Deep Reinforcement Learning Approaches for Portfolio Optimisation

Investigating Actor-Critic Algorithms

School of Computer Science & Applied Mathematics
University of the Witwatersrand

Bongani Shube
2112465

Supervised by Dr. Jacob Majakwara

September 21, 2025

Ethics Clearance Number: XX/XX/XX

A report submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science

Abstract

This study explored the use of Deep Reinforcement Learning (DRL) to optimise portfolio allocation by modelling financial data as a Partially Observable Markov Decision Process (POMDP). Actor-crtic algorithms (A2C, DDPG, TD3 and PPO) were employed to learn dynamic asset allocation strategies, to overcome the shortcomings of traditional methods in adapting to volatile and non-stationary market. The study incorporated technical indicators and sentiment analysis, in addition to historical price data, to evaluate their impact within a specially designed trading environment. Back-testing was used to assess the efficacy of the proposed approach using important financial metrics, such as the Sharpe ratio, annualised return, cumulative return, and maximum drawdown. According to the results, RL agents, especially A2C, performed better than passive strategies in terms of risk management and return stability. By adding sentiment data, volatility and drawdowns were decreased, indicating that DRL-based strategies are a viable substitute for conventional approaches when complemented with market data. Nevertheless, there are still issues with computational cost and practicality.

Declaration

I, Bongani Shube, hereby declare the contents of this report to be my own work. This report is submitted for the degree of Master of Science in Data Science at the University of the Witwatersrand. This work has not been submitted to any other university, or for any other degree.

Contents

1 Introduction

Portfolio optimisation involves the application of quantitative methods to determine the optimal combination of assets or financial instruments. The primary goal is to balance the risk and return of the portfolio [1]. One of the methods, the efficient frontier, introduced by Markowitz in 1952, seeks to represent a collection of portfolios that satisfy the primary goal, that is, to optimise the return for a given level of risk [2]. Sharpe, in 1964, expanded on Markowitz's work by formalising an approach for investors that allows them to choose the most suitable portfolio along the efficient frontier according to their individual risk appetite and investment targets [3].

Investors may find it challenging to optimise their portfolios due to the intricacy and ongoing volatility of financial markets. Artificial intelligence (AI) is rapidly advancing the field by simplifying and scaling the portfolio optimisation process. This is a good thing because traditional methods in most cases struggle to adapt to the dynamic and seasonal changes of the financial markets [4]. They make assumptions about the future market behaviour using historical data, and this is inaccurate, given the unpredictability and dynamic nature of the financial market. Relying on past performance to make decisions about the future can be misleading; there are always other factors to consider. However, AI and machine learning (ML) models, which are often built on historical data, have still managed to outperform traditional approaches thanks to their strong predictive abilities [4]. Furthermore, a more sophisticated area of AI called reinforcement learning (RL) takes things a step further, by directly learning decision-making policies from real-time market data. By applying this single technique, various portfolio optimisation problems can be solved, including market prediction, asset allocation or rebalancing as well as market constraints such as slippage, transaction costs, and liquidity limitations.

Deep reinforcement learning (DRL) brings together deep learning (DL) and RL to handle complex, high-dimensional data using deep neural networks. This makes it a powerful approach for tasks like portfolio optimisation [5]. In this study, Markovian principles were used to model financial data, with a focus on asset prices and returns. The Markov Decision Process (MDP), specifically is the key framework in this research, which is a crucial idea in RL [6]. It is possible to view asset prices over time as a process, a sequence of random outcomes. The state is the underlying structure that influences these outcomes at each point in time. A Markov process should follow the Markov property, which explains that the following state depends only on the present state and is not affected by any previous state [4].

In RL, problems are framed as an MDP, which includes an agent, a policy and an environment. The agent's state is monitored by the environment, which also receives its actions, bestows rewards, and advances to the next state [5]. In this study, the environment (portfolio) utilised a continuous action space, enabling the trading agent to adjust portfolio allocations or asset weights optimally over time. The process was framed as Partially Observable Markov Decision Process (POMDP), since in many real world problems, some parts of the underlying state are unobservable, leaving decision makers with limited information. A practical example in finance would be the challenge of observing or accurately estimating the drift in a stock price [6].

The primary objective of the suggested approach was to simulate the real-world portfolio optimisation and trading processes. To achieve this, an environment was developed, that used sentiment analysis scores from news releases along with technical indicators, into the state observation space. The resulting POMDP problem was addressed by using various DRL actor-critic algorithms, which learn effective decision-making strategies in environments involving complex, continuous ranges of actions across many dimensions. The proposed approach was assessed through back-testing, a widely used technique by traders and portfolio managers to assess a strategy's performance using historical data. In the end, the performance of the proposed DRL algorithms was compared using important financial metrics, such as the Sharpe ratio, annualised return, cumulative return and maximum drawdown.

1.1 Research Problem

In the fast-paced and intricate world of financial markets, investors must seek to achieve the highest possible return while effectively managing and reducing risk. Managing risk is one of the central concepts of investing, fundamentally supported by Modern Portfolio Optimisation Theory [2]. There has always been a need to streamline the entire strategy and allow for some simplification of the process. AI techniques are becoming more and more popular as they support and enhance this process.

Mean-variance optimisation (MVO) and other traditional portfolio optimisation techniques rely heavily on historical data to model and forecast future market behaviour. Market dynamics may shift unexpectedly, causing volatility and non-stationarity in the financial markets [4]. Therefore, it is unrealistic to expect future trends to resemble historical data due to the financial market's rapidity and unpredictability.

AI and ML models which also use historical data, have consistently proven to show better results and improve portfolio optimisation through the use of advanced forecasting techniques. In contrast to many conventional AI and ML techniques, RL offers a comprehensive solution by taking into account real-world issues. It considers constraints such as liquidity limitations, transactions costs and slippage, factors that can significantly diminish investment returns [4]. Most importantly, RL has the potential to directly learn decision making policies from market data and make adjustment in real time. In the face of quick market shifts and volatility, this is a particularly potent technique for modelling financial markets, especially when it comes to portfolio allocation and rebalancing.

1.2 Research Aim

This study aimed to develop an integrated solution that simultaneously addressed the prediction and allocation problem for portfolio optimisation using DRL, particularly through actor-critic algorithms. Technical indicators and news sentiment were incorporated into the state observation representation of the actor-critic algorithms to improve their decision-making. The following evaluation metrics were used to compare the competence of the chosen actor-critic algorithms: Sharpe ratio, annualised return, annualised volatility, cumulative return and maximum drawdown.

1.3 Research Objectives

The overall aim of this research was to address the key challenges in portfolio optimisation, which are prediction and allocation. The proposed solution involved the implementation of DRL techniques, particularly actor-critic algorithms, to address this problem collectively. The specific objectives of the research were as follows:

1. To develop a portfolio optimisation framework using DRL, that collectively addresses the key challenges in portfolio optimisation, which are prediction and allocation.
2. To implement the following actor-critic algorithms: A2C, DDPG, TD3, and PPO.
3. To enhance the state representation of the actor-critic algorithm by incorporating technical indicators and sentiment scores in addition to historical price data.
4. To use the following key financial metrics to evaluate and compare the performance of the proposed actor-critic algorithms: Sharpe ratio, annualised return, annualised volatility, cumulative return, and maximum drawdown.

1.4 Research Questions

This research sought to provide clear answers to the following questions:

1. How can a portfolio optimisation framework be developed using DRL, that can collectively address the key challenges in portfolio optimisation, which are prediction and allocation?
2. How can the following actor-critic algorithms be implemented: A2C, DDPG, TD3, and PPO, in portfolio optimisation?
3. How do sentiment scores and technical indicators affect actor-critic algorithms, when incorporated into the state representation?
4. How do the proposed actor-critic perform and compare relative to each other when evaluated by the key financial metrics, which are Sharpe ratio, annualised return, annualised volatility, cumulative return, and maximum drawdown?

2 Literature Review

The environment in DRL consists of a state space, which contains all the relevant information in which the agent must determine its next move. In this context, the environment is the portfolio. Basic methods rely on historical closing prices [7, 8]. However, more complex methods enhance the state representation with open, high, low, close and volume (OHLCV) data as well as technical indicators [7, 8, 9, 10, 11]. This offers a more thorough comprehension of market dynamics. Depending on the approach, there may be a variation in the number of state features; more features imply greater feature richness. For instance, whereas [7] uses a larger set of 10 indicators, [7, 9] use indicators such as the Commodity Channel Index (CCI), Moving Average Convergence Divergence (MACD), and Relative Strength Index (RSI)

It is crucial to remember that employing an excessive number of indicators or state features could impair the method's performance by causing over fitting or redundancy. [7] incorporates sentiment analysis in an effort to represent the market's technical and fundamental analysis in the state space. [7] uses FinBERT, a financial domain natural language processing (NLP) model to integrate sentiment analysis. Models that use market price data may be easier to train but they may miss the complex and key patterns in the financial market. Sentiment analysis allows the agent to integrate qualitative insights from news headlines, thus improving the model's understanding of the overall market. Moreover, it is important to handle sentiment data with caution because it may introduce subjectivity and noise in the model.

Platforms like FinRL have been developed to make it easy to represent the portfolio environment. In dynamic trading environments, a multi-model input strategy is used in order to improve decision making by managing the balance between computation efficiency and informativeness. [10] strongly agrees with this and concludes that portfolio context, raw price data, technical indicators and macroeconomic signals should be incorporated in the state space to make it effective. The way RL agents manage their portfolios is significantly influenced by the action space's design in addition to the state space.

In simple or trending markets discrete action spaces can be used. They involve allocating all capital to a single asset per time step [9]. Discrete actions are often used in value-based methods like deep Q-network (DQN) and policy based methods like Proximal Policy Optimisation (PPO). However, this lacks flexibility for diversified or nuanced trading. A key distinction exists between discrete and continuous action spaces. In contrast, continuous action spaces are better suited for real world trading needs. They enable smoother rebalancing and more adaptive strategies. They are mostly employed by actor-critic methods like Soft Actor-Critic (SAC) and Deep Deterministic Policy Gradient (DDPG). Due to their realism as well as improved training dynamics, studies such as [7, 9, 10, 11, 12] favour continuous action spaces. [7] proposes a rescaled continuous action space in a range of [-1, 1] to represent real share quantities. Overall, continuous action spaces are more suitable for complex and multi asset environments, compared to the straightforward, less flexible discrete alternatives; this offers more realism and flexibility. Notably, [7] critiques the rigidity of discrete actions space models. Moreover, [7] also incorporates partial fills when cash is limited and caps on sell orders based on holdings, which are practical market constraints.

Study [9] finds that in trending markets, discrete-action methods such DQN and PPO excel where focused asset selection is advantageous. On the other hand, in complex and volatile markets, continuous action methods such SAC and DDPG perform better. Moreover, [9] explores Convolutional Neural Networks (CNNs) and Multilayer Perceptrons (MLPs) for feature extraction in connection to state and action space. It further concludes that CNNs outperform MLPs. Among the four RL algorithms mentioned, they are favoured because to their capacity to identify temporal patterns in financial time series data.

In contrast, studies [7] and [12] focus exclusively on (Delayed Deep Deterministic Policy Gradient) TD3, chosen to mitigate DDPG's overestimation bias through innovations like clipped double Q-learning and target policy smoothing. Risk aversion and transaction costs are included in the reward function in studies [7] and [12]. They do not,

however, undergo stress testing in erratic market conditions. Unlike [7], which does not include such encoding. Moreover, [12] enhances win TD3 with CNN and to model temporal dependencies, Long Short-Term Memory (LSTM) layers are used.

Meanwhile, ensemble methods, which are proposed in studies [8] and [11] increase conceptual awareness, adaptability and flexibility. However, they also introduce greater computational complexity and management overhead in the process, because they combine multiple methods into one. [8] combines Advantage Actor-Critic (A2C), DDPG and PPO, selecting agents based on out-of-sample Sharpe ratios. [11] extends this to make a Remake Ensemble, incorporating SAC, TD3, Actor-Critic with Kronecker-Factored Trust Region (ACTKTR), Trust Region Policy Optimisation (TRPO), and further integrates fuzzy logic probabilistic market classification.

Across the literature, core financial metrics such as cumulative return, Sharpe ratio, and maximum drawdown are widely used to assess DRL agents. However, the rigour and rationale behind metric selection vary, exposing both strengths and limitations in each study. [9] adopts a comprehensive and methodical evaluation, analysing the impact of varying look back periods (e.g., 16 vs. 28 time steps) on learning and over fitting risks—especially for CNN-based models. Additionally, it acknowledges the limitations of cumulative log-returns in reward design. Crucially, [9] includes sophisticated metrics such as the Sortino ratio and the Calmar ratio, which encourage a more investor-aligned and risk-sensitive assessment.

In contrast, [7, 8, 12] rely heavily on standardised metrics (Sharpe ratio, total return, annualised volatility) without critically discussing their applicability or limitations. For instance, [7] reports a Sharpe ratio of 2.68 but lacks comparison against benchmarks or deeper analysis. [8] limits the generalisability of the results by using the same set of metrics in a variety of markets without evaluating the impact of market conditions on metric relevance.

[10] and [11] show superficial use of performance indicators, reporting total return and Sharpe ratio with minimal exploration of downside risk or model robustness. This trend of metric minimalism may overstate performance by ignoring volatility sensitivity and real-world trading constraints.

[12] adds value by benchmarking DRL agents against traditional portfolio strategies (e.g., minimum variance, max Sharpe portfolios), providing a more credible empirical baseline. However, it still lacks evaluation of practical factors such as transaction costs, turnover, or strategy stability.

RL research on portfolio rebalancing varies widely in modelling realism, particularly in the trade-off between continuous and periodic rebalancing. Continuous rebalancing enables fast responsiveness to market signals but suffers far high transaction costs and slippage, potentially undermining returns [9]. A practical compromise is provided by periodic strategies (e.g., rebalancing every 10 periods) that lower these costs while capturing medium-term trends.

Some studies incorporate transaction cost-aware environments to compare rebalancing frequencies (e.g., daily vs. bi-weekly), accounting for bid-ask spreads and price gaps, which improves the practical relevance of their findings [9]. In contrast, others adopt overly simplified assumptions, such as zero slippage, unlimited liquidity, and no mar-

ket impact, which may only be appropriate for highly liquid markets [7]. In [7] and [11], additional restrictions that further simplify strategy dynamics and limit flexibility include non-negative cash balances, no short selling, and prioritising sell orders.

More realistic approaches, such as in [12], embed transaction costs and risk aversion into the reward function using an extended Markowitz utility framework. These models, support cost-sensitive and risk-aware learning, penalise high trading volume and return variance. Likewise, to prevent over fitting to high-turnover strategies, [11] sets trade execution limits (e.g., $K\%$ of portfolio value per trade) to reflect liquidity and compliance constraints.

Conversely, studies like [8] lack such constraints, resulting in unbounded environments that may inflate the perceived performance of RL agents by allowing unrealistic trading behaviours.

3 Review of Methods

The prices of an asset or financial instrument over time can be viewed as a Process – a sequence of random outcomes [6]. The underlying structure that influences these outcomes at each point in time t, is referred to as the state s. Finding the possibility of the upcoming states based on the current and past states is the main area of interest.

3.1 Markov Process

The probability of the next states can be modelled using a Markov Process [7]. Given the present state at time t, the future state at time $t+1$ is independent of past states, a characteristic known as the Markov property.

$$P(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_1) = P(s_{t+1} \mid s_t) \tag{1}$$

Markov processes are categorised on the basis of the type of time and state variables involved, whether they are discrete or continuous. Depending on the state space, a Markov process can be discrete or continuous. When the process has a discrete or countable set of possible states, it is specifically referred to as a Markov chain. The same holds true for time.

3.2 Hidden Markov Models

An important model that is based on the Markov principle is the Hidden Markov Model (HMM), which has been widely adopted in finance especially [13]. In an HMM, the observed outcomes O_t are generated by probability distributions that depend on the state s_t of an underlying unobserved Markov process. A typical financial situation is when a stock price process's drift is difficult to predict and unobservable [14].

$$P(s_t \mid s_{t-1}, s_{t-2}, \ldots, s_1) = P(s_t \mid s_{t-1}), \quad t = 2, 3, \ldots$$

$$P(o_t \mid o_{t-1}, \ldots, o_1, s_t, \ldots, s_1) = P(o_t \mid s_t), \quad t \in \mathbb{N}$$

3.3 Markov Decision Process

Building on the concept of Markov processes, which focusses on predicting the probability of upcoming states based on the present state and past states, the aspect of decision making can then be introduced at last. In an MDP, it is assumed that the system's state also follows the Markov property, which implies that past states don't affect present or future states. [14].

$$P_a(s, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{2}$$

The Markov theorisation is outlined solely regarding the state in Eq. 1. The state transition probability, $P_a(s, s')$, is used to describe the environment in an MDP in Eq. 2.

Given the current state s, it represents the possibility of changing to state s' at time $t+1$ by acting a at time t.

In a scenario such as managing an investor's portfolio, the decision-maker must select an appropriate action, such as determining the optimal asset allocation [15]. After the action is taken, the system transitions to a new state in a random manner, typically governed by a stochastic process, such as changes in asset prices. The primary goal is to take actions that optimise the entire process to maximise the expected total rewards over time.

3.4 Partially Observable Markov Decision Process

POMDPs are useful for financial data, much like HMMs. In many scenarios, decision makers have limited information about the state process, which means that some parts of the state cannot be observed [16]. An important example in finance is when a stock price process's drift is invisible and challenging to predict.

Unlike earlier circumstances where the entire state is fully observable, in this case, the state cannot be fully observed [17]. Consequently, policies are assumed to rely solely on the observed history. This creates additional complexity, since both the reward and the transition process may rely on the unobserved parts of the state. The POMDP satisfies the following properties:

$$P(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots) = P(s_{t+1} \mid s_t, a_t)$$
$$P(o_{t+1} \mid o_t, \ldots, o_1, s_{t+1}, a_t, \ldots) = P(o_{t+1} \mid s_{t+1}, a_t)$$

The equations above define the Markovian transition and observation model. Eq. 5 below describes the agent's policy π_t conditioned on its history of observations.

$$\pi_t = \pi(a_t \mid o_1, o_2, \ldots, o_t) \tag{3}$$

3.5 Reinforcement Learning

MDPs are the core concept in RL. A RL problem is framed as an MDP and involves three components, the environment, the agent, and a policy of the state at per time-step, processes the actions it takes, gives rewards $r_t = r(s_t, a_t)$ and transitions to new states. The ultimate aim of RL is to enable the agent to learn a policy that maximises the expected sum of future discounted rewards.

$$G(t) = \sum_{k=t}^{\infty} \gamma^{k-t} r_k = r_t + \gamma G(t+1)$$

The total of the discounted rewards from time t to the terminal state is the return at time $G(t)$. The real value γ in this equation ranges from 0 to 1, referred to as the discount factor. It shows the comparable importance of future rewards compared to instant ones. The closer γ is to 1, the more future rewards are valued. A smaller γ indicates that future rewards are less significant. This discounting process reflects the fact that the further into the future we look, the more uncertain and harder it is to predict, hence emphasising more on immediate rewards.

A policy π dictates the agent's behaviour and which actions to take in different states [18]. How good it is to be in a given state following a specific policy is quantified by the value function:

$$V^{\pi}(s) = \mathbb{E}\left(G(t) \mid s_t = s; \pi\right) \tag{4}$$

"The total discounted reward from state s, also known as the expected return, is outlined in Eq. 4. " As an alternative, The action-value function determines whether a specific action a can be carried out in state s and proceeding to adhere to policy π:

$$Q^{\pi}(s, a) = \mathbb{E}\left(G(t) \mid s_t = s, a_t = a; \pi\right) \tag{5}$$

Bellman equations can be used to compute these expectations recursively, which has to do with the value of a state or state-action pair to immediate rewards and the value of successor states [18]. Therefore, we can revise Eqs. 4 and 5 in the following manner:

$$V^{\pi}(s) = \mathbb{E}\left(r_t + \gamma V^{\pi}(s')\right)$$

$$Q^{\pi}(s, a) = \mathbb{E}\left(r_t + \gamma \mathbb{E}(Q^{\pi}(s', a))\right)$$

The following Bellman equation is used to find the best behavioural plan:

$$Q^{*}(s, a) = \mathbb{E}\left(r_t + \gamma \max_{\pi} Q^{\pi}(s', a)\right)$$

This involves determining the best possible action-value function, that is outlined as the maximum of Q-values across all policies π. Maximising the expected long-term is the goal of RL, by finding the optimal policy. The optimal policy π^* can be derived by selecting the action that yields the highest Q-value, once $Q^*(s, a)$ is known:

$$\pi^{*} = \arg\max_{a} Q^{\pi}(s, a)$$

This forms an iterative loop consisting of two key steps:

1. Policy estimation - Estimate $Q^*(s, a)$ according to the existing policy.
2. Policy refinement - Refine the policy through the selection of actions that produce the maximum value according to $Q^*(s, a)$.

3.6 Actor-Critic Algorithms

RL methods for estimating MDPs may be categorised into three main types, value function approximation, policy approximation, and actor-critic techniques [15]. Actor-critic techniques integrate both value-based and policy-based strategies, the actor is responsible for updating the policy, and the critic is responsible for evaluating the policy through

value function estimation. The actor then uses the value estimates of the critic to improve learning stability, although the critic improves its value prediction according to the updated actor policy. Since actor-critic methods do not utilise an environment model, they are considered model-free and learn solely from direct experience. [13].

In RL, especially within the Q-learning framework, the agent aims to adjust parameters by minimising the discrepancy between the predicted Q-values and the target values, which are based on observed rewards. One popular technique is to use a parameterised function to approximate the value of performing an action in a specific state. This function could be a neural network or any differentiable model where θ_Q represents its learnable parameters, typically denoted as $Q(s, a; \theta_Q)$.

The update process uses a loss function grounded in the Bellman equation. The loss captures the squared discrepancy between the current Q-value approximation and the desired return, containing both the immediate reward and the discounted estimate of future rewards:

$$J(\theta_Q) = \mathbb{E}\left[\left(r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_Q) - Q(s_t, a_t; \theta_Q)\right)^2\right]$$

Once training has sufficiently converged, the agent can act optimally by choosing actions that maximise the estimated Q-value per state:

$$a^* = \arg\max_a Q^*(s, a; \theta_Q)$$

The state-value function may be used to formalise the main objective, which is to improve the expected performance of the policy. Beyond value-based methods, policy-based approaches explicitly model the policy using a function $\pi_{\theta_\mu}(a \mid s)$, where θ_μ are the tunable parameters.

$$J(\theta_\mu) = V^{\pi_\theta}(s)$$

This function is optimised to directly increase the probability of choosing the high-reward actions using gradient ascent. The actor-critic architecture is a hybrid approach that divides duties between two models. By fusing the advantages of value-based and policy-based approaches, the actor-critic setup facilitates effective learning.

1. By directing the agent towards more lucrative actions, the actor learns the policy by updating θ_μ.
2. By estimating the Q-function and adjusting θ_Q appropriately, the critic assesses the actor's performance.

3.7 Deep Reinforcement Learning

DRL integrates the capabilities of DL and RL to build more advanced AI systems [5]. The primary reason for integrating DL into RL is to leverage the scalability of deep neural networks, especially when handling high-dimensional spaces.

4 Methodology

The actor network handled the policy and decision-making of the model. It processed the current state through several fully connected layers to produce continuous values

that indicate the magnitude of actions, such as the percentage of an asset to buy or sell. This decision-making process was influenced by feedback from the critic, utilising policy gradients to optimise the policy.

In the model, the critic network served as the value function, determining how advantageous the current state was to yield high returns in the future. It helped to compute the advantage, which measured the effectiveness of a taken action compared to the expected outcome. The critic's feedback helped the actor network adjust its policy by providing a benchmark for expected future rewards.

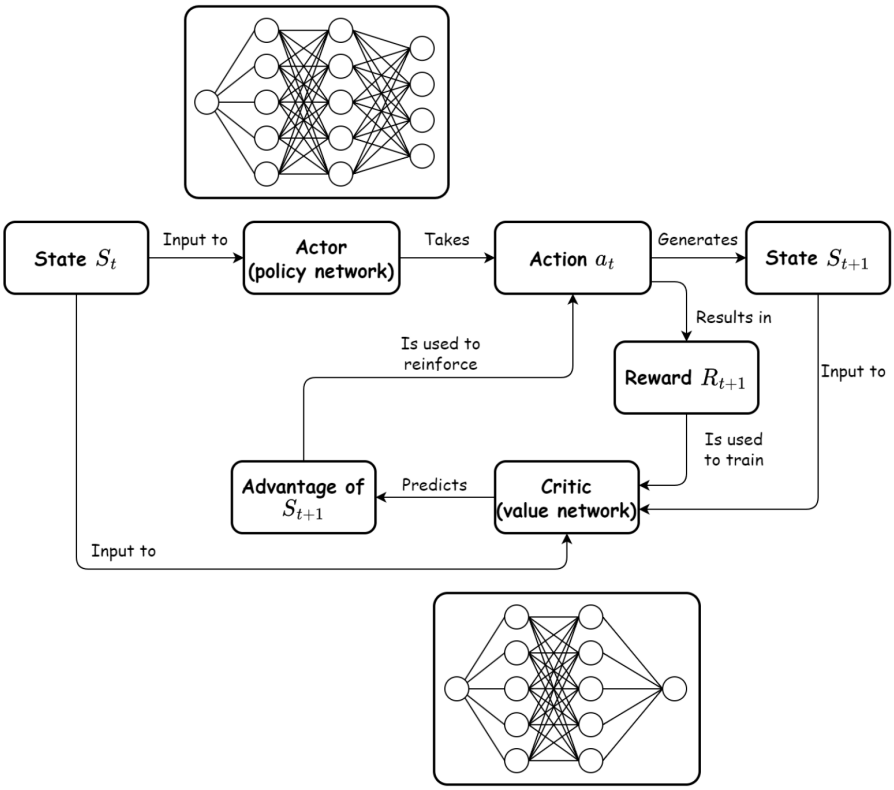

Figure 1: Model architecture

The agent's goal was to maximise its portfolio value by interacting with market data over multiple episodes. In each episode, it observed the current state, and decided the proportion of an asset to buy or sell, and received rewards based on changes in portfolio performance. Two primary losses were optimised during training:

1. The actor loss used the action's log probability under Gaussian policy to capture how well the action fitted the advantage.
2. The critic loss evaluated the value estimate accuracy, typically through calculating the mean square error within the expected and actual values.

Gradient descent was used by the agent to update its policy and value estimates as it continued to interact with the environment, progressively enhancing its trading strategy.

4.1 Algorithms

The four actor-critic algorithms used in this study, A2C, DDPG, TD3, and PPO, all have unique characteristics that should be considered when assessing their effects on portfolio optimisation. Most significantly, each strikes a different balance between stability and exploration. While DDPG and TD3 rely on deterministic policies, with TD3 adding stability improvements, A2C and PPO employ stochastic policies with advantage estimates.

Advantage Actor-Critic, A2C

A2C considered that actions were sampled from a stochastic policy, $a_t \sim \pi_\theta(a_t \mid s_t)$. The policy was updated through gradient ascent as $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$. The actor's objective function in A2C was:

$$J(\theta) = \mathbb{E}_{\pi_\theta} \left[\log \pi_\theta(a_t \mid s_t) \cdot AdvantageFunction \right]$$

The gradient of this objective was given by the policy gradient theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot AdvantageFunction \right]$$

In actuality, the critic was used to estimate the advantage function:

$$A^\pi(s_t, a_t) \approx R_t + \gamma \cdot V_\phi(s_{t+1}) - V_\phi(s_t)$$

Deep Deterministic Policy Gradient, DDPG

The deterministic policy assumed by DDPG was $a_t = \pi_\theta(s_t)$. The DDPG was updated $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$. The actor's objective function in DDPG was:

$$J_{\text{actor}}(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_a Q(s_t, a_t) \mid_{a_t = \pi_\theta(s_t)} \right]$$

The critic's objective was to minimise the temporal difference error. The target y_t is shown below and the critic's loss function respectively:

$$y_t = r_t + \gamma Q'(s_{t+1}, \pi(s_{t+1}))$$

$$J_{\text{critic}}(\phi) = \mathbb{E} \left[\left(Q(s_t, a_t) - y_t \right)^2 \right]$$

Twin Delayed Deep Deterministic Policy Gradient, TD3

The actor's objective function in TD3 was to maximise the action-value $Q(s_t, a_t)$ predicted by the critic:

$$J_{\text{actor}}(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_a Q(s_t, a_t) \mid_{a_t = \pi_\theta(s_t)} \right]$$

The critic's loss function was based on minimising the TD error using double Q-learning and target smoothing. The target y_t is shown below as and the critic's loss function respectively:

$$y_t = r_t + \gamma \min_{i=1,2} Q'(s_{t+1}, \pi'(s_{t+1})) - \epsilon \cdot \mathcal{N}$$

$$J_{\text{critic}}(\phi_1, \phi_2) = \mathbb{E} \left[\left(Q(s_t, a_t; \phi_1) - y_t \right)^2 + \left(Q(s_t, a_t; \phi_2) - y_t \right)^2 \right]$$

Poximal Policy Optimisation, PPO

The actor's objective function in PPO was:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\min\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \cdot A_t,\ \text{clip}\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, 1-\epsilon, 1+\epsilon\right) \cdot A_t\right)\right]$$

The gradient of the objective was:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta\left(\min\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \cdot A_t,\ \text{clip}\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, 1-\epsilon, 1+\epsilon\right) \cdot A_t\right)\right)\right]$$

The advantage estimate A_t was computed using Generalised Advantage Estimation (GAE), where δ_t is the TD error:

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \dots$$

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

4.2 Portfolio Environment

State Space

In the trading environment, the agent observed a structured state at each time step, which captured both portfolio status and market conditions across multiple assets. The custom state space is represented as:

$$S_t = \left[[b_t, h_t], \left\{(C_t^i, SS_t^i, T_t^i) \mid i \in N\right\}\right]$$

where:

- N: Total number of the portfolio's tradable assets.
- b_t: Cash available.
- $h_t = \{h_t^i\}_{i \in N}$: Quantity of shares held per asset.
- C_t^i: Closing price of asset.
- $SS_t^i \in \{-1, 0, 1\}$: Sentiment signal for asset, where -1 denotes negative, 0 denotes neutral, and 1 denotes positive sentiment.
- T_t^i: A dimensional vector containing technical indicators for asset.

The state space was a concatenation of these components, which is flattened into a single vector representing the state at time t.

This representation allowed the agent to consider both individual asset information and broader market conditions when making decisions.

Technical indicators based on past price and volume data, were used in the state space, to identify trends, momentum, and potential reversal points in asset prices.

Sentiment scores were gathered in the following manner, firstly mentions of an asset were identified (e.g., name, ticker, or related keywords) in daily news headlines using rule-based keyword matching.

- Each headline was then analysed with FinBERT [], a sentiment analysis model tailored for financial text, to estimate the probability of it being positive, negative, or neutral.
- For each day, the average sentiment was calculated across all related headlines. If positive sentiment was stronger than negative, a score of +1 was assigned; if negative was stronger, -1 was assigned.

Action Space

The agent operated in a continuous action space, where the action at each time step corresponded to a vector of real values representing the portfolio allocation (or deallocation) for each asset:

$$a_t = [a_t^1, a_t^2, \ldots, a_t^N] \in \mathbb{R}^N$$

where N is the quantity of assets, while a_t^i stands for the action for asset.

4.3 Data Preparation

The dataset included daily historical price data (Open, High, Low, Close, Volume) for 11 individual stocks, each selected to represent one of the 10 major sectors of the S&P 500. These stocks and their corresponding ticker symbols are listed below, along with their industry classifications:

- **AAPL** – Apple Inc. (*Information Technology*): A leading technology company known for consumer electronics and software products.
- **NVDA** – NVIDIA Corporation (*Information Technology*): A major producer of GPUs and AI-focused computing hardware and software.
- **JNJ** – Johnson & Johnson (*Health Care*): A multinational corporation operating in pharmaceuticals, medical devices, and consumer health.
- **JPM** – JPMorgan Chase & Co. (*Financials*): One of the largest global financial institutions offering banking and investment services.
- **AMZN** – Amazon.com, Inc. (*Consumer Discretionary*): A dominant e-commerce and cloud services company.
- **T** – AT&T Inc. (*Communication Services*): A major telecommunications provider offering wireless, broadband, and media services.
- **BA** – The Boeing Company (*Industrials*): A global leader in aerospace manufacturing and defense technology.
- **PG** – Procter & Gamble Co. (*Consumer Staples*): A multinational producer of household, personal care, and hygiene products.
- **XOM** – Exxon Mobil Corporation (*Energy*): One of the world's largest publicly traded oil and gas companies.
- **NEE** – NextEra Energy, Inc. (*Utilities*): A leading clean energy company and electric utility provider.

- **SPG** – Simon Property Group, Inc. (*Real Estate*): A real estate investment trust (REIT) focused on shopping malls and retail properties.

The financial data for these tickers was obtained using the Yahoo Finance API, providing a diverse and sector-representative dataset for portfolio analysis and reinforcement learning applications.

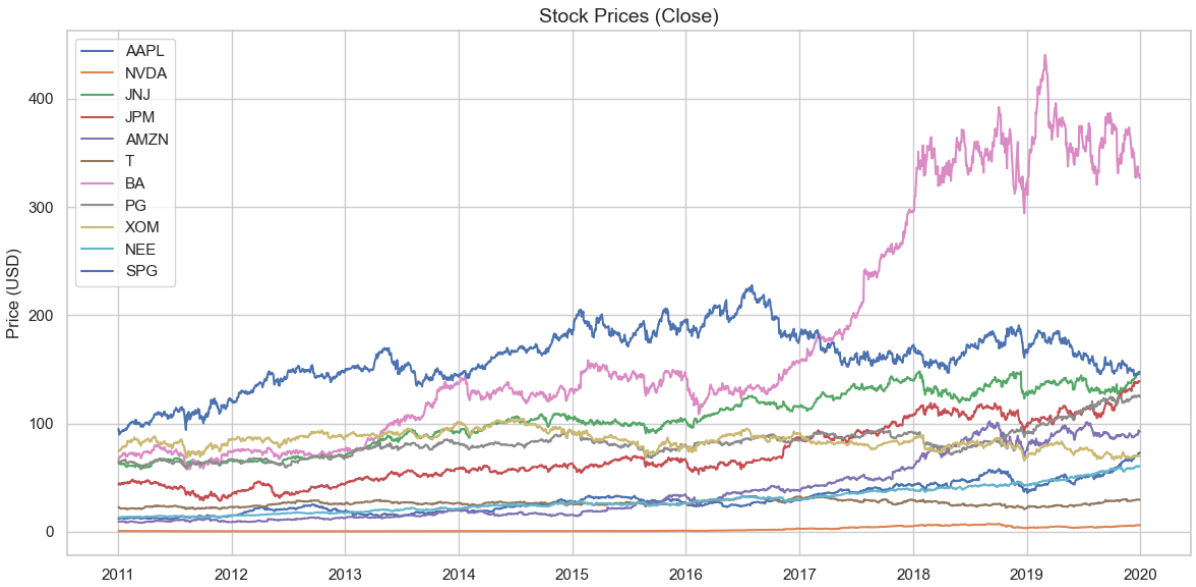

Figure 2: Historical closing price data for the 11 selected stocks from the beginning of 2011 to the end of 2019.

The OHLCV data was used to compute a suite of seven technical indicators for each asset:

- Simple Moving Average (SMA_20)
- Exponential Moving Average (EMA_20)
- Moving Average Convergence Divergence (MACD)
- Accumulation/Distribution Line (A/D)
- Relative Strength Index (RSI_14)
- Average Directional Index (ADX_14)
- Psychological Line Index (PSI_14)

These technical indicators were normalised and, along with the historical close price data, formed the feature set for the RL agent's observation space.

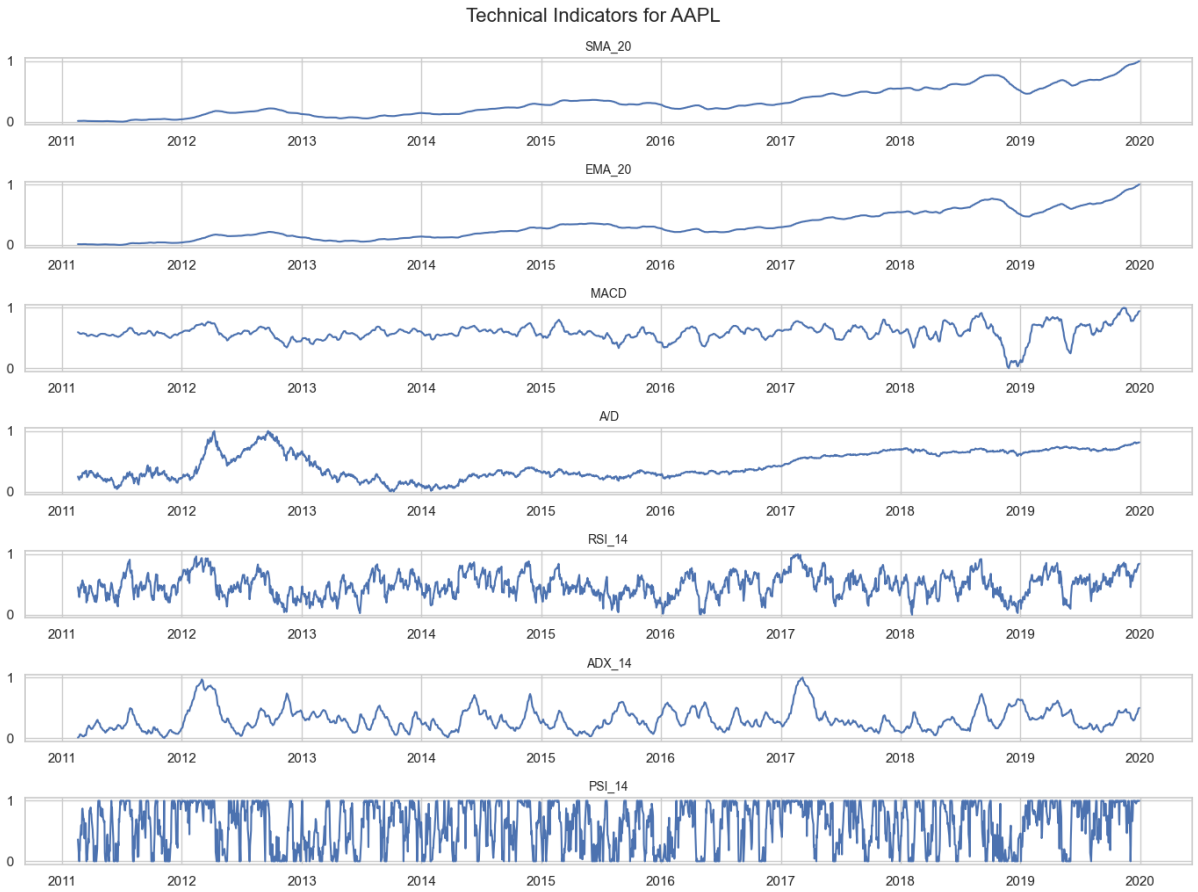

Figure 3: Technical indicators calculated for AAPL, representative of the feature engineering process applied to all assets in the portfolio.

The FNSPID Financial News Dataset [19] was first taken into consideration for use in this study due to its extensive use in scholarly research and its accessibility via Hugging Face, the open-source AI community that also created FinBERT, a financial domain specific variant of Google's BERT model that has been refined on a sizeable corpus of financial texts. But the FNSPID dataset was too big for the computers that could handle it. Alternatively, Benzinga was the source of a lighter-weight financial news dataset that Hugging Face supplied. This dataset, which covered financial headline news for all chosen stock tickers from the start of 2011 to the end of 2019, was used to generate sentiment scores using FinBERT. The sentiment scores of the headlines, which were categorised as -1 (negative), 0 (neutral), or 1 (positive), were averaged to create a single daily sentiment score when there were several headlines for a ticker in a single day. To guarantee compatibility with the model, which does not accept null values, a neutral score of 0 was applied on days when no headlines were available. An example of how sentiment scores changed over time for NVDA and JNJ, two of the chosen tickers, is shown in the figure below.

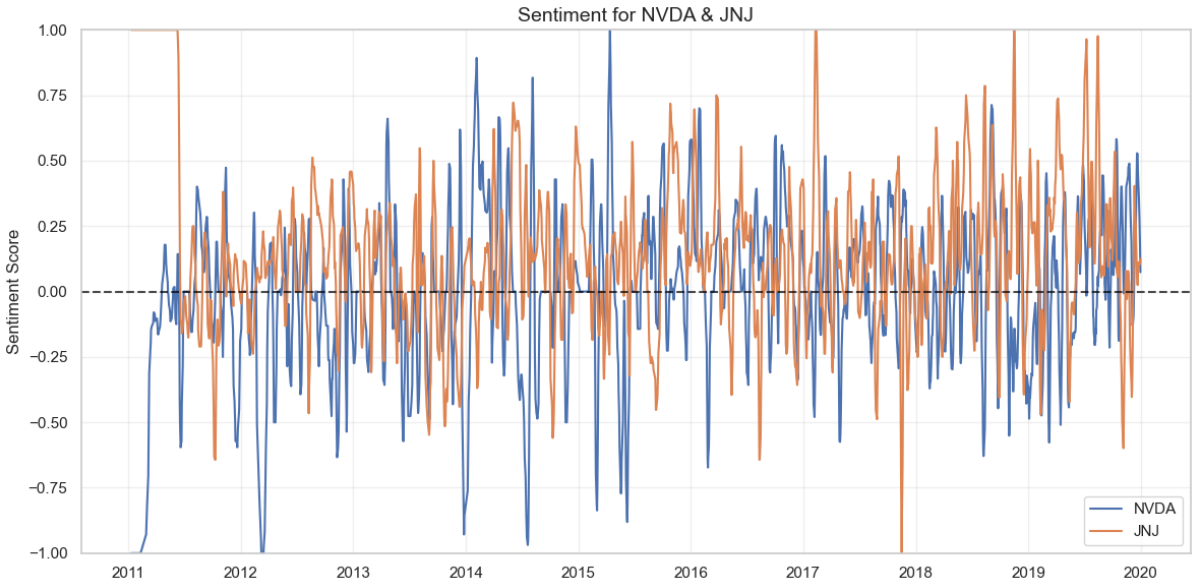

Figure 4: Example of the evolution of sentiment scores over time for two selected tickers, NVDA and JNJ. The plot illustrates how market sentiment fluctuated for these companies during the analysed period..

The first 80% of the data (from early 2011 to mid-2018) was used for training, while the remaining 20% (from mid-2018 to the end of 2019) was set aside for testing and backtesting. This initial 80/20 train-test split was used. This configuration enabled the agents to be assessed on more recent, unseen data while learning from a significant historical period. The model is exposed to a wider range of market behaviours during training on a longer historical dataset, which aids in the development of a more thorough comprehension of financial patterns. On the other hand, utilising insufficient data may result in overfitting to transient patterns and diminish the model's efficacy in various market conditions.

A time-series-specific validation method called Rolling Window Cross-Validation was also used to more accurately evaluate model performance over various time periods. In order to generate multiple train-test splits, this technique entails testing the model on the near future and training it on a fixed-sized window of historical data. The window is then moved forward by a fixed step size. In this study, the window advanced by 10% at each step, with 60% of the data used for training and 20% for testing in each fold. By preventing data leakage and maintaining temporal structure, this method makes sure that every fold replicates a real-world situation in which the model is trained on past data and assessed on future data. Rolling Window Cross-Validation provides a more robust and generalisable evaluation of model performance, especially in financial applications where market conditions are sequential and constantly changing.

4.4 Evaluation Metrics

The performance of each RL agent or model was assessed using the financial metrics listed below, which were then compared. A2C, the baseline agent, was compared across three observation spaces: (1) historical price data only, (2) historical price data combined with indicators, and (3) historical price data, indicators and sentiment scores

data. With all the data included in the observation space, a second comparison between the RL agents was conducted. In order to identify the robust RL agent, all models were evaluated using cross-validation.

- **Cumulative Return:** Total return over the entire period.

$$R_{\text{cumulative}} = \prod_{i=1}^{n}(1 + R_i) - 1$$

- **Annualised Return:** Average return per year.

$$R_{\text{annualised}} = (1 + R_{\text{cumulative}})^{\frac{1}{n}} - 1$$

- **Annualised Volatility:** Standard deviation of returns scaled to a yearly basis.

$$\sigma_{\text{annual}} = \sigma_{\text{periodic}} \times \sqrt{T}$$

- **Sharpe Ratio:** Risk-adjusted return, comparing the portfolio return to the risk-free rate.

$$S = \frac{R_p - R_f}{\sigma_p}$$

- **Maximum Drawdown (MDD):** Maximum observed loss from a peak to a trough before a new peak.

$$\text{MDD} = \max_{t \in [0,T]} \left(\frac{V_{\text{peak}} - V_t}{V_{\text{peak}}} \right)$$

5 Results

The RL agents were trained to actively modify asset allocations in an 11-stock portfolio in order to outperform passive buy-and-hold benchmark strategies such as an equal-weight portfolio and the S&P 500. Two major experiments were carried out, the first examined the effect of different input data types (Price Data, Price + Technical Indicators, Price + Technical Indicators + Sentiment Scores) on agent performance, using A2C as the baseline. The second involved managing a $10,000 portfolio over 750 episodes, comparing the performance of the RL agents to three passive investment benchmarks.

5.1 Effect of State Representation on Agent Performance

This experiment aimed to evaluate how different types of input data affect the performance of a RL agent, over 500 episodes. The A2C algorithm was used as the baseline model. Three configurations of the state observation space were tested:

1. Price Data Only
2. Price Data + Technical Indicators
3. Price Data + Technical Indicators + Sentiment Scores

All other training conditions remained constant. Performance was assessed using cumulative return, annualised return, annualised volatility, Sharpe ratio, and maximum drawdown.

Table 1: Performance of A2C Agent under Different Observation Spaces

Observation Space	Cumulative Return	Annualised Return	Annualised Volatility	Sharpe Ratio	Max Drawdown
Price Data Only	0.2409	0.1282	0.1480	0.8892	0.2179
Price + Indicators	0.2609	0.1406	0.1481	0.9627	0.2192
Price + Indicators + Sentiment	0.2214	0.1414	0.1438	0.9923	0.1940

Technical indicators had little effect on volatility or drawdown, but they did improve returns and raise the Sharpe ratio slightly when added to price data. The addition of sentiment scores further enhanced risk-adjusted performance, lowering maximum drawdown and volatility while increasing the Sharpe ratio to 0.9923, indicating more cautious and stable trading behaviour.

5.2 Comparative Performance of RL Agents

The RL agents used a specially designed trading environment to continuously reallocate asset weights based on technical indicators and past prices in order to manage a $10,000 portfolio. With rewards determined by portfolio log returns, the environment included a continuous observation and action space. Each of the four reinforcement learning algorithms (A2C, DDPG, TD3, and PPO) was trained for 750 episodes using data from 2011 to mid-2018. The algorithms' performance was then assessed using unseen data starting from mid-2018.

The performance of the RL agents was evaluated against three passive investment benchmarks: buying and holding the S&P 500 ETF, buying and holding the NASDAQ-100 ETF, and a static, equal-weighted portfolio of the same 11 assets.

- **Portfolio Growth:** As illustrated in Figure 5, all RL agents outperformed the passive benchmarks over the testing period. The A2C agent achieved the highest cumulative return, followed by TD3 and PPO.
- **Risk-Adjusted Returns:** A2C also delivered the best Sharpe ratio, indicating the most favourable return relative to volatility, TD3 and DDPG followed, as shown in Table **??**. Despite its lower return, PPO demonstrated a more defensive profile, showing the highest volatility and maximum drawdown, but consistent gains.
- **Asset Allocation:** Each RL agent developed distinct allocation preferences. A2C, for example, heavily favoured NEE, JNJ, and AAPL, suggesting a bias toward stable, large-cap stocks across utilities, healthcare, and tech sectors, indicating a diversified and risk-aware strategy.

Table 2: Performance Metrics of RL Agents on Testing Data

Agent	Cumulative Return	Annualised Return	Annualised Volatility	Sharpe Ratio	Max Drawdown
A2C	0.2282	0.1456	0.1437	1.0183	0.1917
DDPG	0.1808	0.1162	0.1589	0.7718	0.2190
TD3	0.2026	0.1298	0.1569	0.8568	0.2150
PPO	0.1943	0.1246	0.1789	0.7462	0.2642

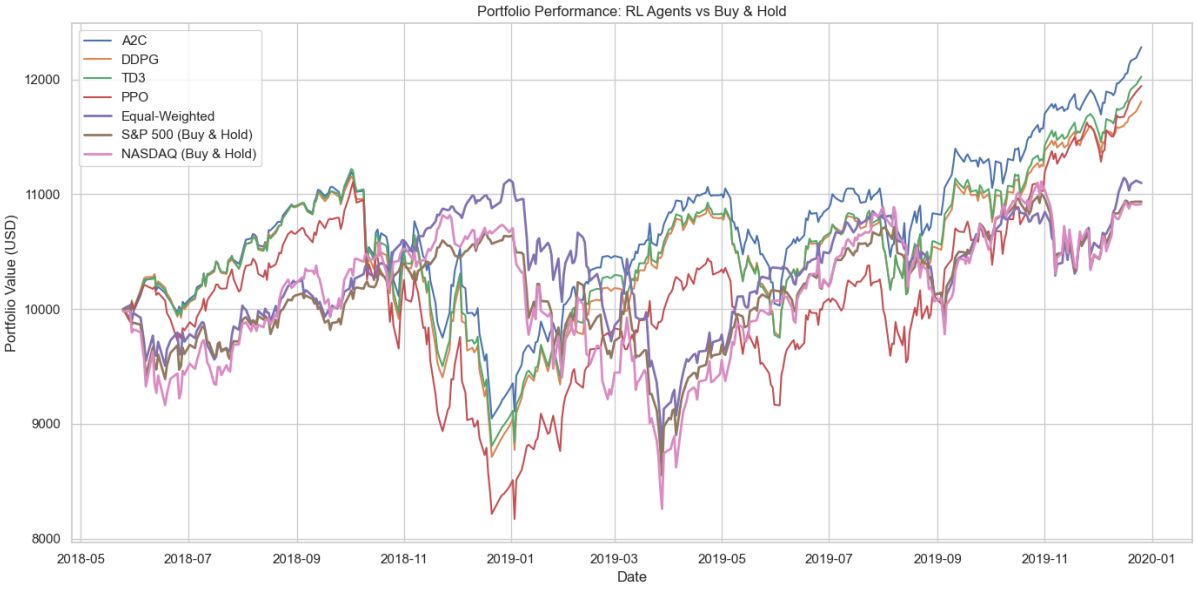

Figure 5: Portfolio value over time for RL agents versus equal-weighted, S&P 500, and NASDAQ buy-and-hold portfolios.

The A2C agent showed the most stable and effective allocation strategy, while DDPG showed the most unstable. Because of its volatility in asset distribution, DDPG's allocation strategy changed dramatically over time. On the other hand, over the course of the period, A2C maintained a more steady and efficient allocation. Below are the specifics of the DDPG allocation over time.

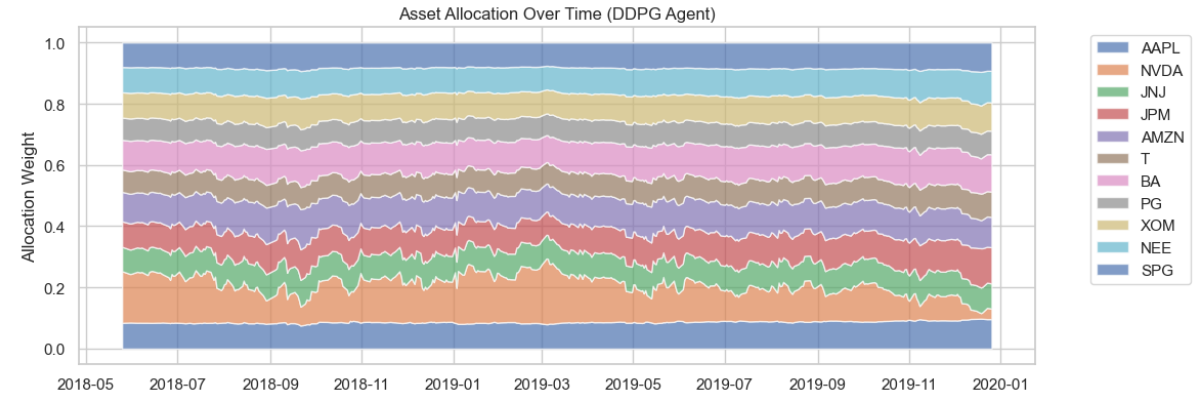

Figure 6: The asset allocation over time for the DDPG agent. The figure shows how the agent adjusts asset weights throughout the investment horizon.

Cross-validation and the performance of the RL agents over 150 episodes revealed that PPO produced the highest annualised and cumulative returns, albeit with the highest drawdown. DDPG had a little higher volatility but showed good returns. Although TD3 performed similarly to A2C but with slightly higher volatility, A2C had the lowest drawdown and the best risk-adjusted performance.

Table 3: Average Test Performance of RL Agents Across Cross-Validation.

Agent	Cumulative Return	Annualised Return	Annualised Volatility	Sharpe Ratio	Max Drawdown
A2C	0.0627	0.2236	0.1313	1.7390	0.0678
DDPG	0.0716	0.2679	0.1458	1.8064	0.0695
TD3	0.0634	0.2255	0.1358	1.6817	0.0724
PPO	**0.0784**	0.2851	0.1576	1.7137	0.0796

6 Discussion

This study shows that RL agents, especially A2C, have the ability to outperform conventional passive investment strategies. The RL agents successfully optimised portfolio allocations in a volatile, non-stationary market by combining sentiment analysis, technical indicators, and historical price data. With higher returns and a lower maximum drawdown, A2C in particular produced the best risk-adjusted performance, demonstrating its stability and appropriateness for investors who are risk averse.

Cross-validation revealed that PPO, while delivering the highest cumulative and annualised returns, also had the highest drawdown, indicating a more aggressive strategy that may be less suitable for risk-averse investors. A2C, on the other hand, demonstrated more consistent performance, with a solid Sharpe ratio and lower volatility, making it ideal for those seeking a more stable investment approach. DDPG and TD3, while yielding decent returns, exhibited higher volatility and less stable asset allocation strategies, which limited their effectiveness compared to A2C.

Despite their limitations, the study's results are encouraging. Because transaction costs, slippage, and liquidity constraints are not present, the models' applicability in the real world may be overestimated. Future research should incorporate these elements to more accurately model real-world market conditions. The high computational cost needed for training and cross-validation is still a practical issue, even though RL agents performed better in back-testing than passive strategies.

By including more data sources, such as macroeconomic indicators, and investigating more effective algorithms to lower computational overhead, future research could improve these models even more. While more work is required to make these strategies more useful in the real world, the combination of sentiment analysis, technical indicators, and RL shows great promise for enhancing portfolio optimisation overall.

7 Conclusion

This study demonstrates how RL agents can outperform conventional investment strategies and optimise portfolio allocation by adjusting to market conditions. The findings show that while careful algorithm selection is essential for striking a balance between risk and returns, the addition of technical indicators and sentiment analysis can enhance an agent's risk-adjusted performance. Nevertheless, the study also identifies a number of areas that require improvement, such as addressing the computational complexity associated with training these models and integrating transaction costs and more varied data. Future studies can get closer to creating RL-based portfolio man-

agers that are both incredibly efficient and useful in the real world by tackling these constraints.

REFERENCES

[1] R. Kissell, "Chapter 10 - Portfolio Construction," in *The Science of Algorithmic Trading and Portfolio Management*, R. Kissell, Ed. San Diego: Academic Press, 2014, pp. 331–360.

[2] H. Markowitz, "Portfolio Selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[3] W. F. Sharpe, "Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk," *The Journal of Finance*, vol. 19, no. 3, pp. 425–442, 1964.

[4] F. Espiga-Fernández, Á. García-Sánchez, and J. Ordieres-Meré, "A Systematic Approach to Portfolio Optimization: A Comparative Study of Reinforcement Learning Agents, Market Signals, and Investment Horizons," *Algorithms*, vol. 17, no. 12, p. 570, 2024.

[5] H. Dong, Z. Ding, and S. Zhang, *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Springer, 2020, ch. 2–6, pp. 47–245.

[6] A. Rao and T. Jelvis, *Foundations of Reinforcement Learning with Applications in Finance*. Chapman and Hall/CRC, 2022, ch. 3, pp. 41–72.

[7] T. Kabbani and E. Duman, "Deep reinforcement learning approach for trading automation in the stock market," *IEEE Access*, vol. 10, pp. 93 564–93 574, 2022.

[8] M. Kong and J. So, "Empirical analysis of automated stock trading using deep reinforcement learning," *Applied Sciences*, vol. 13, no. 1, p. 633, 2023.

[9] F. Espiga-Fernández, Á. García-Sánchez, and J. Ordieres-Meré, "A systematic approach to portfolio optimization: A comparative study of reinforcement learning agents, market signals, and investment horizons," *Algorithms*, vol. 17, no. 12, p. 570, 2024.

[10] M. Rezaei and H. Nezamabadi-Pour, "A taxonomy of literature reviews and experimental study of deep reinforcement learning in portfolio management," *Artificial Intelligence Review*, vol. 58, no. 3, pp. 1–46, 2025.

[11] Z. Hao, H. Zhang, and Y. Zhang, "Stock portfolio management by using fuzzy ensemble deep reinforcement learning algorithm," *Journal of Risk and Financial Management*, vol. 16, no. 3, p. 201, 2023.

[12] Y. Jiang, J. Olmo, and M. Atwi, "Deep reinforcement learning for portfolio selection," *Global Finance Journal*, vol. 62, p. 101016, 2024.

[13] W. Zucchini and I. L. MacDonald, *Hidden Markov Models for Time Series: An Introduction Using R*. Chapman and Hall/CRC, 2009, ch. 2, pp. 29–46.

[14] N. Bäuerle and U. Rieder, *Markov Decision Processes with Applications to Finance*. Springer Science & Business Media, 2011, ch. 5 & 6, pp. 147–189.

[15] K. Balakrishnan, *TensorFlow Reinforcement Learning Quick Start Guide: Get up and running with training and deploying intelligent, self-learning agents using Python*. Packt Publishing Ltd, 2019, ch. 1, pp. 16–17.

[16] O. C. Ibe, "Chapter 12 - Special Random Processes," in *Fundamentals of Applied Probability and Random Processes (Second Edition)*, second edition ed., O. C. Ibe, Ed. Boston: Academic Press, 2014, pp. 369–425.

[17] S. Ahlawat, *Reinforcement Learning for Finance*. Springer, 2022, ch. 3, pp. 233–248.

[18] L. Tang, "An actor-critic-based portfolio investment method inspired by benefit-risk optimization," *Journal of Algorithms & Computational Technology*, vol. 12, no. 4, pp. 351–360, 2018.

[19] Z. Dong, X. Fan, and Z. Peng, "Fnspid: A comprehensive financial news dataset in time series," 2024.

Supplementary Materials

On GitHub, you can find the complete code used for this analysis, which includes data preprocessing, model training, and result visualisation: https://github.com/bonganishube/deep-reinforcement-learning-for-portfolio-optimisation.

Wits University Faculty of Science post-graduate student AI declaration

I understand that the use of generative AI tools (such as ChatGPT or similar) without explicitly declaring such use constitutes a form of plagiarism and is classified by Wits University as academic misconduct.

I declare that in the course of conducting the research towards my degree or in the preparation of this thesis/dissertation/research report (select one by marking with an X):

I **did not** make use of generative AI tools ☐

I **did** make use of generative AI tools for the following (tick all that apply):

1. Idea Generation (research problem/design, hypothesis) ☐
2. Sourcing Related Work (summarising, identifying sources) ☐
3. Methods and Experiment Design (experiment setup, model tuning) ☐
4. Data Analysis (presentation, coding, interpretation) ☒
5. Theoretical Development (theorem proving, conceptual analysis) ☐
6. Code Development (generating algorithms, writing scripts) ☒
7. Presentation (rendering graphics, formatting) ☒
8. Editing (grammar, readability) ☒
9. Writing (text generation, document structuring) ☐
10. Citation Formatting (structuring, organising) ☒

If other uses were involved, please specify below:

Generative AI tool used (list all)	**Used for?**

If generative AI tools were used as an integral part of the experimental design or in the direct execution of my research, I confirm that details of this use are clearly outlined in the relevant experimental/methodology chapters of my thesis/dissertation/research report.

Student number: 2112465

Candidate signature: 

Date: 21 September 2025