectiveness in real world trading conditions.

Fourth, hyperparameter optimization through, e.g., the use of Bayesian optimization and genetic search may help optimize the performance of all three algorithms, and in particular, the profitability of the PPO algorithm, which was rather low. Fifth, other, perhaps even more complex, feature engineering, including, e.g., other technical charts, the use of the sentiment analyses provided by financial news, order book analysis, volatility measures, and deep learning, could potentially help the agents make a better decision. Sixth, the use of ensemble methods, which make use of the predictions provided by all different reinforcement-learning algorithms and hybrid methods, combining reinforcement-learning and quantitative models, could provide better robustness and lower run-to-run variability. Finally, the development of publicly available standard benchmarks and perhaps even libraries, like in other fields, would help advance the field. In terms of application, the results provide valuable insights for individuals wanting to use automated RL trading systems. The high run-to-run fluctuation and frequent losses exhibited by all methods, even those with better features, reveal the harsh realities of trading. The work confirms that despite RL traders occasionally discovering profitable periods, they fail to outperform break-even strategies on a consistent basis and can experience large drawdowns. This fact check is important for banks and individuals who might have high expectations about the performance of RL in the real world. Furthermore, the platform provided in this work can be used in the future and serve as a useful tool in the field, allowing individuals to compare their ideas.

5.6 Closing Statement

This dissertation has demonstrated that while reinforcement learning offers a theoretically compelling framework for automated trading, significant practical challenges remain before these algorithms can reliably generate consistent profits in real-world conditions. The systematic comparison of DQN, PPO, and A2C revealed that A2C shows the most promise, particularly when augmented with technical indicators, yet even this superior performer exhibits substantial variability and downside risk. The research highlights a fundamental tension: the same adaptive learning capabilities that make RL theoretically attractive for capturing complex market patterns also introduce instability and sensitivity to initial conditions in the noisy, non-stationary environments of financial markets.

These findings are not to be viewed as discouraging, but rather as establishing realistic expectations for RL-based trading systems. The improvements observed when incorporating technical indicators validate the importance of bridging classical financial analysis with modern machine learning, suggesting that future progress requires thoughtful integration of domain expertise with algorithmic innovation. While reinforcement learning will undoubtedly remain an active research area as markets evolve and computational resources expand, this study emphasizes that success requires more than algorithmic sophistication, it demands

rigorous evaluation frameworks, realistic acknowledgment of limitations, careful risk management, and continued investigation into whether autonomous agents can consistently outperform simpler strategies in one of the most challenging decision-making environments. The standardized experimental framework provided herein offers a foundation for future research to advance in both theoretical understanding and practical applications of reinforcement learning in quantitative finance.

References

1. Bai, Y., Gao, Y., Wan, R., Zhang, S. and Song, R. (2025) 'A review of reinforcement learning in financial applications'.
2. Bellman, R. (1957) 'A Markovian decision process'.
3. Briola, A., Lezzi, D. and Stocchi, F. (2021) 'PPO-based reinforcement learning for high-frequency trading'.
4. Dakalbab, F., Mojaddidi, M., Jarrah, M. and AlQaisi, H. (2024) 'Artificial intelligence techniques in financial trading: A systematic literature review'.
5. Dempster, M.A.H. and Leemans, V. (2006) 'An automated FX trading system using adaptive reinforcement learning'.
6. Deng, Y., Bao, F., Kong, Y., Ren, Z. and Dai, Q. (2017) 'Deep direct reinforcement learning for financial signal representation and trading'.
7. Dey, R., Kassim, S., Maurya, S., Mahajan, R.A., Kadia, A. and Singh, M. (2025) 'Machine learning-based automated trading strategies for Indian stock market'.
8. Guede-Fernández, A., Martins, J., Fernandez-Gavilanes, M. and Juncal-Martínez, A. (2025) 'Deep learning for algorithmic trading: A systematic review of predictive models and optimization strategies'.
9. Hambly, B., Xu, R. and Yang, H. (2023) 'Recent advances in reinforcement learning for quantitative finance'.
10. Jiang, Z., Xu, D. and Liang, J. (2017) 'A deep reinforcement learning framework for the financial portfolio management problem'.
11. Liu, X.-Y., Yang, H., Zhong, S. and Walid, A. (2018) 'Adaptive trading strategies with deep reinforcement learning methods'.
12. Mnih, V. et al. (2015) 'Human-level control through deep reinforcement learning'.
13. Mnih, V. et al. (2016) 'Asynchronous methods for deep reinforcement learning'.
14. Moody, J. and Saffell, M. (1998) 'Reinforcement learning for trading'.
15. Papia, S.K., Rahman, F., Nashid, S., Akhir, A., Biswas, A. and Islam, A. (2024) 'The role of AI and IT in transforming stock price analysis and decision-making frameworks'.
16. Sahu, R., Verma, R., Kumar, S. and Yadav, S. (2023) 'Deep reinforcement learning approaches for multi-asset portfolio optimization'.
17. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017) 'Proximal policy optimization algorithms'.
18. Teixeira, D.M. and Barbosa, R.S. (2024) 'Stock price prediction in the financial market using machine learning models'.
19. Théate, T. and Ernst, D. (2020) 'An application of deep reinforcement learning to algorithmic trading'.

20. Wang, N. and Zhou, X.Y. (2020) 'Continuous-time mean-variance portfolio selection with reinforcement learning'.
21. Yang, H., Liu, X.-Y., Zhong, S. and Walid, A. (2020) 'Deep reinforcement learning for automated stock trading: An ensemble strategy'.
22. Yasin, A.S. and Gill, P.S. (2024) 'Reinforcement learning framework for quantitative trading'.
23. Zhang, Z., Zohren, S. and Roberts, S. (2019) 'Deep reinforcement learning for trading'.