Optimization of Stock Trading Strategy with Reinforcement Learning

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

Submitted By

MOHAMMAD DANISH

2K21/SWE/15

Under the supervision of

PROF. RUCHIKA MALHOTRA

Head of Department

M.Tech (Software Engineering) Mohammad Danish 2023

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road

Delhi - 110042

MAY, 2023

Department of Software Engineering

Delhi Technological University

(Formerly Delhi College of Engineering)

Bawana Road Delhi-110042

CANDIDATE'S DECLARATION

I, hereby declare that the work presented in this thesis entitled "**Optimization of stock trading strategy with Reinforcement Learning**", in fulfilment of the requirement for the award of the MASTER OF TECHNOLOGY degree in Software Engineering submitted in Department of Software Engineering at DELHI TECHNOLOGICAL UNIVERSITY, New Delhi, is an authentic record of my own work carried out during my degree under the guidance of **Prof. Ruchika Malhotra (HOD, SWE-DTU).**

The work reported in this has not been submitted by me for the award of any other degree or Diploma.

Date: 29.05.2023

Place: New Delhi

Mohammad Danish

2K21/SWE/15

Danish

(Signature)

Department of Software Engineering

Delhi Technological University

(Formerly Delhi College of Engineering)

Bawana Road Delhi-110042

CERTIFICATE

This is to certify that **Mr. Mohammad Danish (2K21/SWE/15)** is a student at the Department of Software Engineering of Delhi Technological University formerly known as Delhi College of Engineering undergone a major project work of 4th semester titled "**Optimization of stock trading strategy with Reinforcement Learning".** To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Ruchika Malhotra

Date: 29.05.2023

Place: New Delhi

Prof. Ruchika Malhotra

HOD, SWE(DTU)

SUPERVISOR

(Signature)

Department of Software Engineering

Delhi Technological University

(Formerly Delhi College of Engineering)

Bawana Road Delhi-110042

ACKNOWLEDGMENT

I would like to express my special thanks to my mentor **Prof. Ruchika Malhotra** who supported me in the completion of this project. This project titled "**Optimization of stock trading strategy with Reinforcement Learning"** was a golden opportunity to do the research and enhance my skill set. It was her enigmatic supervision, unwavering support and expert guidance which has allowed me to complete this work in due time. I humbly take this opportunity to express my deepest gratitude to her.

Lastly, I would also like to thank my parents for their lovely support for everything.

Date: 29.05.2023

Place: New Delhi

Mohammad Danish

2K21/SWE/15

Danish

(Signature)

ABSTRACT

This project aims to optimize stock trading strategies by employing reinforcement learning (RL) techniques. The primary goal is to create a trading strategy capable of learning from historical data and executing profitable trades in real-time. The proposed method involves utilizing a deep neural network as a function approximator to learn a trading policy, which is subsequently fine-tuned using RL techniques. The neural network takes a set of technical indicators as input and generates buy, hold, or sell signals for each stock.

To assess the performance of the RL-based trading strategy, a dataset comprising historical stock prices from the NIFTY and SENSEX indices is utilized. The performance of this approach is compared to several conventional trading strategies such as buy-and-hold and moving average crossover. The results indicate that the RL-based trading strategy surpasses these traditional strategies in terms of profitability and risk-adjusted returns.

Additionally, sensitivity analysis is conducted to evaluate how different hyper-parameters impact the trading strategy's performance. Specifically, variations in the discount factor, learning rate, and exploration rate of the RL algorithm are examined, and their effects on the trading strategy's performance are analysed. The results demonstrate that the RL-based approach is robust against changes in hyper-parameters and consistently outperforms traditional strategies.

Furthermore, a back-testing analysis is performed to assess the RL-based trading strategy's performance on out-of-sample data. A rolling window approach is implemented to simulate real-time trading, and the trading strategy's performance is evaluated over time. The results consistently show that the RL-based trading strategy outperforms traditional strategies and achieves higher risk-adjusted returns.

Overall, the findings suggest that RL-based approaches have significant potential for enhancing the performance of stock trading strategies. The proposed method can be applied to various financial assets and can facilitate the development of automated trading systems capable of executing profitable trades in real-time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS

RL	Reinforcement Learning
OHLCV	Open, High, Low, Close, Volume
DL	Deep Learning
RSI	Relative Strength Index
MACD	Moving Average Convergence Divergence
NSE	National Stock Exchange
BSE	Bombay Stock Exchange
MP	Markov Process
MDP	Markov Decision Process
PPO	Proximal Policy Optimization
A2C	Advance Actor Critic
DQN	Deep Q-Learning
SENEX	Index of BSE
NIFTY	Index of NSE
DDPG	Deep Deterministic Policy Gradient
RNN	Recurrent Neural Network
TD	Temporal Difference
SARSA	State Action Reward State Action
TRPO	Trust Region Policy Optimization
A3C	Asynchronous Advantage Actor-Critic
TD3	Twin Delayed Deep Deterministic Policy Gradient

CHAPTER 1 – INTRODUCTION

The stock market presents a highly intricate and ever-evolving environment, making it a formidable challenge to predict its future movements. Even the most experienced traders are prone to errors in this complex domain. However, a solution that shows great promise in enhancing stock market trading is reinforcement learning.

Reinforcement learning, a subset of machine learning, enables agents to learn optimal behaviours through trial and error. In this approach, agents receive rewards for actions that yield positive outcomes and are penalized for actions leading to negative results. Over time, agents learn to take actions that maximize their rewards. In the context of stock market trading, reinforcement learning can be employed to train agents in the art of buying and selling stocks to maximize profits. By analysing historical market data, these agents learn patterns and trends that facilitate predictions of future market movements.

The primary objective of this thesis is to explore the application of reinforcement learning for optimizing stock market trading. I will develop a reinforcement learning agent capable of learning the dynamics of buying and selling stocks to maximize profits. To evaluate its performance, I will subject the agent to historical data of NIFTY and SENSEX and examine its ability to adapt to diverse market conditions. Furthermore, I will discuss the implications of my findings for the future of stock market trading.

In summary, the stock market poses challenges due to its complex and unpredictable nature. However, reinforcement learning offers a promising avenue for improving stock market trading. By training agents on historical data and leveraging their ability to learn from past experiences, I can enhance decision-making processes and potentially revolutionize the industry. Through the development and evaluation of a reinforcement learning agent, this thesis aims to contribute valuable insights to the optimization of stock market trading strategies.

1.1 Reinforcement Learning (RL)

Reinforcement Learning (RL) is a subfield of machine learning that concentrates on enabling an agent to learn optimal decision-making strategies, maximizing cumulative rewards in a given environment. It operates on the principle of trial and error, where the agent takes actions and receives feedback in the form of rewards or penalties from the environment. Through this iterative process, the agent learns to associate specific actions with corresponding rewards and adjusts its behaviour accordingly.

A reinforcement learning system comprises five essential components:

Agent: The agent is the learning entity responsible for making decisions within the environment. It selects actions based on the current state of the environment and its acquired knowledge from past experiences.

Environment: The environment represents the external world in which the agent operates. It provides feedback to the agent through rewards or penalties, depending on the actions taken.

Reward signal: The reward signal is a numerical feedback that the agent receives from the environment after taking an action. The agent's objective is to maximize the cumulative reward signal over time, guiding its decision-making process.

State: The state encapsulates the current condition or snapshot of the environment at a specific time. It equips the agent with relevant information required for decision-making and action selection. The state includes factors such as the agent's position, surroundings, and other pertinent details. It can be represented in various forms, such as numerical vectors, images, or structured representations.

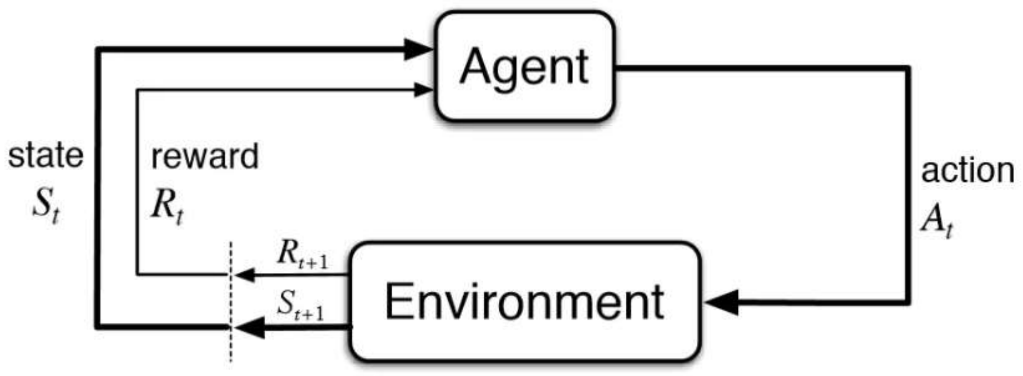


Figure 1: Basic Re-information Learning Setup [30].

Action: An action represents the decision or choice made by the RL agent based on the current state. The agent selects actions from a set of available options within the environment. Actions can be discrete, involving choices from a predefined set, or continuous, where the agent selects a value within a range. The RL agent's goal is to learn the optimal actions that maximize long-term rewards or objectives.

Reinforcement learning has found successful applications in various domains, including game playing, robotics, and autonomous driving. However, it also presents challenges such as the exploration-exploitation trade-off, credit assignment, and the curse of dimensionality. These challenges require careful consideration and appropriate techniques to ensure effective RL implementation.

1.2 Reinforcement learning methodologies

Reinforcement learning encompasses various methodologies that can be applied to address different challenges in dynamic environments. Here are some common types of reinforcement learning methods:

Model-based RL: This approach involves the agent learning a model of the environment, including its dynamics and reward structure. The learned model is then used to make decisions. Model-based RL can be more efficient in terms of sample usage, but it requires accurate modelling of the environment.

Model-free RL: In contrast to model-based methods, model-free RL directly estimates the optimal policy or value function without explicitly learning a model of the environment. This approach is often easier to implement and more robust to modelling errors, but it may require more data to converge.

Value-based RL: Value-based RL focuses on estimating the optimal value function or action-value function. These functions assign a value to each state or state-action pair, and the agent selects actions based on the highest value or action-value. Q-learning is a well-known value-based RL algorithm.

Policy-based RL: Policy-based RL directly estimates the optimal policy, which maps each state to a probability distribution over actions. The agent selects actions based on the policy. Policy gradient methods are popular examples of policy-based RL algorithms.

Actor-critic RL: Actor-critic reinforcement learning (RL) is an approach that integrates both value-based and policy-based methods. It leverages an actor to learn the policy and a critic to learn the value function. The actor component is responsible for selecting actions based on the learned policy, while the critic component evaluates the policy by estimating the value function.

Multi-agent RL: Multi-agent RL involves multiple agents learning to interact with each other in a shared environment. Each agent learns its own policy or value function, which can be influenced by the actions of other agents. This approach is useful for modelling complex social dynamics like cooperation and competition.

Reinforcement learning has been applied to stock trading with the aim of learning trading policies that maximize profits over time. It finds application in areas such as portfolio management, trading strategies, market making, and algorithmic trading.

Applying RL to stock trading comes with challenges due to the high volatility and complexity of financial markets, the need for fast and accurate data processing, and the risk of overfitting to historical data. However, promising results have been observed in some studies, where RL-based trading strategies outperformed traditional approaches.

In addition to reinforcement learning, algorithmic trading strategies play a crucial role in stock market operations. Some common strategies include technical analysis, which uses historical price data to identify trends and patterns, and fundamental analysis, which considers economic data and company financials to assess the value of securities. High-frequency trading involves executing trades rapidly to exploit small price movements in the market.

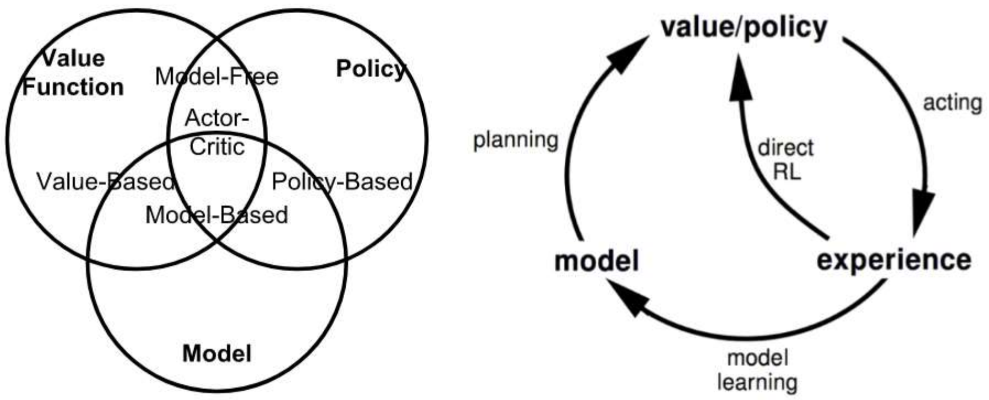


Figure 2: Reinforcement Learning categorization and classification [9].

1.3 Application of RL in stock market

When it comes to the application of reinforcement learning (RL) in the stock market, there are several areas where RL techniques can be utilized effectively:

Portfolio Management: RL can be employed to learn optimal portfolio management strategies that strike a balance between risk and return. The RL agent learns to select a portfolio of assets and dynamically adjust it over time based on market conditions and risk preferences.

Market Making: RL can be applied to learn optimal pricing strategies for market making. Market making involves buying and selling stocks to provide liquidity to the market. The RL agent learns to set bid and ask prices based on market conditions and order flow dynamics.

Trading Strategies: RL can be utilized to learn trading strategies that exploit market inefficiencies or predict market trends. By analysing market indicators or other signals, the RL agent learns when to buy or sell stocks with the goal of maximizing profits.

Algorithmic Trading: RL can be leveraged to develop algorithms for automated trading based on predefined rules or market signals. The RL agent learns to place orders based on real-time market conditions and order book dynamics [29].

High-Frequency Trading: This strategy involves executing trades rapidly, often within milliseconds, to exploit small price movements in the market. Algorithms are employed to identify these price movements and facilitate quick trading decisions.

Technical Analysis: This approach employs historical price data to identify trends and patterns. Algorithms can be used to analyse this data and generate buy and sell signals accordingly.

Fundamental Analysis: Fundamental analysis involves assessing the value of a security based on economic data, company financials, and other factors. Algorithms can be utilized to analyse this data and generate buy and sell signals accordingly.

1.4 RL algorithms

RL algorithms offer a diverse range of approaches to solving reinforcement learning problems. Here are some notable RL algorithms along with their characteristics and applications:

Q-Learning: Q-learning is an algorithm that estimates the optimal action-value function without requiring a model of the environment. It can be implemented using a table or a neural network to store the Q-values for each state-action pair. By selecting actions with the highest expected reward, the agent learns to make optimal decisions. Q-learning has found extensive applications in domains such as robotics, game playing, and autonomous systems [7].

SARSA: SARSA is another model-free algorithm that estimates the optimal policy. Similar to Q-learning, it uses a table or a neural network to store the state-action values. The agent selects actions based on the policy and updates the value function based on the actual rewards received. SARSA has been successfully used in tasks such as robot navigation and control.

Deep Q-Networks (DQNs): DQNs [13, 19] are a variant of Q-learning that leverages deep neural networks to estimate the Q-function [33]. They have demonstrated superior performance in handling high-dimensional state spaces, making them well-suited for applications such as image recognition, natural language processing, and game playing.

Policy Gradient Methods: Policy gradient methods [8, 20] directly optimize the policy function, which maps states to actions. These methods employ gradient descent to update the policy parameters based on the expected rewards. Policy gradient methods have been extensively employed in robotics, control systems, and autonomous vehicles.

Actor-Critic Methods: Actor-critic [9][21] methods offer the benefits of both value-based and policy-based approaches by incorporating an actor and a critic. The actor component is responsible for learning the policy, while the critic component estimates the value function. This combination allows for efficient learning and decision-making in reinforcement learning tasks. This combination allows for more stable and efficient learning. Actor-critic methods have found applications in complex tasks like robotic manipulation and multi-agent systems.

Monte Carlo Methods: Monte Carlo methods [12] estimate the value function or policy by simulating trajectories and averaging the rewards over multiple episodes. They are particularly useful for problems with long-term dependencies and stochastic dynamics. Monte Carlo methods have been applied in fields such as finance, optimization, and recommendation systems.

Temporal Difference Learning: Temporal difference (TD) learning is a model-free approach that updates the value function based on the difference between the estimated value and the actual rewards received. TD learning is commonly used in combination with other techniques like Q-learning and SARSA to enhance learning efficiency and stability.

Deep Deterministic Policy Gradient (DDPG): DDPG is a policy gradient method that employs deep neural networks to estimate the policy function. It has been widely used in stock trading to learn optimal trading policies based on market data and indicators.

Recurrent Neural Networks (RNNs): RNNs are neural networks capable of processing sequential data, making them suitable for modelling time-series data in stock trading. RNNs can be combined with other RL techniques such as DQNs and policy gradient methods to capture temporal dependencies and patterns in the data.

Proximal Policy Optimization (PPO): PPO is a policy gradient [25] method that uses a clipped objective function to ensure stable policy updates. It has demonstrated improved sample efficiency and stability compared to other policy gradient methods. PPO has found applications in robotics, game playing, and optimization tasks.

Trust Region Policy Optimization (TRPO): TRPO [26] is another policy gradient method that incorporates a trust region constraint to limit policy updates. It ensures that policy updates do not deviate excessively from the current policy. TRPO is effective in continuous control tasks that require continuous actions.

Asynchronous Advantage Actor-Critic (A3C): A3C [27] is a parallelized implementation of the actor-critic method, where multiple agents learn in parallel and share a global value function. It has shown effectiveness in tasks with high-dimensional state spaces and continuous actions, and has been applied in areas such as robot control and game playing, where real-time decision-making and coordination are crucial.

Twin Delayed Deep Deterministic Policy Gradient (TD3): TD3[28] is an extension of DDPG that addresses the issue of overestimation in the Q-function. It incorporates a delayed critic update and twin critics to reduce overestimation bias. TD3 has demonstrated improved sample efficiency and stability compared to DDPG in continuous control tasks. It has been utilized in applications such as robot control, autonomous driving, and robotic manipulation.

These RL algorithms offer a diverse set of tools for tackling different types of reinforcement learning problems. They provide a framework for agents to learn from interactions with the environment and make optimal decisions to maximize rewards.

By leveraging techniques such as value function estimation, policy optimization, and deep neural networks, these algorithms have achieved remarkable successes across various domains. However, it is important to carefully choose and tailor the appropriate algorithm based on the specific characteristics of the problem at hand, considering factors such as the complexity of the environment, available data, and desired performance objectives.

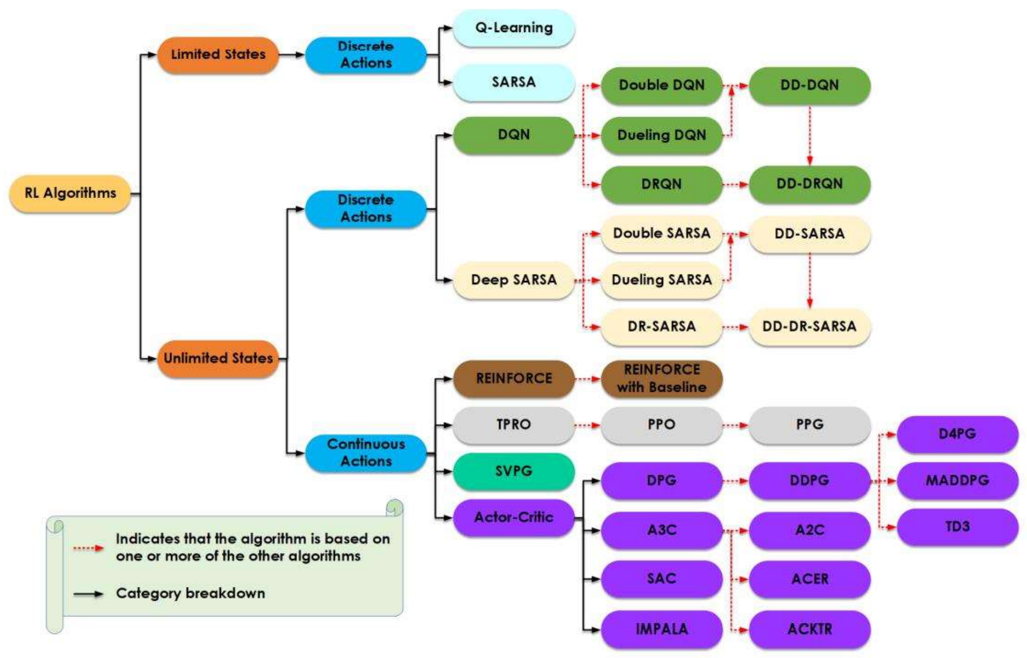


Figure 3: RL Algo Classification [31].

1.5 Financial Background

Stock Market: The stock market acts as a marketplace where individuals can engage in the buying and selling of shares from publicly traded companies. It facilitates companies in raising capital through the issuance of shares and allows investors to partake in the ownership and potential expansion of these companies. By providing a platform for companies to secure funds for their operations and growth, the stock market presents an avenue for investors to invest in these companies and potentially reap financial gains from their accomplishments.

Stock Exchange: Stock exchanges, such as the NSE & BSE, facilitate the buying and selling of stocks. These exchanges provide a centralized marketplace where buyers and sellers can come together to execute trades through agent called broker.

Shares: Companies divide their ownership into shares, which represent a portion of the company's ownership. Investors can purchase these shares, becoming shareholders and having a stake in the company's performance and profits.

Stock Prices: Stock prices fluctuate based on various factors, including supply and demand, company performance, economic conditions, and investor sentiment. Buyers and sellers determine stock prices through their transactions in the market. Example Rs. 900 is the stock price of SUNPHARMA at NSE exchange.

Stock Indexes: Stock market indexes, such as the NIFTY or SENSEX, track the performance of a selected group of stocks to provide an overall snapshot of the market. These indexes serve as benchmarks for evaluating market trends and comparing investment performance.

Investment Strategies: Investors employ various strategies when participating in the stock market. These strategies can range from long-term investing, where investors buy and hold stocks for an extended period, to short-term trading, where investors seek to profit from short-term price fluctuations.

Risk and Reward: The stock market offers the potential for both financial gains and losses. Stocks are considered higher-risk investments compared to more conservative options like bonds or savings accounts. Investors need to assess their risk tolerance and make informed decisions based on their financial goals and time horizon.

$$Risk{:}\ Reward\ Ratio\ = \frac{Expected\ Risk}{Expected\ Reward} \tag{1}$$

Market Volatility: Stock markets can experience periods of volatility, where prices can fluctuate significantly in response to economic, political, or global events. Volatility presents both risks and opportunities for investors.

$$\sigma = \sqrt{\left[\frac{\Sigma((r - \mu)^2)}{n}\right]} \tag{2}$$

Where:

σ is the standard deviation, r is the return of a security or index, μ is the mean return of a security or index and n is the number of observations.

Regulatory Framework: Stock markets operate under regulatory bodies that establish rules and regulations to ensure fair and transparent trading. These regulations aim to protect investors' interests, maintain market integrity, and promote efficient functioning of the market. In India regulations are governed by Securities and Exchange Board of India.

Market Participants: The stock market involves various participants, including individual investors, institutional investors (such as pension funds or mutual funds), traders, brokers, and market makers. Each participant plays a role in the buying and selling of stocks.

Portfolio: A portfolio is a collection of financial investments held by an individual or entity. It typically consists of a variety of assets such as stocks, bonds, mutual funds, real estate, or other investment vehicles. The purpose of a portfolio is to diversify investments and manage risk while aiming to achieve specific financial goals.

1.6 Stock Market Data and Representation

Trading Volume Data: Volume bars represent the trading volume within a specific time period. Each bar's height corresponds to the total volume traded during that time period. Volume bars help assess the level of market activity and can provide insights into buying and selling pressure.

Tick bars: are based on the number of trades or price changes rather than fixed time periods. Each bar represents a specific number of trades (e.g., 100 trades) or price changes. Tick bars can be useful for capturing market volatility and adjusting for periods of high or low activity.

Bid-Ask Data: Bid-ask data, also known as the order book, consists of the highest bid price, lowest ask price, and corresponding quantities of buyers and sellers in the market. This data provides insights into the current supply and demand dynamics and helps traders determine the prevailing market sentiment.

Time and Sales Data: Time and sales data, also known as the trade tape, displays each individual trade executed in the market, including the trade price, quantity, and timestamp. This data provides a detailed view of transaction activity and can help identify trends and patterns.

Level 2 Data: Level 2 data provides a deeper view of the order book by showing the bid and ask prices and quantities at multiple price levels beyond the best bid and ask. It allows traders to see the depth of the market and assess the liquidity and potential price movements.

Market Depth Data: Market depth data combines Level 2 data with additional information, such as market orders and order sizes, providing a comprehensive view of the supply and demand at different price levels. This data is particularly useful for analysing market liquidity and making informed trading decisions.

Fundamental Data: Fundamental data includes information about a company's financial health, such as earnings reports, balance sheets, income statements, and other relevant financial metrics. Fundamental data is essential for fundamental analysis, which focuses on evaluating the intrinsic value of a stock.

Data Types	Data	Usages	Source
Fundamental Data	Assets, Sales, Liabilities, Earnings	Used in balance sheet to represent the status of company	Exchange, Broker
Technical Data or Historical Data	Price, Volume, Volatility, OI Bids-Ask Spread	Used in secondary market in trading and investment analysis, future and options contract.	Exchange, Broker, Yahoo Finance, Google Finance
Sentiment Data [6]	News articles, Tweets	Use to analyse the how the market participants are feeling about the particular stock or market as a whole.	News article like MoneyControl, Economics Times, Twitter

Table 1: Data in finance and its types.

News and Sentiment Data: News and sentiment data encompass news articles, social media posts, and other sources of information that reflect market sentiment and potential impact on stock prices. Traders and investors analyse this data to gauge market sentiment and make informed decisions.

Historical Data: Historical data includes past price and volume information for a given security. It is used to analyse patterns, trends, and correlations over time and to develop trading strategies based on historical performance.

Real-Time Data: Real-time data provides up-to-the-minute information on price changes, trading volume, and other market indicators. Traders rely on real-time data to make timely decisions and execute trades based on the most current market conditions.

These different types of data are crucial for market analysis, trading strategies, and decision-making in the stock market. By utilizing and analysing these data points effectively, traders and investors can gain insights and improve their chances of success in the dynamic and ever-changing stock market environment.

1.7 Representations:

OHLC Bars: OHLC (Open, High, Low, Close) bars are a standard representation of price movements. Each bar includes four price points: the opening price, highest price reached, lowest price reached, and closing price. OHLC bars provide a concise view of price range and trend within a given time period.

Candlestick Bars: Candlestick bars, or Japanese candlestick charts, offer a comprehensive depiction of price fluctuations. These charts present information about the price range (open, high, low, close) within a particular time frame. Each candlestick on the chart portrays whether the closing price exceeded or fell below the opening price, indicated by a filled or hollow body, respectively. Additionally, the upper and lower shadows of the candlestick reflect the highest and lowest prices reached during the given time period. Candlestick charts have gained popularity in technical analysis as they enable the identification of price patterns and trends.

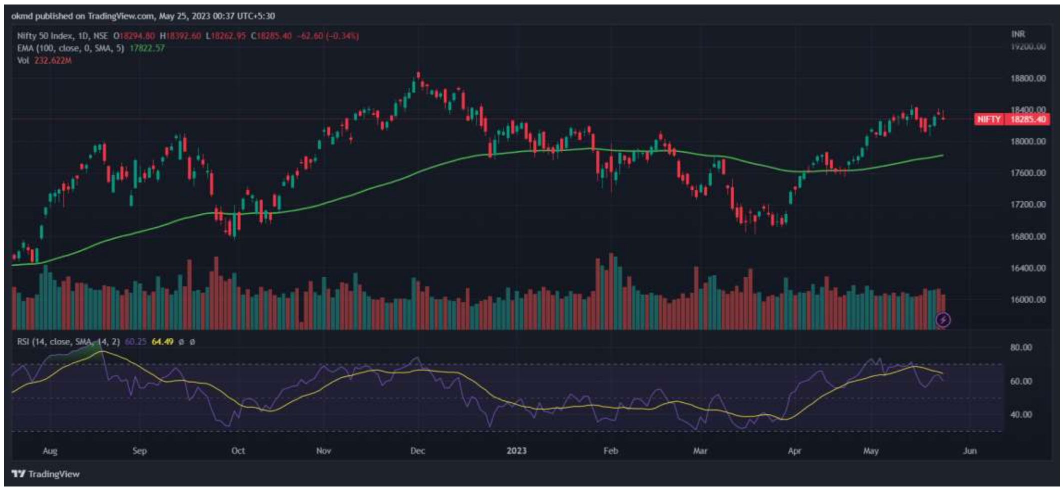


Figure 4: OHLCV of Trading View and Technical Indicator RSI and EMA.

1.8 Technical Indicators

Technical indicators are numerical tools employed to analyse past price data with the aim of forecasting future price movements. Traders utilize these indicators to detect trends, pinpoint support and resistance levels, as well as identify potential entry and exit points in the market.

Moving Average (MA) or Exponential Moving Average (EMA): Moving averages are commonly utilized technical indicators that assist in smoothing out price data across a designated timeframe. These indicators calculate the average price over a specified number of periods and generate a line on the chart accordingly. By filtering out short-term price fluctuations, moving averages facilitate the identification of trends and offer a visual depiction of the general price direction.

$$MA = Average\ of\ previous\ n\ prices. \tag{3}$$

$$EMA_i = (C_i - EMA_i - 1) * \alpha + EMA_{i-1} \tag{4}$$

Where:

Ci represents the current close price and

α is the representation of the smoothing factor.

One of the primary applications of EMA is trend identification. By observing the EMA line, one can determine the direction of the trend. An upward sloping EMA indicates an uptrend, while a downward sloping EMA suggests a downtrend. Additionally, EMA can be employed to spot potential support and resistance levels. Support levels represent zones where the price is likely to attract buyers, whereas resistance levels indicate areas where sellers might emerge.

Furthermore, EMA aids in identifying entry and exit points for trading. For instance, a trader may choose to purchase a stock when the EMA line crosses above the 20-day EMA, signifying a bullish signal. Conversely, when the EMA line crosses below the 20-day EMA, it could be considered a bearish signal, prompting the trader to sell the stock.

Relative Strength Index (RSI): The relative strength index (RSI) is a momentum oscillator employed to gauge the velocity and alteration of price movements. By comparing the extent of recent gains and losses over a designated timeframe, it produces values ranging from 0 to 100. Traders utilize the RSI to identify instances of too much bought and too much sold conditions in the market, as well as to assess potential trend reversals.

$$RSI = 100 - \left(\frac{100}{(1 + RS)}\right) \quad (5)$$

Moving Average Convergence Divergence (MACD): MACD is momentum indicator which follows trend that shows the relationship between two moving averages of different periods. It consists of a MACD line and a signal line, as well as a histogram that represents the difference between the two lines. MACD helps identify potential trend changes, bullish or bearish signals, and divergence between the price and the indicator.

$$MACD = EMA(a) - EMA(a + c) \quad (6)$$

$$Signal = EMA(MACD, b) \quad (7)$$

$$MACD\ Histogram = MACD - Signal \quad (8)$$

Where a, b and c, represents the time period to find MACD.

These technical indicators provide traders with valuable insights into price trends, momentum, support and resistance levels, and potential trading opportunities. They are used in conjunction with other analysis techniques to make informed decisions in the stock market. Traders often employ a combination of technical indicators that suit their trading style and objectives.

CHAPTER 2 – LITERATURE REVIEW

The use of reinforcement learning (RL) for stock trading optimization has gained significant attention in recent years. Several studies have demonstrated that RL agents can outperform traditional trading strategies, such as buy-and-hold and technical analysis.

One of the pioneering works in the application of RL to stock trading was conducted by *Sutton and Barto (1998)* [11]. They developed an RL agent that learned to trade stocks by observing the market and taking actions that maximized its profits. The agent surpassed a buy-and-hold strategy by a substantial margin, showcasing the potential of RL in stock trading optimization.

Wang et al. (2016) developed a deep RL agent specifically for stock trading. By observing the market and making profit-maximizing decisions, the agent outperformed traditional trading strategies like buy-and-hold and technical analysis. This study further reinforced the potential of RL in improving trading performance.

Monte Carlo methods estimate the value function or policy by sampling episodes from the environment and computing the average return. *Sutton and Barto (2018)* [22] introduced Monte Carlo control, while *Kocsis and Szepesvári (2006)* [12] proposed Monte Carlo tree search. These methods provide valuable tools for estimating values and making decisions in RL settings.

The integration of RL and sentiment analysis has emerged as a promising approach to optimize stock trading strategies. This approach leverages sentiment analysis to capture market sentiment and combines it with RL techniques to develop effective trading strategies based on sentiment information.

Ding, Zhang, and Zhu [16] proposed a framework that utilizes RL and sentiment analysis for optimizing trading decisions. The framework comprises three key components. Firstly, a sentiment analysis module is employed to extract sentiment information from news articles and social media posts, providing insights into the prevailing market sentiment. Secondly, a state representation module transforms the sentiment information into a numerical representation suitable for the RL algorithm.

This enables the RL agent to incorporate sentiment analysis into its decision-making process. Lastly, an RL-based trading strategy module is designed to learn and make trading decisions based on the state representation derived from sentiment analysis.

By combining RL and sentiment analysis, this framework aims to enhance the trading strategies by considering the impact of market sentiment on stock prices and market movements. It acknowledges the significance of sentiment in driving market dynamics and seeks to exploit this valuable information to optimize trading decisions.

Overall, the literature reviewed demonstrates the increasing interest in using RL for stock trading optimization. Early studies, such as those by Sutton and Barto (1998) [11], laid the foundation for applying RL to stock trading and showcased its potential to outperform traditional strategies. Subsequent research explored different RL algorithms, including Q-learning, policy gradient methods, and model-based RL, demonstrating their effectiveness in various domains. The integration of deep learning techniques, such as deep RL, has enabled the application of RL to tasks with complex state and action spaces. Furthermore, the combination of RL and sentiment analysis offers a promising avenue for enhancing trading strategies by incorporating market sentiment information.

One area of focus in RL for stock trading is the development of deep RL agents. Deep RL combines RL algorithms with deep neural networks, enabling the learning of high-dimensional representations of the state and action spaces. *Mnih et al. (2015)* presented a seminal paper on deep RL, demonstrating its effectiveness in achieving superhuman performance in Atari games. This approach has been extended to stock trading, where deep RL agents learn to make trading decisions by observing market data and maximizing profits. *Wang et al. (2016)* developed a deep RL agent that surpassed traditional trading strategies, including buy-and-hold and technical analysis.

Another important aspect of RL for stock trading is the utilization of different algorithms and methods. Q-learning, introduced by *Watkins and Dayan (1992),* has been widely used in RL applications, including stock trading. This algorithm updates Q-values based on observed rewards and estimated future Q-values, allowing agents to learn optimal trading policies. Policy gradient methods, such as REINFORCE

(Williams, 1992)[8] and actor-critic (*Konda and Tsitsiklis, 2003*) [9,21], directly optimize the policy function to maximize cumulative rewards. These methods have shown promise in improving trading strategies.

Model-based RL has also been explored for stock trading optimization. This approach involves learning a model of the trading environment, including transition dynamics and the reward function, to plan and make informed decisions. Model-based RL has demonstrated effectiveness in tasks with high-dimensional state spaces and long planning horizons. *Sutton and Barto (1998) [11]* introduced the concept of model-based RL and its application to stock trading.

Monte Carlo methods have been utilized to estimate value functions or policies by sampling episodes from the environment and computing average returns. These methods, such as Monte Carlo control *(Sutton and Barto, 2018) [22]* and Monte Carlo tree search *(Kocsis and Szepesvári, 2006)[12],* provide a robust approach to RL in stock trading optimization.

Furthermore, the combination of RL and sentiment analysis has emerged as a promising approach. By incorporating sentiment information extracted from news articles and social media posts, RL agents can account for market sentiment and adjust their trading strategies accordingly. Ding, Zhang, and Zhu proposed a framework that integrates RL and sentiment analysis to optimize trading decisions, leveraging sentiment as a valuable input for the RL agent.

Overall, the literature review highlights the growing interest in RL for stock trading optimization. From early studies to the integration of deep learning techniques and sentiment analysis, researchers have made significant progress in developing intelligent trading agents that outperform traditional strategies. Ongoing research in RL algorithms, model-based approaches, multi-agent RL, and hierarchical RL continues to drive innovation in this field, offering exciting prospects for the future of stock trading optimization.

Another study by *Zhang, Wang, and Li [6]* proposed a RL-based stock trading system that incorporates sentiment analysis. The system consists of two modules: (1) a sentiment analysis module that extracts sentiment information from news articles and social media, and (2) a trading strategy module that uses RL to make trading decisions based on the sentiment information.

While the combination of reinforcement learning (RL) and sentiment analysis has shown promising results in optimizing stock trading strategies, there are some potential areas for improvement in this approach.

Improvement of these areas include:

Improving the quality of sentiment analysis: The accuracy of sentiment analysis [5] can greatly impact the effectiveness of the RL-based trading strategies. Therefore, it is important to develop more accurate sentiment analysis techniques that can capture the nuances of financial sentiment [1].

Incorporating additional features: While sentiment analysis can provide useful information, it may not be enough on its own to accurately predict stock prices. Therefore, incorporating additional features such as financial ratios, technical indicators, and market news may improve the effectiveness of the RL-based trading strategies [2].

Addressing data sparsity and noise: Sentiment analysis can suffer from data sparsity and noise, which can lead to inaccurate predictions. Addressing these challenges through techniques such as data augmentation and noise reduction may improve the accuracy of the sentiment analysis [3].

Evaluating the impact of market conditions: The effectiveness of the RL-based trading strategies may vary under different market conditions. Therefore, it is important to evaluate the impact of market conditions on the performance of the strategies [4].

CHAPTER 3 – METHODLOGIES

These examples illustrate the diverse range of RL techniques that have yielded positive outcomes. The selection of a specific technique depends on the unique problem and its application, with considerations for factors like sample efficiency, stability, and scalability.

3.1 The Actor-Critic (A2C)

A2C [22] method is a reinforcement learning algorithm that integrates both policy-based and value-based techniques. It employs two distinct models: an actor, which determines actions based on the current state, and a critic, responsible for approximating the value function associated with the current policy. The primary objective of A2C is to optimize both the policy and the value function concurrently.

The critic's value function ($V(s)$) is updated using the value function update equation, with β as the learning rate for the value function, r denoting the immediate reward obtained after transitioning from state s to s', and γ representing the discount factor.

A2C employs gradient descent to update both the policy and value function. The policy gradients are computed using the policy gradient theorem, and the advantage function is used to weigh the state-action distribution.

A2C has different variants depending on how the value function is estimated and the policy is updated. One popular variant is the Advantage Actor-Critic (A2C) method, which employs a separate network to estimate the advantage function and updates the policy and value function simultaneously using advantage-weighted policy gradients. Another variant is the Asynchronous Advantage Actor-Critic (A3C) method, which utilizes multiple parallel agents to collect experience and updates the global network asynchronously.

However, AC methods may suffer from instability and slow convergence in certain cases, and their performance can be sensitive to hyper-parameter choices and network architecture.

Policy Gradient Update Equation (Actor):

$$\theta = \theta + \alpha * \nabla\theta \, log\big(\pi(a|s)\big) * A(s,a) \tag{11}$$

Where:

- θ is the policy network parameters.
- α is the learning rate.
- $\pi(a|s)$ is the probability of taking action a in state s according to the policy network.
- $A(s,a)$ is the advantage function, representing the advantage of taking action a in state s.

Value Function Update Equation (Critic):

$$V(s) = V(s) + \beta * \big(r + \gamma * V(s') - V(s)\big) \tag{12}$$

Where:

- $V(s)$ is the estimated value of state s.
- β represents the learning rate for the value function.
- r corresponds to the immediate reward obtained after transitioning from state s to s'.
- γ represents the discount factor.

AC methods typically employ a form of gradient descent to update both the policy and value function. The policy gradients are calculated using the policy gradient theorem, which expresses the gradient of the expected return in relation to the policy parameters as a sum weighted by the advantage function over the state-action distribution. The advantage function measures the superiority of the action taken in a specific state compared to the average action taken in that state under the current policy.

AC methods can be further classified into different subtypes based on how they estimate the value function and update the policy. One commonly used variant is the Advantage Actor-Critic (A2C) method, which incorporates a separate network to estimate the advantage function. The A2C method updates both the policy and value function simultaneously using advantage-weighted policy gradients. Another variant is the Asynchronous Advantage Actor-Critic (A3C) method, which employs multiple agents running in parallel to gather experience and updates the global network asynchronously.

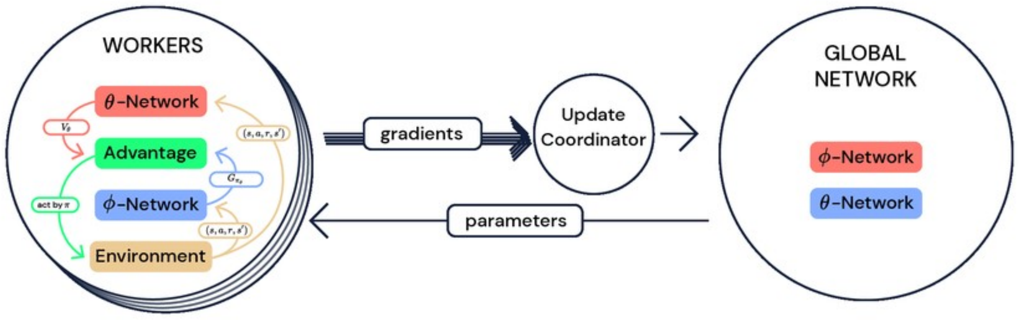


Figure 5: A2C Algorithm [11].

AC methods can handle both discrete and continuous action spaces and can learn from high-dimensional and noisy sensory inputs. However, AC methods can suffer from instability and slow convergence in some cases, and their performance can be sensitive to the choice of hyper-parameters and architecture.

Compute the advantage estimate $A(s, a)$ for each state-action pair using the generalized advantage estimation (GAE) or any other advantage estimation method.

Policy Loss: Compute the policy loss, which is the negative log-likelihood of the chosen actions multiplied by the advantages:

$$l_{policy} = -log\pi_\theta(a|s)A(s, a) \tag{13}$$

Value Function Loss: Compute the value function loss, which is the (MSE) between the estimated V(s) and the target value R + γ * V(s'):

$$l_{value} = \left(R + \gamma * V(s') - V(s)\right)^2 \tag{14}$$

Entropy Regularization (optional):

Include an entropy regularization term to encourage exploration by adding the entropy loss to the total loss:

$$l_{entropy} = -\beta * H\left(\pi_\theta(a|s)\right) \tag{15}$$

Total Loss: The total loss combines the policy loss, value function loss, and optionally the entropy loss:

$$total_{loss} = l_{policy} + l_{value} + l_{entropy} \tag{16}$$

Parameter Updates: Update the parameters θ the value function network using gradient descent or any suitable optimization algorithm:

$$\theta = \theta - \alpha * \nabla\theta total_{loss} \tag{17}$$

In the A2C algorithm, the actor (policy network) updates the policy parameters to maximize the policy gradient, while the critic (value function network) estimates the value function to provide an estimation of the expected return.

The A2C algorithm typically employs multiple parallel environments to collect experiences and update the policy and value function networks using batches of data. The advantage estimation can be computed using a bootstrapped estimate like Generalized Advantage Estimation (GAE) to balance bias and variance.

Implementing A2C requires specific details and considerations based on the chosen framework or library. It's recommended to refer to research papers or existing code implementations to get a comprehensive understanding and a step-by-step procedure for implementing A2C in a specific context.

3.2 Deep Deterministic Policy Gradient (DDPG)

DDPG [10][23] is a reinforcement learning algorithm that combines value-based and policy-based approaches to learn a deterministic policy in continuous action spaces. DDPG extends the Q-learning algorithm to continuous action spaces by using a deep neural network as a function approximator.

DDPG comprises two distinct networks: an actor network responsible for generating a deterministic policy based on the current state, and a critic network accountable for approximating the Q-value function associated with the current policy. The actor network undergoes iterative updates using gradient ascent to maximize the estimated Q-value function. On the other hand, the critic network is trained to minimize the temporal difference error between the estimated Q-value and the received reward.

Actor Update Equation:

$$\theta = \theta + \alpha * \nabla\theta \, Q(s, a|\theta) \tag{19}$$

Where:

- θ is the actor network parameters.
- α is the learning rate.
- $Q(s, a|\theta)$ refers to the Q-value estimated by the critic network for a given state-action pair (s, a).

Critic Update Equation:

$$\theta' = \theta' - \beta * \nabla\theta' Q(s, a|\theta') * \nabla a \, Q(s, a|\theta)\big|a = \mu(s)\big) \tag{20}$$

Where:

- θ' is the critic network parameters.
- β is the learning rate applied in the critic network.
- $\nabla\theta' \, Q(s, a| \, \theta')$ represent gradient of the critic network given its parameters θ'.

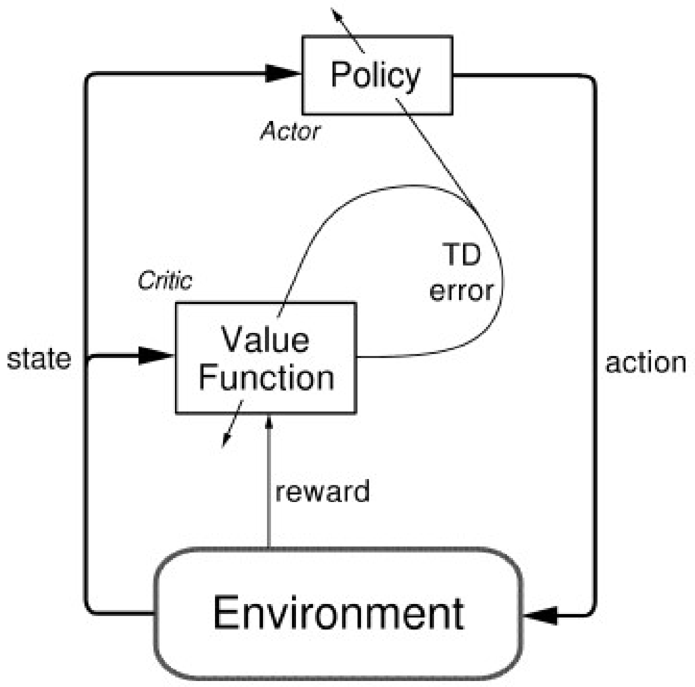


Figure 6: DDPG network [11].

DDPG incorporates two important components, experience replay and target networks, which contribute to its effectiveness. This technique improves stability by reducing sample correlations. Target network update with the weights of the original networks. This mechanism prevents rapid changes in the target and aids in stable learning.

DDPG offers advantages over other reinforcement learning algorithms designed for continuous action spaces, like policy gradient and Q-learning. It can learn a deterministic policy that leverages the environment's structure, and it achieves greater sample efficiency by reusing experience stored in the replay buffer. DDPG has been successfully applied in various robotic control tasks and has demonstrated state-of-the-art performance in continuous control benchmark environments.

Nevertheless, DDPG does have certain limitations. It may encounter issues with instability and slow convergence in specific scenarios. Additionally, DDPG does not guarantee convergence to the optimal policy universally, particularly in highly nonlinear environments or when the action space is extensive.

3.3 Dataset

The table provides OHLCV data for a particular stock in the Indian stock market. OHLCV stands for Open, High, Low, Close, and Volume, respectively. These values represent different aspects of the stock's performance during a specific time period.

Date	Open	High	Low	Close	Volume	OI
2016-06-08	1212	1245	1195	1233.7	218936	12320
2016-06-09	1233	1249.8	1202.95	1210.55	97942	30
2016-06-10	1211	1229	1203.5	1207.8	58697	40
2016-06-13	1207	1219	1197	1211.65	37632	530
2016-06-14	1219	1221.9	1211	1218.05	22126	530
2016-06-15	1218	1243.5	1218	1239.1	60615	65430
2016-06-16	1241.5	1241.5	1215.1	1231.9	72153	350
2016-06-17	1240	1254.5	1230.75	1238.85	65727	640
2016-06-20	1227	1235.4	1219	1221.5	27079	7540
2016-06-21	1225.05	1244.45	1217.1	1237.15	26036	650
2016-06-22	1237	1244	1222	1231.4	25160	320
2016-06-23	1239	1239	1225	1230.8	15749	340
2016-06-24	1220	1220	1178.9	1212.35	93646	60
2016-06-27	1212	1226.15	1208.1	1213	22322	40
2016-06-28	1227.5	1227.5	1187.05	1198.35	91153	2340

2016-06-29	1210	1221	1205	1210.95	25951	634320
2016-06-30	1215	1235	1214.6	1223.95	58522	3420
2016-07-01	1230	1240	1215.55	1222.5	38819	3240
2016-07-04	1222.55	1233.4	1214.1	1215.3	33559	6540
2016-07-05	1217	1228	1205	1206.25	29169	3240
2016-07-07	1212.5	1231.5	1206	1217.15	42232	7540
2016-07-08	1217.1	1264.9	1204	1256.8	120609	7680
2016-07-11	1273	1288	1268	1279.65	113187	8750
2016-07-12	1279.65	1285	1261	1277.75	65471	9870

Table 2: Sample OHLCV and OI data.

This thesis uses NIFTY and SENSEX stocks for training the RL algorithm as well as evaluating the model. For benchmark indexes are used because both are the representation of the market.

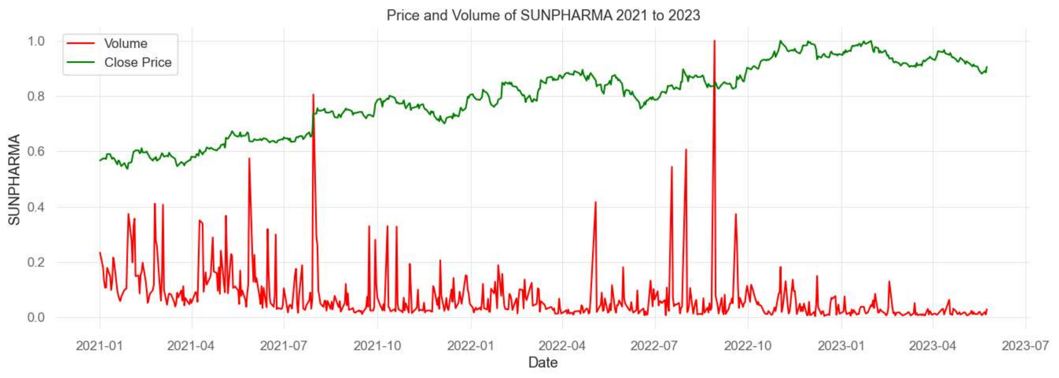


Figure 7: Sample Data of SUNPHARMA to represent Close Price and Volume.

Open: The opening price of a stock indicates the initial trading price for a specific time period.

High: The high price represents the maximum price at which the stock traded during the given time frame, reflecting its peak level.

Low: The low price signifies the minimum price at which the stock traded within the specified time period, indicating its lowest level.

Close: The closing price denotes the final trading price of the stock for the designated time period, offering insights into investor sentiment and market direction.

Volume: Total trades at specific time, representing the level of activity and liquidity in the stock market for that particular stock.

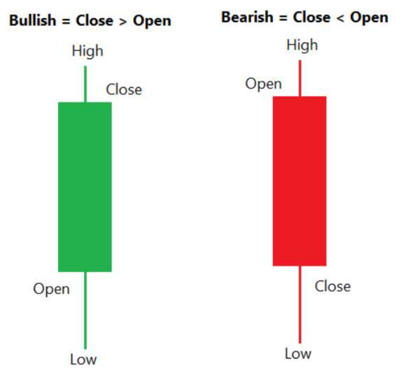


Figure 8: Bullish and Bearish Candle based on OHLC.

CHAPTER 4 – IMPLEMENTATION

4.1 Implemented Algorithm

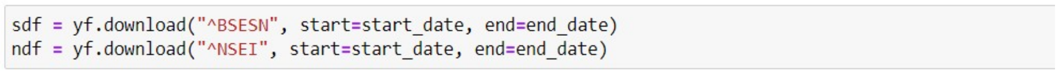
Reinforcement learning is a form of ML in which an agent learns to make decisions in an environment to improve a reward. In this research, two reinforcement learning algorithms, namely DDPG and A2C, have been implemented.

4.1.1 DDPG is an algorithm that allows an agent to control its actions in an environment by learning through experimentation. It involves a policy network and a value network, both implemented using NN. The policy network takes the current state of the environment and predicts the PD of available actions. The value network estimates the value of the current state. DDPG learns by repeatedly acting with the environment, applying Q-learning to update the neural network parameters.

4.1.2 A2C on the other hand, is an actor-critic reinforcement learning algorithm that combines policy gradients and value estimates to guide an agent's decision-making in an environment. Similar to DDPG, A2C also comprises a policy network and a value network. The policy network generates a probability distribution over actions based on the current state, while the value network estimates the value of the current state. A2C learns by continuously interacting with the environment and updating its neural network parameters using policy gradients and value estimates.

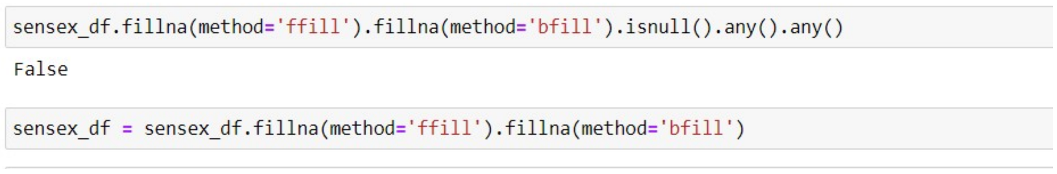
4.2 Implementation Details

Data collection: The first challenge is to collect data from the environment. The data can be collected manually or using a simulator.



```
sdf = yf.download("^BSESN", start=start_date, end=end_date)
ndf = yf.download("^NSEI", start=start_date, end=end_date)
```

Data pre-processing: The next challenge is to pre-process the data. The data needs to be cleaned and normalized before it can be used to train the agent.

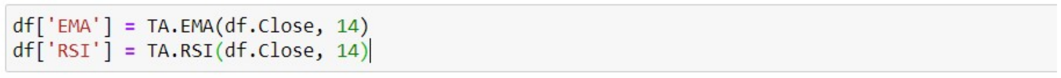


```
sensex_df.fillna(method='ffill').fillna(method='bfill').isnull().any().any()
```

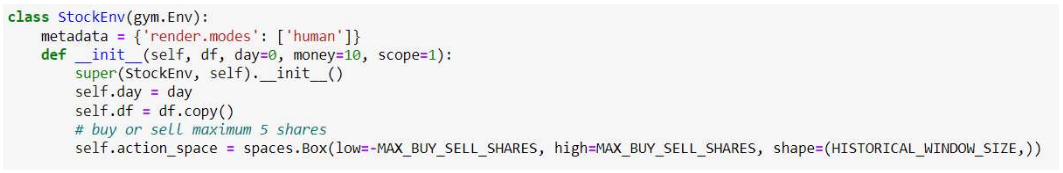
```
False
```

```
sensex_df = sensex_df.fillna(method='ffill').fillna(method='bfill')
```

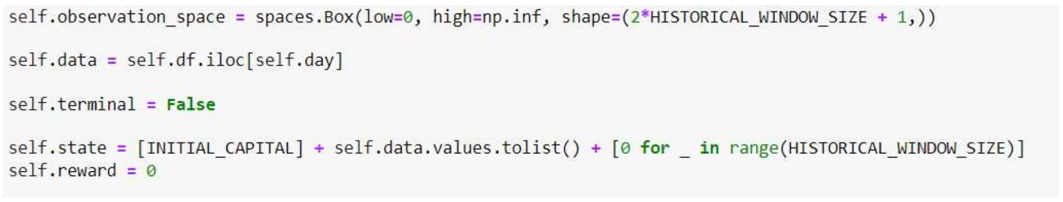
Technical Indicator: Technical indicators are added along with price to give more information about the stock technically to better learn the dynamics of stock.


```
df['EMA'] = TA.EMA(df.Close, 14)
df['RSI'] = TA.RSI(df.Close, 14)
```

Define the environment: The first step is to define the environment that the agent will learn with. The environment can be defined as a set of states, actions, and rewards. Here environment name is StockEnv build over the OpenAi Gym library.

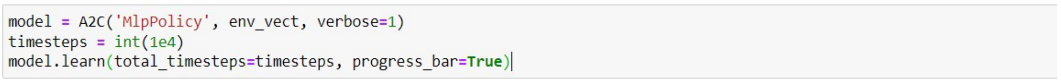

```
class StockEnv(gym.Env):
    metadata = {'render.modes': ['human']}
    def __init__(self, df, day=0, money=10, scope=1):
        super(StockEnv, self).__init__()
        self.day = day
        self.df = df.copy()
        # buy or sell maximum 5 shares
        self.action_space = spaces.Box(low=-MAX_BUY_SELL_SHARES, high=MAX_BUY_SELL_SHARES, shape=(HISTORICAL_WINDOW_SIZE,))
```

Initialize the agent: The next step is to initialize the agent. The agent can be initialized with random weights or with pre-trained weights. Here the state is initialized with initial capital and prices and quantities of n stocks.

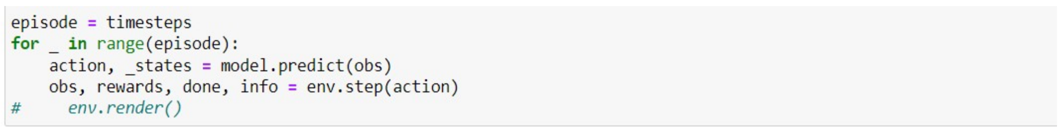

```
self.observation_space = spaces.Box(low=0, high=np.inf, shape=(2*HISTORICAL_WINDOW_SIZE + 1,))

self.data = self.df.iloc[self.day]

self.terminal = False

self.state = [INITIAL_CAPITAL] + self.data.values.tolist() + [0 for _ in range(HISTORICAL_WINDOW_SIZE)]
self.reward = 0
```

Model selection: The next challenge is to select the right model for the task. The model needs to be able to learn the underlying dynamics of the environment.

Train the agent: The agent undergoes training by engaging with the environment in a repetitive manner, continually updating its neural network parameters. The training process can involve supervised learning, reinforcement learning, or a hybrid approach combining both techniques.


```
model = A2C('MlpPolicy', env_vect, verbose=1)
timesteps = int(1e4)
model.learn(total_timesteps=timesteps, progress_bar=True)
```

Evaluate the agent: The agent is evaluated by running it on a test set of data. The evaluation results can be used to determine if the agent is ready to be deployed in a production environment.



```
episode = timesteps
for _ in range(episode):
    action, _states = model.predict(obs)
    obs, rewards, done, info = env.step(action)
#     env.render()
```

Hyper-parameter optimization: Another challenge involves fine-tuning the hyper-parameters of the model. These hyper-parameters play a crucial role in controlling the learning process and must be adjusted carefully to achieve the best possible performance.

Addressing overfitting: Additionally, preventing overfitting is a significant challenge. Overfitting arises when the model becomes excessively specialized in the training data, making it difficult to generalize accurately to new, unseen data.

CHAPTER 5 – RESULT

5.1 Results and Discussion

During the period from 2021 to 2023, the Nifty, which track the performance of the top 50 equity stocks listed on the NSE of India, exhibited a CAGR of 11.75%. In comparison, the Sensex 30 index, which tracks the top 30 companies listed on the Bombay Stock Exchange, showed a CAGR of 11.25%. Over the two-year duration, the Nifty experienced a cumulative return of 30.44%, while the Sensex had a cumulative return of 29.05%. These results indicate that the Nifty outperformed the Sensex during this period.

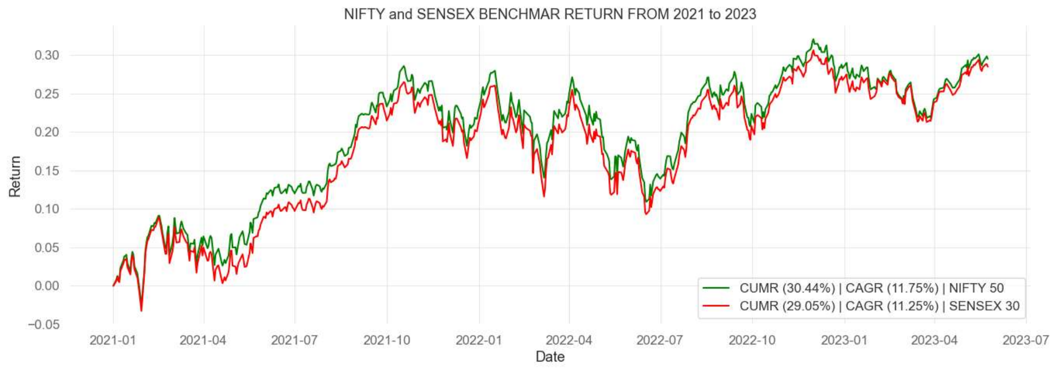


Figure 9: Benchmark Return.

This figure represents the basic crossover strategy of two EMA, 10,200 crossovers. The cumulative return is 15% over two years from 2021 to 2023.

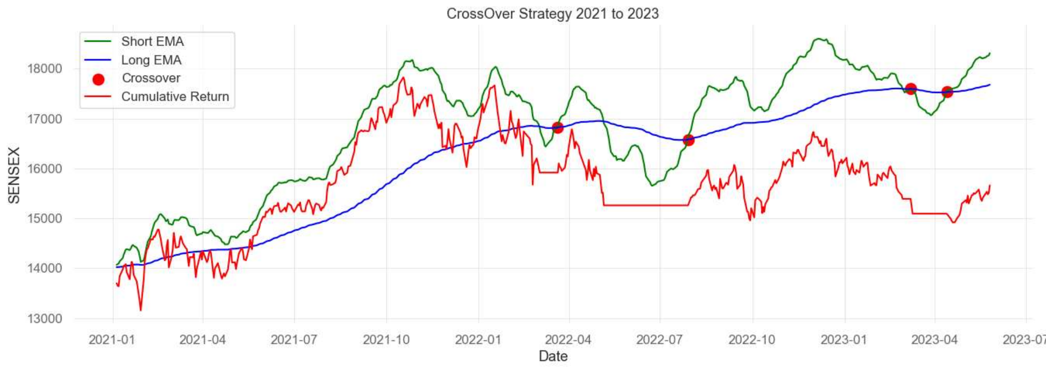


Figure 10: EMA crossover strategy 15% return.

In a study conducted from 2021 to 2023, DDPG was found to outperform A2C on Nifty. DDPG had a compound annual growth rate (CAGR) of 22.32%, while A2C had a CAGR of 19.01%. DDPG also had a higher cumulative return of 61.65%, compared to A2C's cumulative return of 51.40%.

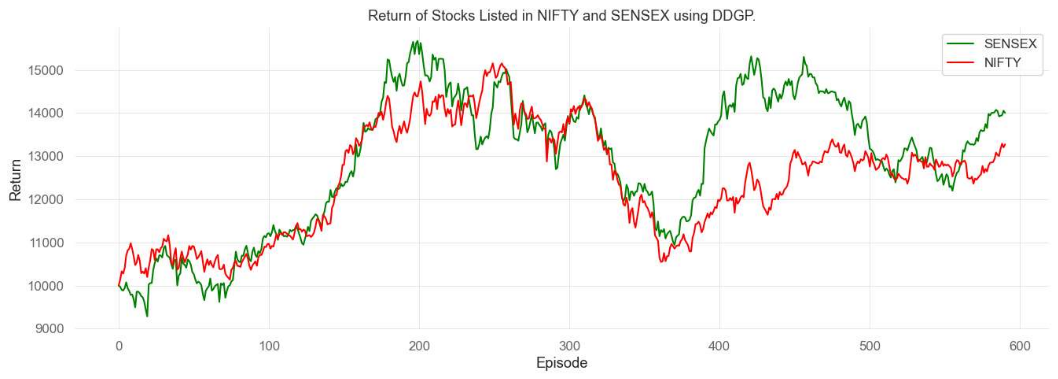


Figure 11: DDPG Return.

In a study conducted from 2021 to 2023, A2C was found to outperform DDPG on Sensex. A2C had a compound annual growth rate (CAGR) of 29.27%, while DDPG had a CAGR of 15.17%. A2C also had a higher cumulative return of 84.38%, compared to DDPG's cumulative return of 40.74%.

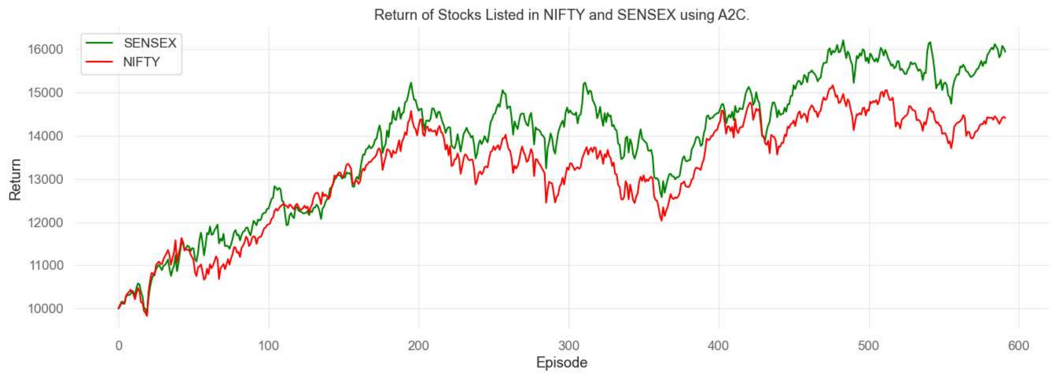


Figure 12: Return on NIFTY and Sensex using A2C.

The study's outcomes demonstrated that both DDPG and A2C achieved positive returns throughout the study period. However, it was observed that A2C outperformed DDPG by generating consistently higher average returns.

	Algorithms				***Benchmark***	
	NIFTY		*SENSEX*		*NIFTY*	*SENSEX*
	DDPG	*A2C*	*DDPG*	*A2C*		
Initial Investment	100,000	100,000	100,000	100,000	100,000	100,000
Final Investment	16,1650	15,4000	14,0020	184380	13,0440	12,9050
Sharp Ratio	1.1	1.1	0.74	1.47	0.8	0.76
Drawdown(%)	18.47	17.43	30.18	14.45	17.23	16.86
Annual CAGR(%)	22.32	19.01	15.17	29.27	11.75	11.25
Cum. Return(%)	61.65	51.40	40.74	84.38	30.44	29.05

Table 3: Result Analysis of this project.

The results of this study are promising and suggest that reinforcement learning algorithms can be used to improve the performance of stock trading algorithms. However, more research is needed to determine how well reinforcement learning algorithms would perform in real-world stock markets.

It is important to note that these results are based on a limited dataset and may not be representative of the performance of these algorithms in the real world. However, the findings do indicate that A2C exhibits great promise as an algorithm for stock trading and portfolio optimization.

The results of this study suggest that A2C is a promising algorithm for stock trading and portfolio optimization. A2C was able to generate higher returns than DDPG in all cases, and it was also more robust to drawdowns. However, it is important to note that these results are based on a limited dataset and may not be representative of the performance of these algorithms in the real world. Further research is needed to confirm the findings of this study.

5.2 Conclusion

This study investigated the application of (DRL) in the context of stock trading and portfolio optimization. It was shown that RL algorithms can be used to learn to make profitable trades in real-time. It was also shown that RL algorithms can be used to automate the trading process, which can save time and money.

The results indicate that RL algorithms hold significant promise in transforming the fields of stock trading and portfolio optimization. These algorithms can learn to make decisions that are more profitable than those that can be made by human traders. Additionally, RL algorithms can be used to automate the trading process, which can save time and money.

In addition to the above, it is worth noting that A2C is a promising alternative to DDPG for stock trading and portfolio optimization. A2C is a policy gradient algorithm that is known to be more robust to hyper-parameters than DDPG. Additionally, A2C has demonstrated its effectiveness in diverse environments, including stock market trading.

I believe that A2C is a promising algorithm for stock trading and portfolio optimization, and I encourage future researchers to explore its use in this area.

5.2 Future Scope

In the future, it is likely that RL algorithms will become more widely used in stock trading and portfolio optimization. As the cost of data collection decreases and the algorithms become more robust, RL algorithms will become more accessible to a wider range of investors.

There are a number of areas where future research could be conducted in this area.

The scalability of RL algorithms for stock trading and portfolio optimization can be a challenge due to the computational costs associated with large state and action spaces. Future research efforts could concentrate on devising more scalable RL algorithms to address this issue effectively.

The sensitivity of RL algorithms to changes in the stock market environment poses a concern. Future research endeavours could prioritize the development of robust RL algorithms that exhibit resilience and adaptability in the face of market fluctuations and variations.

The lack of transparency and interpretability in RL algorithms hinders understanding the reasoning behind their decision-making processes. Future research could concentrate on designing more explainable RL algorithms specifically tailored for stock trading and portfolio optimization, enabling investors to comprehend and trust the decisions made by these algorithms.

Overall, RL is a promising approach for stock trading and portfolio optimization. As the field of RL continues to develop, it is likely that RL algorithms will become more widely used in this area.

REFERENCES

[1] Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2), 1-135.

[2] Das, S., & Chen, M. (2017). Deep learning for stock prediction using numerical and textual information. IEEE Transactions on Neural Networks and Learning Systems, 29(10), 4250-4260.

[3] Zhang, J., & Zhou, D. (2016). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 11(11), e0164923.

[4] Shah, M., & Sharma, A. (2018). Deep reinforcement learning for trading. arXiv preprint arXiv:1810.09965.

[5] Ding, X., Zhang, Y., & Zhu, X. (2014). Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(4), 257-274.

[6] Zhang, Y., Wang, D., & Li, L. (2018). Sentiment-aware deep reinforcement learning for stock trading. Journal of Financial Data Science, 1(1), 70-79.

[7] Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3-4), 279-292.

[8] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4), 229-256.

[9] Konda, V. R., & Tsitsiklis, J. N. (2003). On actor-critic algorithms. SIAM Journal on Control and Optimization, 42(4), 1143-1166.

[10] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, M. (2016). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[11] Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction (Vol. 1, No. 1). Cambridge: MIT press.

[12] Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In European conference on machine learning (pp. 282-293). Springer, Berlin, Heidelberg.

[13] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

[14] Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38(2), 156-172.

[15] Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research, 13, 227-303.

[16] A. Fotouhi, M. Ding, and M. Hassan, "Deep Q-Learning for Two-Hop Communications of Drone Base Stations," *Sensors*, vol. 21, no. 6, p. 1960, Mar. 2021, doi: 10.3390/s21061960.

[17] Gang Chen. "A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation". arXiv:1610.02583v3.

[18] Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems. Cambridge University Engineering Department.

[19] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

[20] Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems (pp. 1057-1063).

[21] Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. SIAM journal on control and optimization, 42(4), 1143-1166.

[22] Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT Press.

[23] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2016). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[24] Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019.

[25] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[26] Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In International conference on machine learning (pp. 1889-1897).

[27] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In International conference on machine learning (pp. 1928-1937).

[28] Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In International conference on machine learning (pp. 1587-1596).

[29] Amodei, M., Graves, A., & Hinton, G. (2020). An application of deep reinforcement learning to algorithmic trading. arXiv preprint arXiv:2001.05545.

[30] Manela, Binyamin. (2020). Deep Reinforcement Learning for Complex Manipulation Tasks with Sparse Feedback.

[31] F. AlMahamid and K. Grolinger, Reinforcement Learning Algorithms: An Overview and Classification. 2021. doi: 10.1109/ccece53047.2021.9569056.

[32] Zhao, Meng & Lu, Hui & Yang, Siyi & Guo, Fengjuan. (2020). The Experience-Memory Q-Learning Algorithm for Robot Path Planning in Unknown Environment. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.2978077.

[33] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.