University of Macau

Faculty of Science and Technology

Enhancing Stock Trading Strategies with Integrated Feature Enhancement and Reward Shaping: A Reinforcement Learning Approach

by

WONG KAI YUAN, Student No: DC026157

Graduation Project Report submitted in partial fulfillment
of the requirements of the Degree of
Bachelor of Science in Computer Science

Project Supervisor

Prof Ryan Leong Hou U

13 May 2024

DECLARATION

I sincerely declare that:

1. I and my teammates are the sole authors of this report,
2. All the information contained in this report is certain and correct to the best of my knowledge,
3. I declare that the thesis here submitted is original except for the source materials explicitly acknowledged and that this thesis or parts of this thesis have not been previously submitted for the same degree or for a different degree, and
4. I also acknowledge that I am aware of the Rules on Handling Student Academic Dishonesty and the Regulations of the Student Discipline of the University of Macau.

Signature	:	WONG KAI YUAN
Name	:	WONG KAI YUAN
Student No.	:	DC026157
Date	:	13 May 2024

ACKNOWLEDGEMENTS

Thanks for the opportunity to work on this project. I also want to extend my thanks to all those who have supported me over my academic career.

I take this opportunity to extend my gratitude to my supervisor, Prof. Ryan, and all the teachers and the staff of the Computer Science Department at the University of Macau. The work of teaching, of making the school a place where people are curious, open to new ideas over those four years, helped me grow a lot. All this is due to a strong curriculum and highly committed professors from the department, from whom I have collected the skills required for the job and beyond.

The other thing is that I want to thank the University of Macau for giving tools and facilities. By accessing the technology in the labs and study materials that were needed for the project, it was possible to finish the project and learn more.

Lastly, I would like to acknowledge my family and thank them from the bottom of my heart for all the love and support in school. They have been my rock.

This would not have been possible without the helpful environment in the University of Macau, where people are on the same trip. It's a project which grows and learns in four years.

ABSTRACT

This project looks into how reinforcement learning (RL) can be used in stock trading. The goal is to improve trade strategies by using advanced techniques for feature enhancement and reward shaping. Because financial markets are so complicated and hard to predict, the project needed a very careful way of develop and testing models.

The study started with a thorough review of the existing literature to learn about current approaches in RL and how they can be used in different areas, with a focus on stock trading. Based on this basic information, the experimental design was guided, and different methods for improving features were used to help the model learn more about how markets work. At first, the model only used basic price data. Over time, it got more expanded by adding more market indicators, such as technical analysis tools like MACD, RSI, and lastly, even candlestick patterns.

Along with features enhancement, the project tried a number of reward shaping strategies to help the model make trading decisions that are profitable while also minimizing risk. At first, there was a simple reward function that linked actions directly to changes in price. Later, more complex reward shaping ideas were added like loss aversion, risk handling, avoid overconfidence, and trend following finance strategy.

To make sure these strategies worked, they were put through a lot of testing using historical data from stocks like Goldman Sachs. The models were then tried on other stocks (APPL, KO, XOM ..) to see how robust and flexible they were. Performance indicators like total profit, Sharpe ratio, and maximum drawdown were used to rate each version of the model.

The results show that reinforcement learning has a lot of potential to improve stock trading methods. However, the success of these kinds of applications depends a lot on how well the feature sets and reward functions are designed. This project not only shows how RL could be used in financial situations, but it also shows how challenging and complicated it can be.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION 9

1.1 Introduction to Reinforcement Learning 9

1.2 Introduction to Stocks Trading 10

1.3 Research Question and Hypothesis 11
1.3.1 Research Question 11
1.3.2 Hypothesis 12

1.4 Objectives 12

1.5 Scope and Delimitation 13

CHAPTER 2. LITERATURE REVIEW 15

2.1 Reinforcement Learning Literature Review 15

2.2 Reinforcement Learning in Stocks Trading 16

2.3 Review of Feature Enhancement Techniques 17

2.4 Review of Reward Shaping Techniques 18

2.5 Identification of Research Gap 19

CHAPTER 3. PROJECT EXECUTION SCHEDULE 20

3.1 Foundation in Reinforcement Learning (October - December 2023) 20

3.2 Exploration of Environments and Algorithm & Writing Interim Report (January - February 2024) 21

3.3 Development and Testing (March - April 2024) 21

3.4 Finalization and Documentation (April - May 2024) 21

3.5 Milestones & Deliverables 22

CHAPTER 4. METHODOLOGY FOR FEATURE ENHANCEMENT 23

4.1 Problem Statement 23

4.2 Original Feature Function 23

4.3 Version 1 Feature Enhancement: Extended Market Features 24

4.4 Version 2 Feature Enhancement: Integrating Technical Indicators 25
4.4.1 Integrating Technical Indicators 25
4.4.2 Modification of Model Architect 27

4.5 Version 3 Feature Enhancement: Integrating Candlestick Pattern 28

CHAPTER 5. METHODOLOGY FOR REWARD SHAPING 31

5.1 Problem Statement 31

5.2 Original Reward Function 32

5.3 Version 1 Reward Shaping: Behavioral Reinforcement Reward 32
5.3.1 Static Behavioral Reinforcement Reward 32
5.3.2 Dynamic Behavioral Reinforcement Reward 33

5.4 Version 2 Reward Shaping: Risk Handling via thresholding 34

5.5 Version 3 Reward Shaping: Risk Handling via Avoid Overconfidence 36

5.6 Notable Failed Reward Shaping 37

CHAPTER 6. RESULTS 39

6.1 Implementation of experiment 39

6.2 Benchmark and Performance Graph 40

6.3 Testing Across Different Stocks 42
6.3.1 Rationale 42
6.3.2 Testing Results 42

CHAPTER 7. DISCUSSION 44

7.1 What I've Learned in this project 44

7.2 Challenge Met 45

7.3 Future Work 46

CHAPTER 8. ETHICS AND PROFESSIONAL CONDUCT 47

8.1 Ethical Consideration 47

8.2 Ethical Guideline Proposal 48

CHAPTER 9. CONCLUSION 49

9.1 Summary of work 49

9.2 Final Thoughts 50

CHAPTER 10. REFERENCE 51

CHAPTER 11. APPENDIX 53

LIST OF FIGURES

Figure 1: Illustration of stock price of Berkshire 9

Figure 2: MDP in stocks trading 10

Figure 3: Literature Timeline of Key development of RL 17

Figure 4: RL framework using ensemble strategies 18

Figure 5: Gantt Chart of each task in big overview 20

Figure 6: Gantt Chart for each subtask in detailed view 20

Figure 7: Feature Enhancement Overview 30

Figure 8: Reward Shaping Overview 37

Figure 9: Constant buying on failed reward shaping 37

Figure 10: Bar Chart of result of feature enhancement and reward shaping 41

Figure 11: Paired bar charts showing the comparison of benchmark and enhanced results 43

LIST OF TABLES

Table 1: Delimitation of the project 14
Table 2: Milestones & Deliverables in detailed 22
Table 3: Technical Indicators Details 26
Table 4: Testing result on different model architecture 27
Table 5: Details of Candlestick Pattern 28
Table 6: Characteristic of rewards 31
Table 7: Overview of results of feature enhancement 40
Table 8: Overview of results of reward shaping 41
Table 9: Testing result across different stocks 42
Table 10: Ethical Guideline Proposal 48
Table 11: Full Technical Indicators Details 54
Table 12: Full Candlestick Pattern Details 56

CHAPTER 1. INTRODUCTION

1.1 Motivation

Stock trading, which is the buying and selling of shares in publicly traded companies, is one of the most important components of the financial markets. Traders and buyers in the stock market try to make money by speculating how prices will move. They do this through short-term trading plans and long-term investment plans. This market is a complicated, ever-changing system that is affected by many factors that make it volatile and hard to predict.

At its core, the stock market is based on how supply and demand work together. At any given time, the price of a stock is set by the balance between people who want to buy and people who want to sell. This way of setting prices is also affected by macroeconomic indicators like interest rates, inflation, and political stability. It is also affected by factors that happen within the company, like earnings reports, management changes, or new product launches. Because of these things, the stock market is very responsive to new information, which means that stock prices change very quickly. This non-stationarility makes it hard for any trading strategy to work because past patterns and behaviors may not accurately predict what will happen in the future.

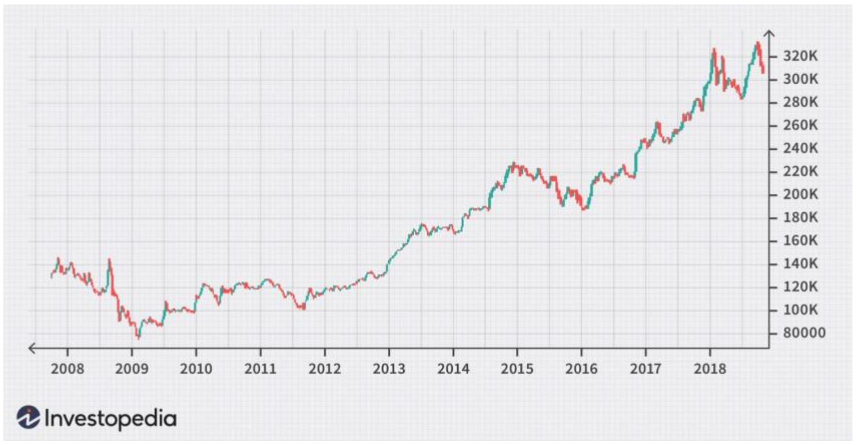


Figure 1: Illustration of stock price of Berkshire [1]

In the past few years, there was a lot of voice about how AI would change many fields, including stock trading. As an example, algorithmic trading has changed the stock market in a new way, making it work better and trade deals go through faster. The way the market works has changed because of high-frequency trading (HFT) programmes, which can carry out orders in a matter of seconds. These changes have an effect on both market volatility and liquidity. Many trades are now made on financial markets around the world by HFT and other programmes that trade automatically. [2]

On the other hand, Reinforcement Learning (RL) is a distinct type of machine learning that lets agents make the best choices by interacting with a dynamic environment and learning from their actions. In supervised learning, examples from a labeled dataset are used to teach, but in RL, agents learn from the results of their actions, getting rewards or penalties depending on how well they do. This way let's agents come up with strategies that maximise cumulative rewards over time. In difficult decision-making tasks, they often do better than humans.

When I discover more about reinforcement learning (RL) and learn about its application like master games like AlphaGo [3] and do different tasks at superhuman levels, I was very interested. This made I wonder if RL could also get good at the "game" of stock trading. The initial motivation I started this project in the first place was to look into how RL could help me understand and maybe even come up with new strategy to improve trade strategies. At this point, RL seems like the perfect fit for unpredictable nature of stocks trading.

1.2 Background of the research

The agent, the environment, the policy, the reward system, and the value function are the most important elements of an RL system. Based on what it observes in its environment, the agent acts, and for that it gets rewards that help it decide what to do next. The behavior of an agent is controlled by policies, which map states to actions. The goal is to maximize the total future reward. The value function is very important because it calculates the expected return from each state, which shows how likely of future success. [4]

The Markov Decision Process (MDP) is one of the most important ideas in RL. It's a mathematical framework that describes an environment in RL where results are partly random and partly controlled by an agent who makes decisions. MDPs have characteristics like: set of states, actions, a transition model, and reward functions. They give us the ideas we need to understand a lot of RL problems, with the main idea that the future is independent from the past.

Therefore, it's important for us to transform our stock trading into MDP model before we start with anything:

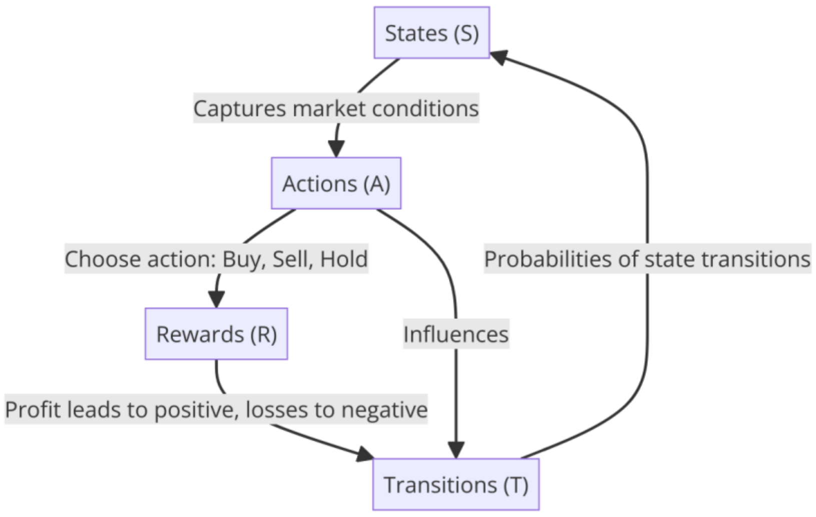

States (S)
Captures market conditions
Actions (A)
Choose action: Buy, Sell, Hold
Probabilities of state transitions
Rewards (R)
Influences
Profit leads to positive, losses to negative
Transitions (T)

Figure 2: MDP in stocks trading

- **States (S)**: States show the current state of the environment. These states can be seen through market indicators like stock prices, volumes, and related technical indicators. St = {price, volume, indicators}

- **Actions (A)**: The agent can buy, sell as an action. Actions are the choices it can make at each state. It's at = [buy, sell]

- **Rewards (R)**: Rewards give the agent quick feedback on their actions, which is usually measured by how much money they make or lose from trading decisions. R(st,at) = gain or loss after doing move at in state st

- **Transitions (T)**: The odds that show how likely it is that something will change from one state to another after an action. The movements of the stock market have an effect on these odds. $M(st+1|st,bt)$=the chance of going from state b to state t+1 after action bt.P(st+1|st,at) is the chance of going from state st to state st+1 after action at. [6]

Reinforcement Learning (RL) has a lot of potential in this area because it can change and learn the best ways to do things by constantly interacting with its envinronment.[5] RL agents are made to maximize a cumulative reward in environments that are complex and changeable. This makes them a candidate for handling the stock market's many complexities. It's possible that these agents can come up with trade strategies that adapt to new information and changing market conditions. This could lead to higher returns than with traditional models or trading by hand.

As I dive deeper, it became clear that while RL has a lot of potential, the standard approach of RL don't always work when used in the financial markets. There are some problems that are only found in stock trading, like a lot of noise, non-stationary data, and the need for accuracy when the economy is unclear. I knew that improving RL's ability to perceive and make decisions was important if I wanted to really use it in this field. This realisation led me to study and research on feature enhancement and reward shaping, two important techniques that RL models depend on.

1.3 Research Question and Hypothesis

1.3.1 Research Question

The question this research tries to answer is: "How do feature enhancement and reward shaping affect the performance of a RL model in an unstable financial market?" To find out how combining two important parts of reinforcement learning affects trading methods in unstable financial markets, this question is at the heart of the research.

- Feature Enhancement: Improving the data that is fed into the RL model by adding more relevant features or changing current features to better capture the patterns that are hidden in the data.

- Reward Shaping: Any change to the form of rewards in reinforcement learning that helps the learning process work better. This means making the reward function work in a way that encourages good behaviour and stops bad behaviour.

1.3.2 Hypothesis

The study is based on two main hypothesis that were made to look into how feature enhancement and reward shaping affect the performance of stock trading models:

The first hypothesis is that adding more features to reinforcement learning models in stock trading makes them more accurate at predicting the future and more profitable by giving more detailed information about how the market works. This idea says that if you give the model a more complete and detailed set of input features, it will be able to make better choices, which will improve its performance metrics.

The second hypothesis is that reward shaping improves the risk-adjusted returns of reinforcement learning models used in stock trading compared to models that don't use custom reward functions. This idea says that the model can be tweaked to favour strategies that give higher risk-adjusted returns by carefully developing the reward system. This could lead to more stable long-term profitability.

1.4 Objectives

This project has three primary objectives that are meant to cover both the academic and practical sides of using RL in the complex environment of stock trading:

1. To create an RL model that combines feature enhancement with dynamic reward shaping in a manner that works well together :

This goal tries to use the best parts of both feature improvement (to make the input data more accurate and useful) and reward shaping (to make sure that the model's learning goals are in line with long-term strategies for making a profit and reducing risk). The goal of the combination is to make the model more responsive to the market, which will help it make better decisions in real-world trading situations. Practically, Putting this kind of combined model into action could completely change the way trades do their jobs by letting them make decisions in more complex and flexible ways. This would also give financial analysts better tools for managing risk while making the most profit possible.

2. To see how well this combined method works compared to separate RL models that focus on either enhancing features or adjusting rewards:

This study wants to show that a combined strategy is better at improving the model's general performance and its ability to adapt to changing market conditions by comparing the integrated model with traditional approaches. To find out what the real benefits of the integrated method are, the evaluation will focus on metrics like total return, Sharpe Ratio, and max drawdown. By testing this integrated method, we can see for ourselves that it works better than others. This could lead to the use of similar models in professional trading settings and decision-support systems in financial institutions.

3. To test how robustness and scalable the model is in a range of market situations and trading benchmarks:

The study uses 10 years of past data from Goldman Sachs to test how robust the model is. 95% of the data is used for training, and the other 5% is used for testing. This method lets us carefully look at how the model worked over a long time, taking into account a variety of market conditions that can be found in the past data. Even though the testing doesn't include real-time market conditions or different market environments, the historical data is a good way to make sure that the model works well and can be changed within the time frame that was tried. This setup is meant to show that the model can be useful for traders and financial experts by showing that it can consistently and reliably do what it's supposed to do based on past trends.

1.5 Scope and Delimitation

The project looks into a lot of different areas of computational finance and machine learning, experimenting under specific delimitation (see Table 1: Delimitation of the project) with a focus on a few main scope below:

- **Customised Development of RL Environments:**
  Using resources like the AnyTrading GYM, the project will create an environment that mimics how markets work in the real world. However, the environment needs to be customised in order to fit with different feature enhancement and reward shaping. This environment will be used to test the RL model because it is a practical setting for testing and reproducing.

- **Feature Enhancement & Reward Shaping Techniques**:
  The research will concentrate on identifying and integrating innovative input features and reward structures into RL models. These features will reflect both market data and behavioral factors influencing trader behaviors. The enhancements will focus on advanced data representations such as technical indicators, sentiment analysis, and behavioral biases quantification.

- **Methodology Validation Across Different Stocks**:
  Instead of using a single model across various markets, the research will apply the developed methodologies (feature enhancement and reward shaping) to train separate models for different stocks. This will test the adaptability and effectiveness of the methodologies across various economic sectors and market environments.

- **Performance Evaluation Metrics**:
  Performance metrics such as the Sharpe ratio, the maximum drawdown, and the total profitability will be used to do thorough testing. These metrics will make it easier to see how well the integrated method works in real trading situations, showing how it changes the results of trades.

Delimitation	**Description**
Single Market Focus	Pay attention to just one stock in that market at one time. This limit is meant to keep things manageable and give enough information for a thorough study, while still

	allowing for the possibility that each stock will have its own unique pattern.
Temporal Scope	Up to a certain point in time, use market data from the past. The project doesn't include forecasting or real-time.
Model Complexity	As the project works to improve feature sets and reward structure, it will keep the balance between making the model more complicated and easier to understand. Therefore, we are using models from SB3.
Algorithmic Trading	Focus on trading that is done automatically and based on algorithms through RL. Do not include any intervention from humans.

Table 1: Delimitation of the project

In its conclusion, this study wants to make important theoretical and practical additions to how reinforcement learning can be used in stock trading. The project's goal is to come up with strong trading techniques that work well in both theory and the real world of financial markets by looking into new approaches to improve features and define rewards. If this model is put into reality and evaluated well, it could lead to improvements in automated trading, which would be good for countless individuals in the financial world.

CHAPTER 2. LITERATURE REVIEW

2.1 Reinforcement Learning Literature Review

Reinforcement Learning (RL) has gone through transformative changes that have made it much more useful in many areas, especially when making hard decisions when there is uncertainty.

One of the most important steps forward in RL is the creation of core algorithms that most RL applications today. Some of these are Temporal-Difference (TD) learning, Dynamic Programming, and Monte Carlo methods. There is a different way to learn the best strategies with each of these approaches: Dynamic Programming is a complete framework for solving problems where a full model of the environment is available. Monte Carlo methods are used for problems where the environment are unknown and must be estimated from experience. TD learning combines the sampling of Monte Carlo with the bootstrapping of Dynamic Programming to make a new approach that works in both situations. [7]

Deep Reinforcement Learning (DRL), which combines deep learning methods with reinforcement learning, has been one of the most interesting developments. DRL has made it possible for RL to be used in areas with large state spaces, like robots and video games, where other traditional RL methods would not be effective. This was powerfully shown by DeepMind's AlphaGo, which not only beat human experts at the difficult board game Go but also did so by learning new and unusual strategies. [8].

Another big step forward is policy optimization methods like Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO). By finding the best balance between exploration of new policy and exploitation of known profitable behaviours, these algorithms have made policy learning in RL faster and more stable, especially in continuous action spaces. [9].

RL has also been expanded to multi-agents environments, which has led to new applications in a wide range of fields, from cooperative robots to competitive settings like electronic sports. Not only have these changes pushed the limits of what can be done with autonomous agents, but they have also given us more information about how learning and decision-making work in complicated, interactive settings [10].

Even with these improvements, RL still has a long way to go. One of the biggest problems is computational demands of training models, and it's not always easy to figure out what the learned policies mean. Still, ongoing study into making RL models more efficient, scalable, and easy to understand keeps expanding the range of situations in which they can be applied.

2.2 Reinforcement Learning in Stocks Trading

Reinforcement Learning (RL) has started to become a transformative force in the way people trade stocks. It promises to create automated trading methods that are both flexible and very high efficiency. RL is different from traditional models because it helps to make decisions by interacting directly with market data and constantly adapting based on new information to improve trading actions.

Deep Reinforcement Learning (DRL), which combines both RL and deep neural networks, has been at the forefront of this innovation. DRL's ability to change trading positions instantly in response to changes in the market has consistently shown that it is better than traditional trading algorithms. The advancements mentioned above give DRL models an edge in markets that change quickly. This is shown by studies that show they can do better than normal trading systems[11].

DRL has also made a lot of progress in the area of portfolio management. Here, RL agents have been taught how to handle portfolios so that they get the best returns while also keeping risk levels low. This new way of managing portfolios improves old ones like mean-variance optimization by changing with the market in real time. This creates a dynamic risk-return trade-off that is important for today's financial markets [12].

In high-frequency trade (HFT), where speed and accuracy are very important, RL has also been useful. RL-based trading bots have shown they can carry out orders in milliseconds while processing huge amounts of data to take advantage of quick market chances. This quick decision-making is very important in HFT settings, where dealing by humans alone can't compete [13].

The way traders handle risk has changed since RL added more advanced risk assessment methods, such as Conditional Value at Risk (CVaR). RL models that include CVaR have been created to focus on both making the most profit and meticulous loss control. This helps traders come up with strategies that are both aggresive and smart [14].

All of these different applications show how RL has the ability to impact stock trading. However, there are still problems, such as the fact that model decision-making processes are not always clear and they often require a lot of computing power. As the field moves forward, study is being done to make RL applications in finance more transparent, efficient, and scalable.

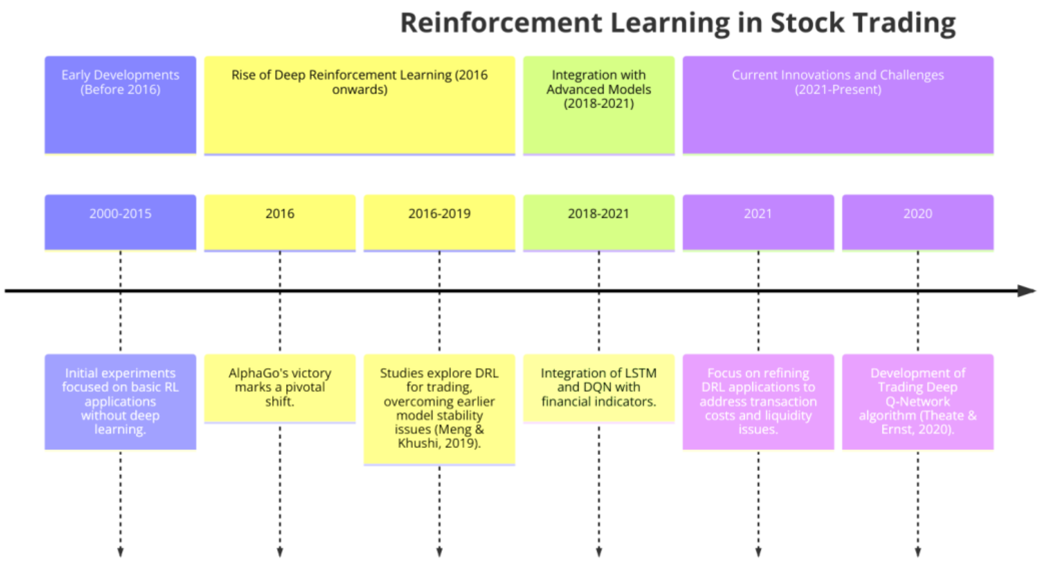

Reinforcement Learning in Stock Trading
Early Developments (Before 2016)
Rise of Deep Reinforcement Learning (2016 onwards)
Integration with Advanced Models (2018-2021)
Current Innovations and Challenges (2021-Present)
2000-2015
2016
2016-2019
2018-2021
2021
2020
Initial experiments focused on basic RL applications without deep learning.
AlphaGo's victory marks a pivotal shift.
Studies explore DRL for trading, overcoming earlier model stability issues (Meng & Khushi, 2019).
Integration of LSTM and DQN with financial indicators.
Focus on refining DRL applications to address transaction costs and liquidity issues.
Development of Trading Deep Q-Network algorithm (Theate & Ernst, 2020).

Figure 3: Literature Timeline of Key development of RL

2.3 Review of Feature Enhancement Techniques

In RL for stock trading, the quality and effectiveness of the input features have a big impact on how well the learning algorithm works. So, feature enhancement is a key part of making the data that passes into the RL model better so that it can pick up on important market signs and patterns. Good feature engineering not only makes the model better at predicting the future, it also makes it better at making profitable trades when market conditions change.

There are different ways to enhance features in financial markets that have been published about. Traders have used moving averages, price momentum indicators, volatility indices, and other statistical tools in the past to predict how the market will move. One popular method is to use these tools together. As technology has improved recently, machine learning methods have been added to dynamically choose and improve these features. Researchers like Bajpai (2021) [15] have highlighted how important it is to carefully choose features that strongly correlate with fluctuations in the market. This helps RL models make better choices.

Also, research like Yang (2020) [16] shows that ensemble methods are useful for feature selection. In these methods, more than one algorithm is used to figure out how useful each feature is. This method not only makes the feature set more robust, but it also makes sure that the model doesn't rely too much on any one market indicator. This lowers the risk of overfitting and makes the trading strategy more useful in different situations.

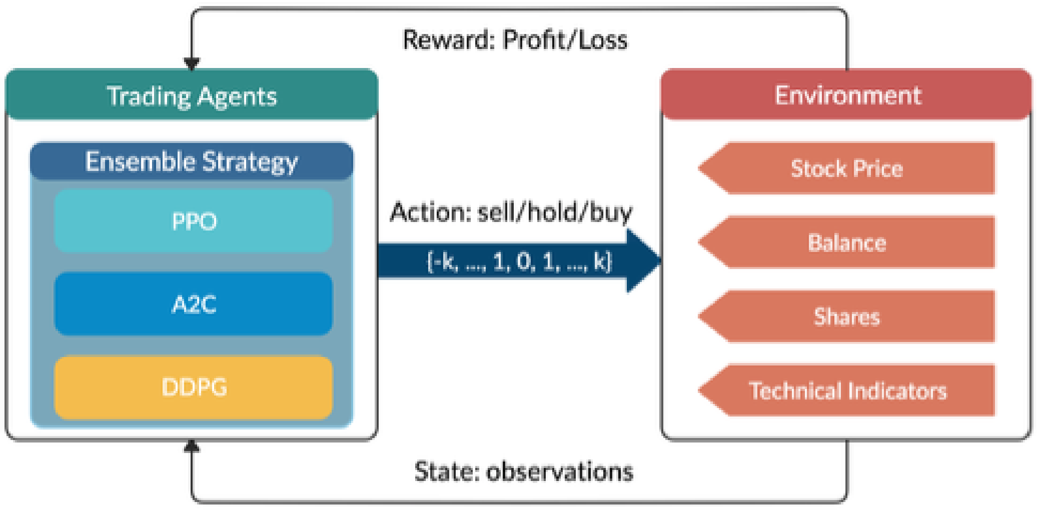


Figure 4: RL framework using ensemble strategies

- **Ensemble Methods**: In feature selection, ensemble methods use more than one learning algorithm to get better prediction performance than any of the individual learning algorithms could provide." For example, combining results from different models lowers the chance of wrong estimates caused by noise in a single model.

2.4 Review of Reward Shaping Techniques

When trading stocks, how the reward function is set up can have a big impact on how risky and profitable a trading plan is. When reward shaping is done right, it helps the RL agent choose tactics that maximise returns while minimising risks.

The research on reward shaping in financial RL shows a number of different approaches that are meant to make sure that the agent's actions lead to profitable and long-lasting trading results. In the 2019 paper [17] talks about how risk management can be added to reward functions so that the model is punished for high-risk trades and rewarded for strategies that find an appropriate balance between risk and return. This method not only makes the trading more profitable, but it also makes sure that it will work for a long time.

Koratamaddi et al. (2021) [18] add to the idea of reward shaping by looking at market sentiment analysis as part of the reward system. By giving the RL agent extra weights for trades that are in line with both technical indicators and market sentiment, their method lets the RL agent use both quantitative data and qualitative market insights, which can help them make better trading choices that could potentially make more profit.

2.5 Identification of Research Gap

The current research literature talks a lot about feature enhancement and reward shaping in RL applied to stock trading, but there aren't many studies that look at these two important parts together in one framework. Most studies try to find the best ways to improve either the input features or the reward systems on their own, so they might miss the benefits that could come from looking at the whole picture. This could be because most of the works are focused on a certain area of study.

This project fills in that gap by creating an RL model that uses both advanced techniques for improving features and advanced strategies for defining rewards. By adding these parts, the model should not only do better in short-term trades, but also be better able to adapt to changes in the market over time. This will set a new standard for how RL can be used in stock trading. It is expected that this combined method will use the best parts of both feature enhancement and reward shaping, resulting in better model performance and more stable trading strategies in a wide range of market conditions.

Also, there are needs to be done on how stable and flexible RL models are in different market conditions and stock types. Some studies try RL models in controlled or specific market settings, but not many look at how well these models work in a wide range of market situations. The goal of this project is also to see how flexible the developed RL models are by applying them on different stocks from different industries and checking how well they do during different market phases, such as bull and bear markets and times of high volatility.

CHAPTER 3. PROJECT EXECUTION SCHEDULE

From the project's start in October 2023 to its expected end in May 2024, this part lays out the project timeline. This organised timeline has been very helpful for exploration and application using reinforcement learning (RL) methods in stock trading. It includes steps from learning the basics to putting them into practice and testing them.

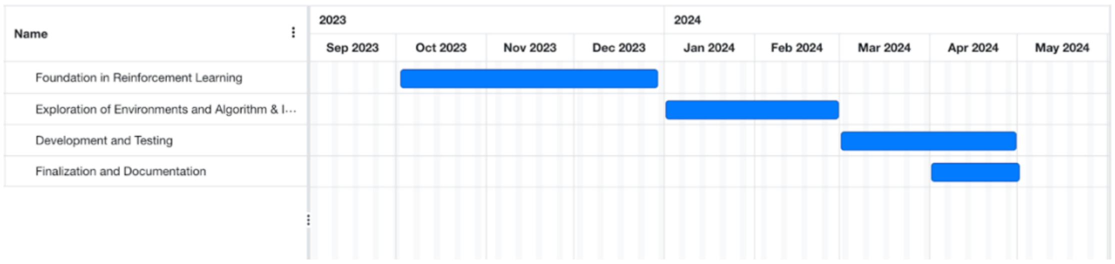


Figure 5: Gantt Chart of each task in big overview

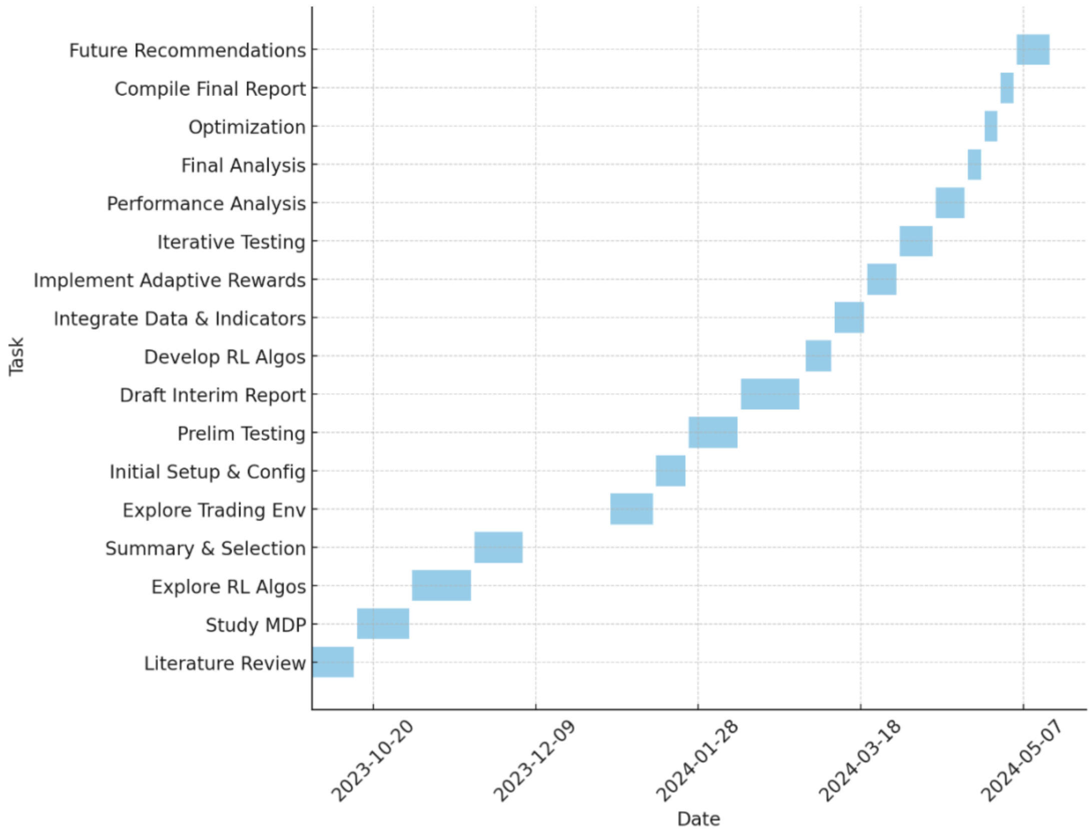


Figure 6: Gantt Chart for each subtask in detailed view

3.1 Foundation in Reinforcement Learning (October - December 2023)

During the first part of the project, a strong base was built in RL by learning its core principles and methods that could be used in stock trading. Some of the things that were studied in detail were Markov Decision Processes (MDP) and basic RL algorithms like SARSA and Q-learning. A wide range of online courses and tutorials gave both practical and academic knowledge that needed to understand how to use RL in the financial markets.

Important books like Sutton and Barto's (2018) Reinforcement Learning: An Introduction explained how complicated RL algorithms are and how they might be used in different areas. This initial part made sure that we had a solid understanding of both the theoretical and practical sides of RL. It also set the stage for more advanced studies and application development.

3.2 Exploration of Environments and Algorithm & Writing Interim Report (January - February 2024)

At this point, the project's main goal changed to finding and testing the best RL environments and methods for stock trading uses. We looked at three main environments: AnyTrading GYM, FinRL, and a stock trading environment from GYM. Each had its own features that made it good for different types of financial trading models.

Deep Q-Networks (DQN), Proximal Policy Optimisation (PPO), Recurrent PPO, A2C, and Trust Region Policy Optimisation (TRPO) were some of the algorithms that were looked at to see which ones would work best for the project.

An interim report was prepared to summarise what was learned during this exploratory phase. It discusses the pros and cons of the environments and methods that were chosen, what we have learned, and what paper we have read. Based on early comments and insights, this report was an important checkpoint that helped shape the project's path.

3.3 Development and Testing (March - April 2024)

Once the project had a good grasp of the relevant environments and methods, it moved on to creating specific feature sets and reward functions. During this step, different combinations were designed and tested to see how they affected the RL model's trading performance.

The main goal was to improve the RL model's ability to make decisions by combining feature improvement and new reward shaping. The real tests were mostly about how flexible and useful the model was in controlled scenarios, focusing on how well it could understand and respond to complicated market data.

3.4 Finalization and Documentation (April - May 2024)

In the last phase, the RL model was improved based on what is gained during development and testing. The model's speed and ability to be scaled up for possible use in the real world were improved by making changes to the way feature enhancement and reward shaping work together.

The whole project was written up in a detailed final report that documented all the steps, from the academic background to the actual results. This paper also laid the groundwork for a final presentation that would show off the unique parts of the project.

3.5 Milestones & Deliverables

Clear milestones have been set to make sure the project moves forward in a planned and measurable way. There are key deliverables that go with each milestone and show that important parts of the study have been finished:

	Milestone	Deliverables
1)	Completion of Preliminary Research (December 2023)	• Literature study document that lists what is known now and where more research is needed. • A list of RL environments and algorithms that can be used for stock trading.
2)	Development of Initial Trading Model (February 2024)	• The chosen algorithms and starting feature sets are used to make a basic RL model. • An interim report
3)	Integration of Advanced Features and Reward Functions (April 2024)	• The RL model has been improved by adding advanced feature enhancement and reward shaping methods. • Full testing that shows how well the model works in different settings.
4)	Final Evaluation and Optimization (May 2024)	• Finalised RL model that was improved after a lot of testing and feedback. • Final report on the project that includes all the study, findings, and analyses. • Presentations

Table 2: Milestones & Deliverables in detailed

These milestones are meant to make sure that each part of the project moves along smoothly and is in line with the research's goals.

CHAPTER 4. METHODOLOGY FOR FEATURE ENHANCEMENT

4.1 Problem Statement

Reinforcement Learning (RL) has worked very well in an environment with abundance of rich data that facilitates agents to get clear feedback right away. But applying RL in stock trading is hard, mostly because of the way the data is structured and how complicated the setting is. Stock trade data is usually just a few lines of price information, which may seem like not much compared to other RL-friendly domains. This small set of data contrasts sharply from the complicated decisions that need to be made in the stock market, where price changes are affected by many factors, such as economic indicators, market sentiment, and global events. [19]

Also, because financial markets are so unpredictable, it's not easy to tell if RL will work in stock trading. RL's usual strengths come from learning by trial and error in environments with lots of data and the same settings. However, in stock trading, where data is sparse, noisy, and subject to abrupt changes. It's hard to tell if an RL model is learning a good trading plan or just getting used to noise because of this uncertainty.

Knowing there’s already challenges with stock data. It’s worth knowing that lack of data diversity led to poor RL performance (can view the Original Feature Function at section 4.2), that is why we are trying to avoid such poor performance via feature enhancement such as integrating extended basic data, technical indicators and candlestick pattern.

4.2 Original Feature Function

The Stocks Trading Environment we use is Anytrading Gym, it is a collection of OpenAI Gym environments for reinforcement learning-based trading algorithms. It comes with an original feature function, which utilises only 2 features (closing price and price difference) from dataframe as input for agent. [20]

Here, the data we are using are 10 years of daily data of Goldman Sachs Company, where 0.95 are used for training, 0.05 are used for testing. Below is the testing result without any feature enhancement :

Performance Metric	Results
Total Profit :	0.80 (means loss 20%, 1.0 being breakeven)
Sharpe Ratio :	-1.61 (perform poorly in terms of risk-adjusted return)
Maximum Drawdown :	-0.35 (max drawdown was 35%)

Problem of original feature function:

The initial set of features in the model, which were just price and its difference, wasn't enough to accurately reflect the complex movements of the stock market. Poor performance metrics showed that this simplicity made it impossible to model and predict market movements correctly with so little data. Using only price and price difference restricted the model's input to very basic market data, which could have caused it to not fit well enough. When a model doesn't pick up on important underlying patterns in more complex market situations, this is called underfitting. This basic set of features also didn't take into account other important things, like volume, volatility, or how different market measures work together. These parts are very important for a more complete market analysis.

Proposed Solution:

To address the problems with the original feature function, it was clear that the input data needed to be enriched. Our first solution is adding all available columns from the dataframe as the input feature. Our hypothesis is, this will make the model better at capturing more complex market patterns, detailed in Section 4.3.

4.3 Version 1 Feature Enhancement: Extended Market Features

Recognizing the need for a broader input dataset, we initially expanded our model's feature set to utilise all available columns from the DataFrame. Our dataset, sourced from Yahoo Finance, consists of 10 years of daily data for Goldman Sachs Group, including the columns: ["Open", "Close", "Low", "High", "Volume", "Adj Close"]. Also, to maximise the information from these columns, we introduced each column as an individual feature, along with the day-over-day differences, resulting in a total of 12 features.

Code snippet:

```
# Calculate differences for price columns (change from previous day)
for column in ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']:
diff = np.insert(np.diff(data[column].to_numpy()), 0, 0)
data[f'{column}_diff'] = diff

# Create an array of signal features, including all processed data
features_columns = [col for col in data.columns]
signal_features = data[features_columns].to_numpy().astype(np.float32)
```

Rationale:
Expanding the feature set to include all available columns from the DataFrame aims to provide a richer dataset for the agent to learn from. By incorporating more diverse inputs such as volume, open, close, high, and low prices, the model can potentially identify more complex patterns that are predictive of future price movements. This approach aligns with the notion in data-driven models that more comprehensive data can lead to better predictive accuracy. [21]

Result on testing data:

Performance Metric	Results
Total Profit :	1.19
Sharpe Ratio :	0.43
Maximum Drawdown :	-0.32

Discussion:

There was a noticeable good result from this feature enhancement, which supported the hypothesis that adding more data features helps the agent learn better. This idea led to the second version of feature enhancement, which added technical indicators to the set of features to make them even more. The goal was to use both historical data and predictive technical analysis tools to help traders make better decisions. (more information in Section 4.4)

4.4 Version 2 Feature Enhancement: Integrating Technical Indicators

4.4.1 Integrating Technical Indicators

Although we already have prior knowledge on technical indicators [22] that would benefit RL agents to learn better from our Chapter 2 : Literature Review, we will still implement this for further validation. This is because we will not only just include technical indicators, but also the additional features from Version 1. These indicators are crucial tools in financial analysis, providing insights into market trends, momentum, and volatility that are not directly observable from raw price data alone. Luckily, technical indicators were easily calculated from a Python library like TA, which we don't have to manually do. Additional features are MACD, EMA, RSI, Stochastic Oscillator, Williams %R, OBV, MFI, ATR, and Bollinger Bands.

Technical Indicators	Description	Usefulness to RL agent
MACD	Measures the momentum by subtracting two moving averages.	Helps in identifying trend reversals and momentum, guiding entry and exit points.
EMA	Provides a moving average weighted more towards recent prices.	Useful for tracking price trends more closely, allowing for timely reactions to price changes.
Stochastic Oscillator	Momentum indicator comparing a particular closing price of a security to a range of its prices over a certain period of time.	Useful for predicting price turning points by comparing the closing price to its price range.
	Remaining part can be found in the appendix part	

Table 3: Technical Indicators Details

<u>Code snippet:</u>

```
data['MACD'] = ta.trend.MACD(data['Close']).macd()
# Momentum indicators
data['RSI'] = ta.momentum.RSIIndicator(data['Close']).rsi()
# Volume indicators
data['Volume_OBV']      =      ta.volume.OnBalanceVolumeIndicator(data['Close'],
data['Volume']).on_balance_volume()
# Volatility indicators
data['ATR']    =    ta.volatility.AverageTrueRange(data['High'],    data['Low'],
data['Close']).average_true_range()
# Other technical indicators are added in in similar manner
```

<u>Result on testing data:</u>

Performance Metric	Results
Total Profit :	1.16
Sharpe Ratio :	0.52
Maximum Drawdown :	-0.32

<u>Observation and Problem Identification:</u>

The results weren't showing improvement that was expected, which could mean that the model wasn't fitting well enough, since both training and validation losses were high at the same time. This shows that our model might not be able to learn and use the complex patterns in the data well, even though we have a lot of data to work with. This is a common sign that the model architecture might need to be changed.

4.4.2 Modification of Model Architect

We experiment with deeper network designs to deal with possible underfitting. For example, we try adding more layers to the network as well as more neurons per layer. The objective was to find the best configuration that strikes a balance between complexity and speed, with the main goal of making our trading model better at capturing and utilising the rich features offered by the large collection of technical indicators. Keep in mind that the original model's design is 64*64.

Model Architect	Rationale	Total Profit	Sharpe Ratio	Max Drawdown
64*64*64	add additional layer while keeping the neuron count consistent, deepen the model	1.31	0.83	-0.31
256*256	Increases the neuron count in each layer, testing the impact of a wider network.	1.07	0.21	-0.39
128*128*128	balanced approach with more neurons per layer than the original and an additional layer, providing both depth and width	1.26	0.71	-0.33

Table 4: Testing result on different model architecture

Several reasons led to the choice of the 64x64x64 model architect:

- Balanced Complexity: This setup takes a balanced approach by adding only one more layer and not greatly increasing the number of neurons in each layer. It makes it easier for the model to learn from more complicated patterns without making it too hard to run or too likely to overfit.

- Better Learning Capability: The third layer in this design lets you extract and change features more deeply, which is very important when working with a set of features with many different types, like the one we got from our technical indicators. This level of detail is especially helpful for showing how the different signs are connected and depend on each other.

- Resource Efficiency: This setup uses fewer resources than the bigger 128x128x128 setup, so the extra computing power is worth it because the model works better.

4.5 Version 3 Feature Enhancement: Integrating Candlestick Pattern

After the integrating of technical indicators like Moving Averages, RSI, MACD, and many others, we can observe that there's improvement in the model to successfully capture the market dynamics. These indicators are likely to provide the model with more understanding with trends, momentum and volume on historical data.

However, there's one problem with technical indicators, which is they often lag the market because they are calculated from past price data. Such delay might result in missed opportunities to enter and exit. Therefore, we bring in the topic of candlestick pattern. Our hypothesis is, such info will significantly increase the model ability to predict the market, and hence improve the agent profitability.

Rationale :

People love candlestick patterns because they can show us how the market is performing and how prices are moving right now. Through certain patterns, they can show bullish or bearish reversals or continuations that aren't always clear from pure technical indicators. By adding candlestick patterns, the model might be able to make better decisions by responding to changes in market mood in real time and changing its strategies to match.

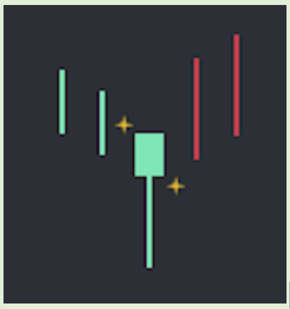
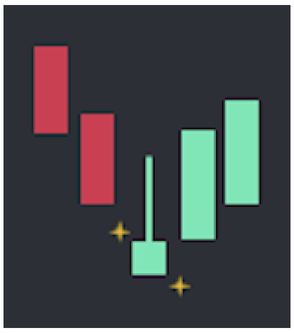
Name	Candlestick Pattern	Description
Hammer	 [23]	A bullish reversal pattern that occurs during a downtrend. The long lower shadow indicates that sellers drove prices down during the session, but buyers were able to push the prices back up close to the open.
Inverted Hammer		A variation of the hammer and can also be a bullish reversal pattern. It shows that buyers attempted to push the price up, but sellers brought it back down, yet the closing price was still higher than the opening price.
	Remaining part can be found in the appendix part	

Table 5: Details of Candlestick Pattern

Code snippet:

The code will scan through the columns of Open, Close, High and Low. There such pattern is found, it will assign to value 1, else, will be 0.

```
data['Engulfing']                                                    =
talib.CDLENGULFING(data['Open'],data['Close'],data['High'],data['Low'])
data['Hammer']                                                       =
talib.CDLHAMMER(data['Open'],data['Close'],data['High'],data['Low'])
# Other candlestick pattern are added in in similar manner
```

Result on testing data:

Performance Metric	Results
Total Profit :	1.30
Sharpe Ratio :	1.11
Maximum Drawdown :	-0.15

Discussion:

A big improvement in performance metrics was not seen as was expected in the hypothesis. The results on testing data remained very similar to Version2 (technical indicators). There might be several reasons why candlestick pattern is not improving agent significantly as expected.

First and foremost, the prediction of the candlestick pattern is not necessarily true. We need to understand candlestick patterns found by humans based on the psychology of buying and selling. It might be true that they let us know the market in real time, but not guarantee how prices will move in the future. There is a chance that predictions will not come true, which can make it harder for these patterns to regularly improve trading outcomes.

Furthermore, candlestick pattern data is usually sparse; not every time frame shows a clear pattern. This lack of pattern leads to feature sets of candlestick that are mostly zeros, which can lessen the effect of useful data to the system. From the view of the agent, all input features are the same, they wouldn’t know this list of numbers represent candlestick patterns or closing price.

Proposed Solution:

Despite the challenges observed with the integration of candlestick patterns, the potential for Version 3 feature enhancement in enhancing RL models for stock trading remains substantial as it has the most numbers of features as input to the agent. To fully realise this potential, it may be beneficial to shift focus towards refining the model's reward function—this approach leads us into the next phase of our research, detailed in Chapter 5: Reward Shaping.

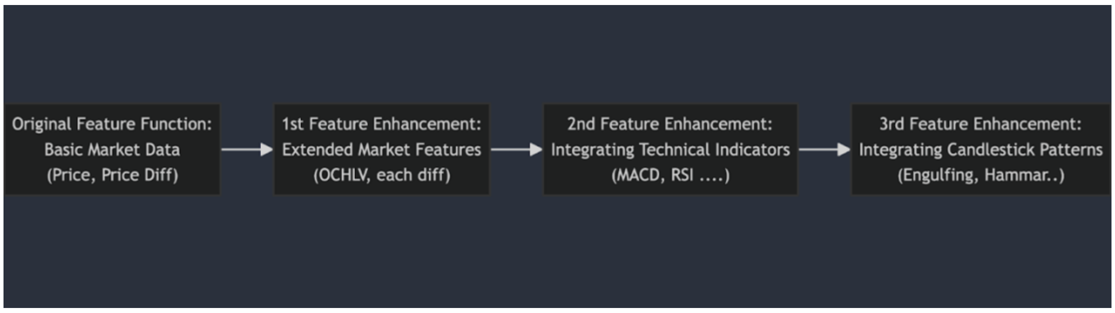

Original Feature Function: Basic Market Data (Price, Price Diff)
1st Feature Enhancement: Extended Market Features (OCHLV, each diff)
2nd Feature Enhancement: Integrating Technical Indicators (MACD, RSI ....)
3rd Feature Enhancement: Integrating Candlestick Patterns (Engulfing, Hammar..)

Figure 7: Feature Enhancement Overview

CHAPTER 5. METHODOLOGY FOR REWARD SHAPING

5.1 Problem Statement

In reinforcement learning (RL), reward shaping means changing the way rewards are given to help the agent behave in the way towards a long term goal. While this method has been very useful in many other areas, it is hard to use it to shape rewards for stock trading environment because of the nature of the financial market.

Characteristic of rewards in stocks trading including:

Reward Type	Description
Delayed Reward	It is true that other RL applications will have delayed reward, but the delay rewards are especially significant and have big effects towards long term goal. Financial decision often involve the forecasting of long term outcome based on current action. This can make it much harder for RL agent to learn because reward are not instantly visible after actions are taken. [24]
Sparse Reward	Trading opportunities that are profitable are rare, but they also appear under certain conditions that may not happen again for a long time. This is different from situations like video games or navigation jobs, where the agent can get rewards (or punishments) more quickly and often based on what they do.
Noisy Reward	Financial markets are naturally noisy due to their sensitivity to many factors such as economic indicator, global events. Therefore, agent might receive reward from not correctly predicting, not just the noise from external factors. This level of noise in the reward signal is generally higher and more complex in stock trading than in domains with more controlled environments.

Table 6: Characteristic of rewards

It’s clear to see that RL performance heavily based on good reward function design, so that model can get quality reward and come out with optimum policy for given input. For example, can view our experiment in section 5.6 to see the importance of well reward shaping. Therefore, our methodology focus on improving the reward function such as loss aversion, risk handling and avoid overconfidence.

5.2 Original Reward Function

The original reward function in stock trading is direct reward and penalty based on the agent's action that leads to price difference of current trade and previous trade.

Code snippet:

```
price_diff = current_price - last_trade_price
step_reward += price_diff
```

The function gives rewards based on whether trading activities made money or lost money. The agent finds the difference between the current price and the last trade price when it chooses to close a position. If the agent was long, the return is positive if the price has gone up since the last trade (they are making money because stock prices are going up) and negative if the price has gone down (they are losing money because stock prices are going down).

Problem of original reward function:

The main problem with the original reward function is that it is too simple, which makes it much less useful. This function only cares about short-term gains from trading, no longer financial goals like long-term growth, risk management, or capital preservation. It treats all gains and loses the same and doesn't take into account the risk of each trade, which could cause high-risk strategies. Also, the function doesn't have any ways to change the rewards based on how volatile an asset is or how exposed the whole portfolio is. It also doesn't promote financial trading strategy. These problems show that we need a more complex reward function to help the agent trade in a way that is lasting and profitable.

The first thing we try is to fix the fact that there aren't any multi-objective factors by giving the reward and the penalty different weights, not just price difference. (More information in part 5.3)

5.3 Version 1 Reward Shaping: Behavioral Reinforcement Reward

5.3.1 Static Behavioral Reinforcement Reward

The most important change here is that reward and penalty are treated differently: rewards for profitable trades stay at their original value, but penalties for losses are multiplied by 1.2. This means that losses have a bigger effect on the person than gains of the same size.

Rationale:

This is aligning with a principle of behavioural finance called loss aversion, which says that losses are seen as worse than gains. Therefore, the RL agent will avoid losses rather than pursuing equivalent winnings. Our hypothesis is that the agent can come up with a trading plan that prioritise capital preservation first, which is very important in a volatile financial market.

Code snippet:

```
price_diff = current_price - last_trade_price
step_reward += price_diff
If step_reward < 0:
    Step_reward *= 1.2
Else:
    Step_reward *=1
```

Result on testing data:

Performance Metric	Results
Total Profit :	1.47
Sharpe Ratio :	1.06
Maximum Drawdown :	-0.35

Discussion

By using different weights for penalty and reward, the goal is to give the agent a multi-objective reward system. The results show that the model's ability got significantly better. One visible problem with this reward shaping, though, is that the weight of penalty stays the same and doesn't change as market conditions change. The fixed weight doesn't take into account changes in the market volatility, which can affect how well the risk rating built into the reward function works.

So, to make this reward function even better, we add a dynamic weight factor that changes based on how volatile the market is. The goal of this method is to make the penalty factor more adaptable to the current trading situation.

5.3.2 Dynamic Behavioral Reinforcement Reward

Code snippet:

The penalty factor is adjusted based on the normalised volatility. It's calculated by the standard deviation of the prices over a specified window and divides it by the average price within the same window. When the market is more unstable, the penalty for losses goes up. This makes agent less likely to take risks when things are unclear. On the other hand, when the market is stable and volatility is low, the penalty is closer to the base level (1.0), which means that trading activities are less risky.

```
def calculate_normalized_volatility(prices, window=20):
    if len(prices) < window:
        return 1.0 # Default to no adjustment if not enough data
    recent_prices = prices[-window:]
    avg_price = np.mean(recent_prices)
    volatility = np.std(recent_prices) / avg_price # Normalized volatility as a
percentage of the average price
    return volatility
```

```
if step_reward < 0: # Apply a dynamic penalty scaling factor based on normalized
volatility
    penalty_factor  =  1.0  +  calculate_normalized_volatility(prices)  #  Scales
linearly with volatility
    step_reward *= penalty_factor
else: step_reward *= 1 # No change for gains
```

Results :

Performance Metric	Results
Total Profit :	1.45
Sharpe Ratio :	1.35
Maximum Drawdown :	-0.21

Discussion :

Using a dynamic penalty weight has shown the similar performance improvements as the old static method. However, it is more reliable as the risk is lower. This can be seen from the performance metric, where the Sharpe Ratio of 1.35 is higher than the previous static way of 1.06. This improvement makes sure that the model's risk assessments are more in line with how the market is doing right now. This makes the trading strategy more flexible and might be useful for long-lasting in a variety of situations. We will now look at some more reward-shaping methods that focus on risk management. (detailed in section 5.4)

5.4 Version 2 Reward Shaping: Risk Handling via thresholding

We will now turn our attention to risk management, which is another important part of trading stocks. It's important to handle risks well so that you can make money in the long run and avoid big losses. In order to do this, we introduce a very simple reward shaping strategy that is meant to directly encourage our trading methods to keep risk under control, which is rewarding agent if the total profit stays above a certain threshold.

Rationale:

The idea behind this method is to directly give a small reward to the agent for keeping the total profit above a certain level, in this case a profit of 1. Our hypothesis is that, giving the model a reward when the total profit is above a certain level and a penalty when it falls below that level encourages it to come up with and stick to trade strategies that not only make money but also handle risks well.

Code snippet:

```
step_reward = 0
if self._total_profit >= 1:
    step_reward += 1
else:
```

```
            step_reward -= 0.5
        return step_reward
```

Result :

Performance Metric	Results
Total Profit :	0.91
Sharpe Ratio :	-0.36
Maximum Drawdown :	-0.53

Problem :

However, the overall performance went down contrary to our hypothesis. This result shows that the goal of improving risk management through direct rewards is clear, but the way it was put into action may be too simple. There are 2 possible problems, the simplistic nature of the profit threshold and fixed nature of reward and penalty.

Propose Solution:

To address these problems, several attempts for modification to make threshold and penalty/reward adjustments more dynamic, below is one of the examples: which is using the 20 days average as threshold.

```
        step_reward = 0
        moving_average_threshold = np.mean(self.prices[-20:]) # Average of the last
    20 profits
        current_profit = self.profits[-1]
        if current_profit >= moving_average_threshold:
            step_reward += 1 # Reward for exceeding the moving average of profits
        else: step_reward -= 0.5 # Penalty for not meeting the moving average return
    step_reward
```

The performance of the model in such reward shaping has not gotten much better despite many changes were tried, such as changing the reward and penalty to dynamic value and trying out different threshold settings. These results show that it is still hard to make a threshold-based reward system work well with the trade strategy of the model.

Analysis of performance issue:

This persistent bad performance suggests that a reward system based on thresholds might not work well. Even when making it into dynamic value, such a system seems to fail to capture the stock's pattern.

First reason might be, a threshold system is not able to respond quickly to sudden market changes, resulting in strategies that don't perform well in downturns or fail capturing sudden upturns. Second reason, simplifying a complex environment into binary cases(above or below), may not be a good approach to reflect the true situation.

These reasons as well as the performance result show the limitation of threshold based approach for effective risk management in reward shaping.

Due to the issues we saw with the threshold-based reward shaping method, we looked at risk management from a different angle: the trading behaviour of the RL agent. In section 5.5, we'll talk about a common problem of trading strategy: Overconfidence.

5.5 Version 3 Reward Shaping: Risk Handling via Avoid Overconfidence

Managing risk with a threshold-based method didn't work very well in last section. So, we're now looking at a different way to do it by looking at RL model behaviour. In particular, we talk about the problem of overtrading, which happens when agents are too sure of themselves. Overtrading can make a trading strategy much less effective, causing both higher transaction costs and the chance of losses.

Rationale:

This new approach of risk handling focuses on managing how often the trades happen. The goal of this approach is to stop agent from trading too much by giving penalty. We want to improve the model's general risk management by reducing overtrading. The hypothesis here is the agent will reduce the possibility of losses when only focusing on high quality trades.

Code snippet:

Our solution is to include a time window (10) where the number of trades (3) is limited. This method is meant to keep the agent from making too many trades in too little time, which should lead to more deliberate and well-thought-out trading choices.

```
def reward_function_5(self, action, trade_window=10, max_trades_allowed=3):
    step_reward = 0
    recent_trades = self.recent_trades[-trade_window:]
    if len(recent_trades) > max_trades_allowed:
        step_reward -= 0.5
```

Result:

Performance Metric	Results
Total Profit :	1.69
Sharpe Ratio :	1.25
Maximum Drawdown :	-0.24

Discussion:

The implementation of focus on behaviour of agent's trading show significant results, where total profit is 1.69 with high Sharpe Ratio 1.25. By setting a limit on the number of allowable trades within a specific period, the agent becomes more selective about the trades it executes, which helps to filter out impulsive decisions based on fleeting market noise. The improved performance confirms our hypothesis.

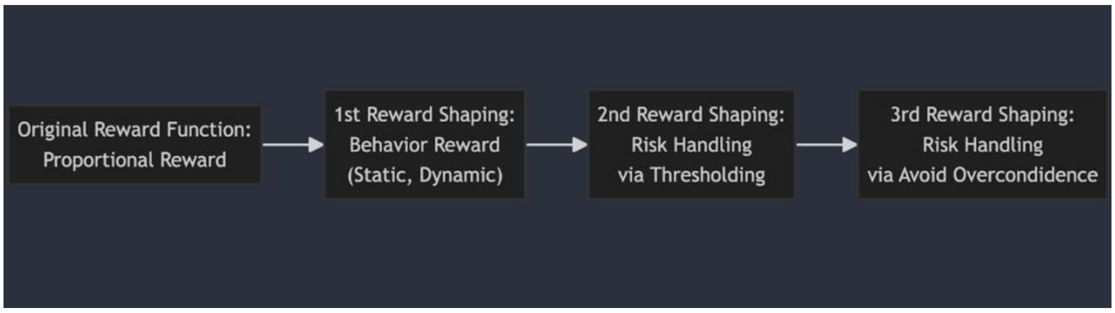

Original Reward Function: Proportional Reward
1st Reward Shaping: Behavior Reward (Static, Dynamic)
2nd Reward Shaping: Risk Handling via Thresholding
3rd Reward Shaping: Risk Handling via Avoid Overcondidence

Figure 8: Reward Shaping Overview

5.6 Notable Failed Reward Shaping

On my very first attempt on reward shaping, I designed a very complex reward function based on the financial trend following strategy. Assuming that since financial strategies have proven effective [25], a reward function designed around these theories would also be successful.

However, when it was put into implementation, the agent showed strange behaviours that were not expected. For example, the agent tended to mostly buy, no matter what the market conditions were. Since it's a rather complex reward function, much time and effort was put in to fix such a problem, but still to no avail.

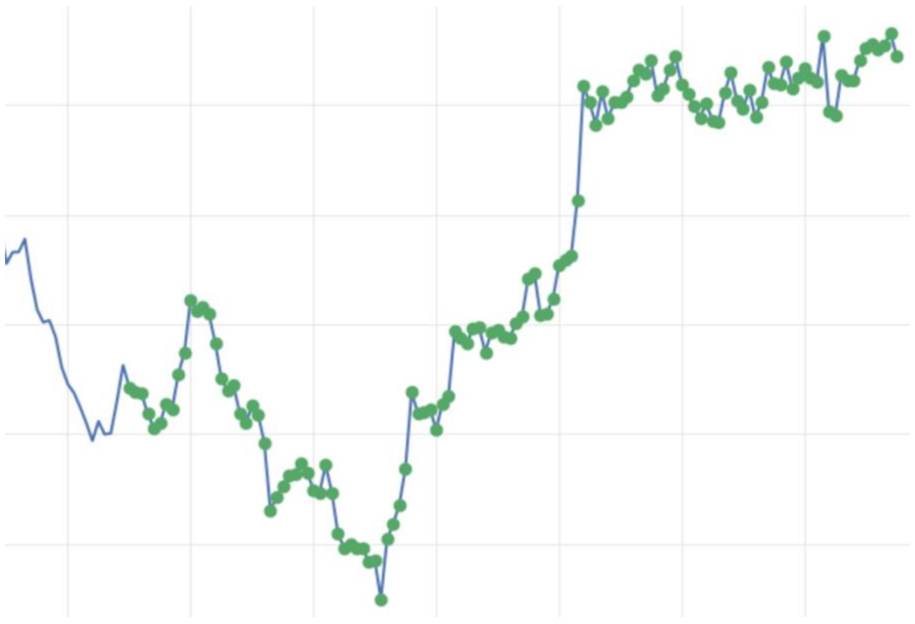


Figure 9: Constant buying on failed reward shaping

Some plausible reasons for such a phenomenon are, reward/penalty are not well designed, so that agent might find it statistically more rewarding over time to engage in buying than penalty. Furthermore, the reward function overemphasis on trend alignment, without considering the broader market context.

The most important thing I learned from this is how important it is to start with simpler reward systems and add more depth over time through iterative improvements. Complex reward systems can be useful, but they can also make it hard to train trading models that are stable and effective. This method ensures that the model's ability grows in a more controlled way, which makes it easier to incorporate more advanced strategies.

CHAPTER 6. RESULTS

6.1 Implementation of experiment

Data Handling

For this project, data from Yahoo Finance was used. It has daily trade data for Goldman Sachs (GS) from 2014 to 2024. Goldman Sachs was picked because it has a big influence in the financial sector and is more volatile than broad market indices like the S&P 500. Because of this volatility of GS, the RL agent gets to learn more effectively since the signal might be stronger. The dataset was carefully split into training (95% of it) and testing sets (5% of it). Strict steps were taken to keep data from getting out and to make sure that testing matches real, unseen market conditions.

Model Choice

PPO was chosen as the preferred algorithm for this project because PPO has advantages in handling environments with high-dimension space. Also, PPO is known for its stability and reliability, as it avoids large policy updates, making it suitable for the nuanced decisions required in stock trading. [26]

Environment

AnyTrading GYM is what our environment is based on, but some settings were customized for different feature methods and reward shaping setups. This makes sure that the environment fits the needs of the project.

Training:

During training, the model interacts with a simulated market and makes trades based on its current policy. It also changes its strategy based on feedback it gets from the custom reward functions. This phase is iterative, which lets the model improve its trading choices over a number of episodes. The model will be saved once training is finished.

Testing:

The model is tested on the testing dataset to see how well it does on new data that it hasn't seen before. Then, total profit, the Sharpe ratio, and maximum drawdown are some of the most important metrics used to evaluate the model. These measures give information about the return on investment, the risk-adjusted return, and the possible losses in bad situations. Not only that, some logs such as training loss, value loss will be considered.

6.2 Benchmark and Performance Graph

Setting a benchmark using simple buy and hold strategies provides a reference for comparing the later enhanced models:

Testing period starting from 19/07/2023 to 29/02/2024.

- **Buy and Hold Strategy on GS on same testing period**:

Total Profit	**1.15**
Max Drawdown	-0.19
Sharpe Ratio	1.11

- **S&P 500 Index on same testing period**:

Total Profit	**1.11**
Max Drawdown	-0.10
Sharpe Ratio	1.56

Below is an overview of the results of feature enhancement.

Feature Enhancement	**Total Profit**	**Max Drawdown**	**Sharpe Ratio**
Extended Market Feature	1.19	-0.32	0.43
Integrating Technical Indicators	1.31	-0.31	0.83
Integrating Candlestick Pattern	1.30	-0.15	1.11

Table 7: Overview of results of feature enhancement

Below is an overview of the results of reward shaping.

Reward Shaping	Total Profit	Max Drawdown	Sharpe Ratio
Behavior Reinforcement Reward	1.47	-0.35	1.06
Risk Handling via Thresholding	0.91	-0.36	-0.53
Risk Handling via Avoid Overconfidence	1.69	-0.24	1.25

Table 8: Overview of results of reward shaping

The integrating of both feature enhancement and reward shaping clearly perform better than both benchmarks of Buy & Hold strategy and S&P500 index in term of total profit and Sharpe ratio, which are the 2 most important thing we care, respectively the ability to profit, and the ability to risk.

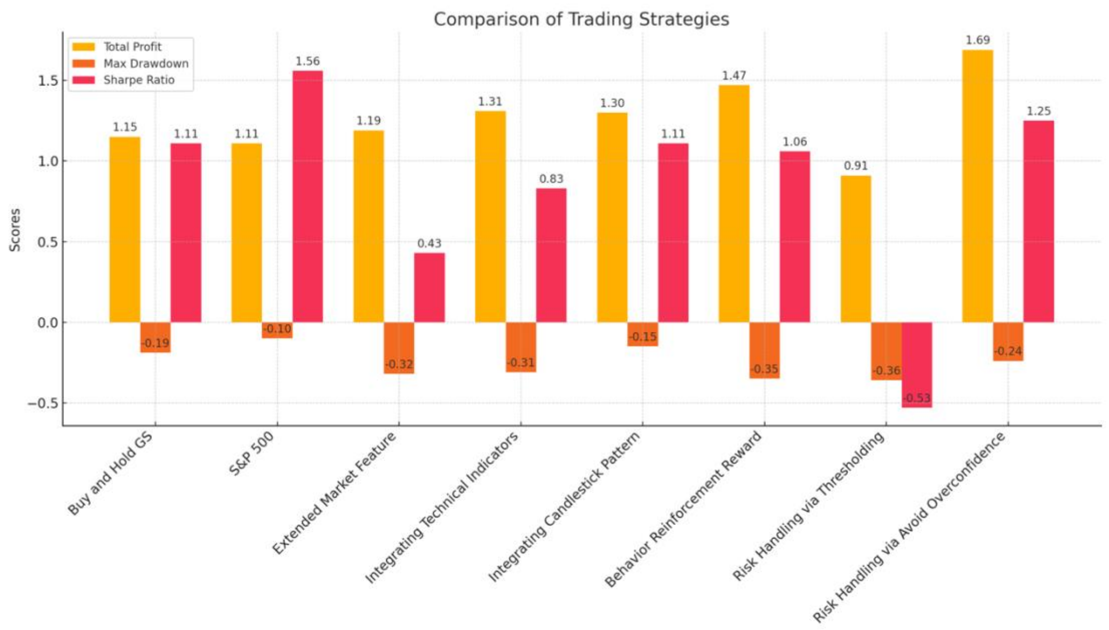


Figure 10: Bar Chart of result of feature enhancement and reward shaping

6.3 Testing Across Different Stocks

6.3.1 Rationale

To show that our methodology is valid and can be used with more than just the original dataset from Goldman Sachs (GS) stock, we tested it on a number of different stocks. This method helps to see if the developed methods, which include both strategies for feature enhancement and reward shaping, work in a variety of market conditions and stock types.

We chose a wide range of stocks to make sure that the model's meet most different scenarios. This group of stocks comes from a range of industries and market sizes. It includes tech giants like Apple (AAPL), blue chip stock like Costco(COST), pharmaceutical leaders like Pfizer (PFE), consumer goods giants like Coca-Cola (KO), and energy companies like ExxonMobil (XOM). This variety makes it easier to evaluate how well the methodology works in a range of industry settings and economic situations.

For each stock, it will undergo the same training and testing process, same data preparation way, same model configuration, same feature enhancement and reward shaping, as well as same performance evaluation method. Where the benchmark to pass is the buy and hold strategy in the same testing period.

6.3.2 Testing Results

Stocks	Benchmark	Enhanced version
AAPL	Total profit : 0.97 Sharpe Ratio : -0.43 Max Drawdown : -0.12	Total profit : 1.45 Sharpe Ratio : 1.07 Max Drawdown : -0.17
PFE	Total profit : 0.74 Sharpe Ratio : -2.25 Max Drawdown : -0.26	Total profit : 0.83 Sharpe Ratio : -2.07 Max Drawdown : -0.25
COST	Total profit : 1.37 Sharpe Ratio : 3.66 Max Drawdown : -0.05	Total profit : 1.33 Sharpe Ratio : 3.55 Max Drawdown : -0.4
KO	Total profit : 0.98 Sharpe Ratio : -0.16 Max Drawdown : -0.12	Total profit : 1.12 Sharpe Ratio : -0.24 Max Drawdown : -0.18
XOM	Total profit : 0.94 Sharpe Ratio : -0.51 Max Drawdown : -0.19	Total profit : 1.41 Sharpe Ratio : 1.03 Max Drawdown : -0.28

Table 9: Testing result across different stocks

The results of the tests were very informative. Overall, the improved models (integrating feature enhancement and reward shaping) did better than the simple "buy and hold" benchmarks. This shows that using advanced RL methods in stock trading strategies can be successful. Notably, stocks like COST didn’t improved a lot in terms of overall profit and Sharpe ratio. This showed that our methods didn’t worked very well in sectors that are known for being stable and not having a lot of room to grow. It’s reasonable as our reward shaping is mostly based on risk handling, rather than chasing for maximum profit.

There is, however, still room for improvement. It might be possible to get even better results by customizing the reward functions and hyperparameters for each stock based on its unique traits and domain knowledge.

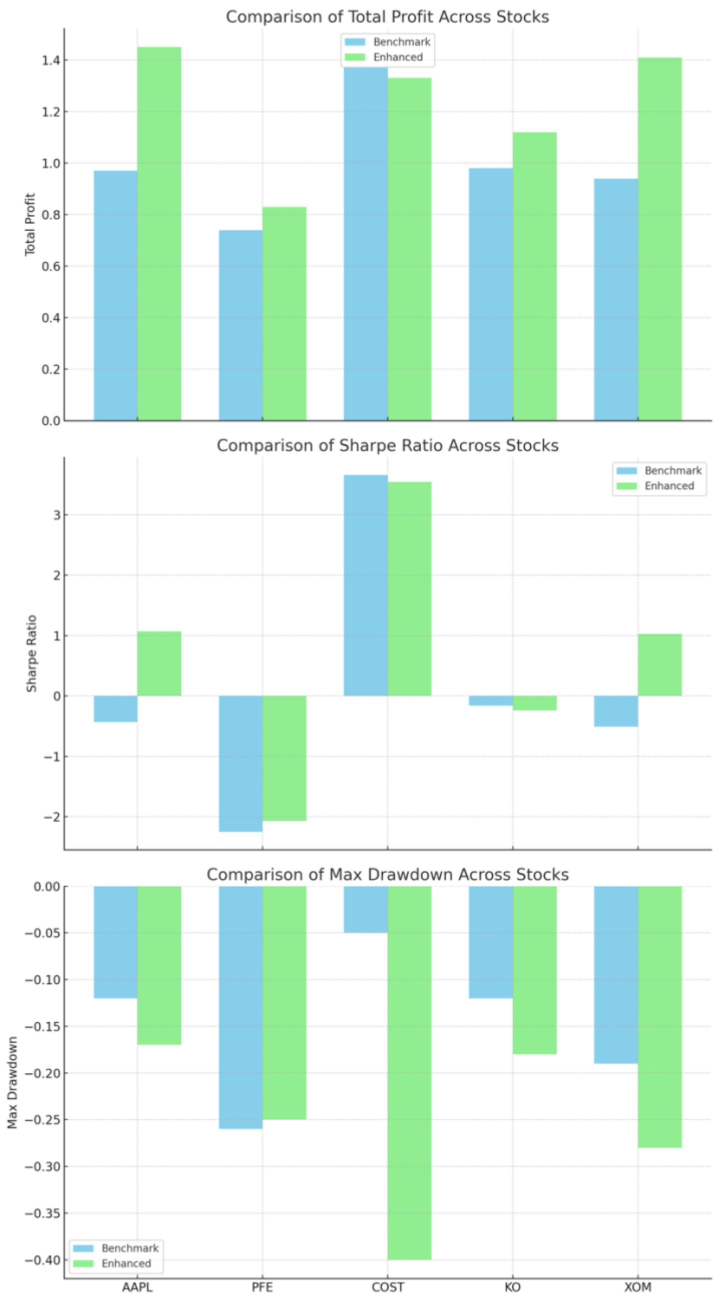


Figure 11: Paired bar charts showing the comparison of benchmark and enhanced results

CHAPTER 7. DISCUSSION

7.1 What I’ve Learned in this project

This project has been a profound journey on Reinforcement Learning. Because of the results and experiences, we now know a lot more about the pros and cons of using advanced machine learning methods in financial markets. Here are some important things I learned from this project:

1. <u>The Importance of Feature Engineering:</u>

One of the most important things I've learned from this project is how important feature engineering is for making good RL models. The model can understand and predict market changes a lot better if it has the right data. Using techniques like moving averages, RSI, and MACD, along with other technical indicators, has shown that complex and thorough data representations help an RL agent make better decisions. But this also showed how thin the line is between adding useful information and making the model too complicated, which could cause it to overfit.

2. <u>The Double-Edged Sword of Reward Shaping:</u>

Reward shaping became a useful way to manipulate the RL agent's trade behaviour to achieve certain goals. It worked to steer the model's strategies by changing the reward function to include penalties for overtrading and rewards for following successful trends. But this process also showed how sensitive the model's success is to the details of how the rewards are set up. It was hard and educational to come up with reward functions that are a good mix between motivation and punishment without causing people to do things they weren't supposed to.

3. <u>Learning from Mistakes:</u>

Some plans and changes didn't work out as planned, but these mistakes taught us a lot. They talked about how important it is to try and change things over and over again. For instance, the trend-following reward strategy that mostly led to buying behaviors taught me how important it is to make models that are both complicated and useful.

4. <u>The Importance of Iterative improvement:</u>

It was very important to test different configurations over and over again and keep improving the model based on comments on how it worked. This method not only helps to improve tactics, but it also helps to comprehend how the model and market work on a deeper level.

7.2 Challenge Met

As we worked on this project, we learned about some problems that come up when try to use reinforcement learning in stock trade. I will briefly mention some challenges I met and how I dealt with them.

One big problem was that model variability, the testing performance changed even with the same methodology. This is a typical problem in reinforcement learning because it is random when it explores and learns. At first, using set random seeds to try to stabilise the results didn't work very well. In order to fix this, we decided to average the results of several runs. This method helped us lessen the impact of chance and gave us a more accurate picture of how well the model worked.

Another big problem was making the reward function work well. When the reward function was first designed, it sometimes caused trading behaviors that weren't meant to happen, so the model's actions weren't always in line with the best trade strategies. We used an iterative process to improve the reward function, making changes over and over and testing the effects of different reward structures. This careful process of making changes helped us create a reward function that works better with the trade results we wanted.

At first, it was hard to test the model's reliability in different market situations because the environment had to be rewrite. We changed our plan to use the model on a wide range of stocks to make the process easier while still making sure the testing was thorough. A simpler way to customise the environment was used in this method, which also showed that the model could work well in a wider range of real-world situations. Testing the model on a wide range of stocks gave us a full picture of how well it worked and how well it could be used in other situations, which proved that our method was sound.

Iterative tuning and the need to train complex models made training very hard in terms of computing. Because tuning was done over and over again and training rounds were long, a lot of computing power was needed. To deal with this, we made our code more efficient and used parallel methods to make computations go faster.

7.3 Future Work

The current project has made a lot of progress in applying reinforcement learning to stock trading. However, due to the time limit, many areas are not yet explore and I believe the results of this study could be further expanded and improved in a number of areas by future research. Here are some important areas to look into further:

First and foremost, integration of alternative data source. News feeds, social media sentiment, and economic indicators are some other data sources that could be useful for future versions of the project. These sources can give the model more background information that could help it make more accurate predictions about how the market will move and respond to events happening in real time, which could lead to better trading choices.

Next, Hybrid models combining RL with ML might be another work. In the future, researchers could also look into hybrid models that combine reinforcement learning with other machine learning or statistical methods, like Bayesian networks or ensemble methods. These combinations could use the best parts of different methods to make trading systems that are stronger and more reliable.

Furthermore, cross market or asset development can be done. Test the model in more than one market and asset class could show how flexible and scalable it is. This could mean testing the model on international stock markets, commodities, or cryptocurrencies and seeing how well it works in various market and governmental settings.

Last but not least, putting the model to use in a real-time trading environment would be a natural next step for this study. In this step, execution slippage, transaction costs, and the rules and regulations of trade would be dealt with. This would give a full test of the model's usefulness in real life.

CHAPTER 8. ETHICS AND PROFESSIONAL CONDUCT

When creating and using advanced trade models that use reinforcement learning (RL), professional ethics and behaviour are very important to make sure that these technologies help the markets without hurting them too much. This part talks about the possible effects on financial markets, the ethical issues that come up when using RL in trading.

8.1 Ethical Consideration

Using RL in financial trading brings up a number of ethical problems, mostly related to how open, accountable, and fair automated systems should be. In traditional trading, choices are often made by people, but in RL systems, choices are made by programmes that learn from a huge amount of data. This makes things more efficient and objective, but it also makes it harder to understand and interpret how choices are made.

- **Algorithmic Transparency**:

The fact that machine learning algorithms are hard to understand is a major ethical issue. This is often called the "black box" problem. During the "Flash Crash" of 2010, for example, algorithmic trade systems were criticised for not being clear, which led to major market problems. It is very important for coders to make these systems as clear as possible so that anyone can understand them. This includes not only writing down and explaining how the algorithms work, but also making sure that regulatory bodies and other stakeeholders can understand why trading choices are made the way they are. Building trust and making sure that these systems are used properly are both helped by more openness.

- **Accountability**:

There must be clear ways for people to be held responsible when automated trading causes price drops or market disruptions. To do this, frameworks must be set up that make it clear who is responsible for the choices made by the RL systems, while also allowing humans to keep an eye on things and step in when necessary. For example, unfair market practices happened when trading algorithms used historical data that held discriminatory pricing strategies. This shows how important it is to keep auditing and updating systems.

- **Fairness**:

It is also very important to make sure that the RL systems don't accidentally lead to unfair market benefits or discriminatory practices. The data sets and algorithms that these systems use need to be constantly checked for errors and updated so that they don't have any biases that could hurt other market players.

- **Influence on Market Liquidity:**

RL trade models can make the market more or less liquid, depending on the case. They can add more liquidity by making trading more active, but they can also take it away quickly during rough times because these models might all respond the same way to certain market signs, which can cause liquidity shortages

8.2 Ethical Guideline Proposal

As shown by the real-life examples and possible market effects talked about, using RL technologies in trading can have unintended effects like market volatility, liquidity crises, and unfair trading practices if they are not handled properly. Because of this, ethical guidelines are needed to make sure that these technologies are created and used in a way that is fair, open, and responsible. Below are some guideline we proposed:

Guideline	Details
Transparency and Explainability	Create and use RL systems that are open about their decision-making methods, data sources, and algorithms. Make sure that models can be interpreted so that users can understand and trust them.
Accountability	Set up robust ways to make sure that people are held accountable for the choices that RL systems make. This includes providing a handful of human oversight and the capability for intervention.
Data Integrity and Privacy	Maintain strict standards for the protection and accuracy of data. Make sure that all the data that is used in trading algorithms follows all privacy rules and morals.
Bias Mitigation	Do audits on a regular basis to find and fix methods that are biassed. Change the training data and models so that they represent fair and honest business practices.
Continuous Monitoring	Set up rules for regularly checking and watching RL systems to make sure they work the way they're supposed to and don't change how they behave over time.

Table 10: Ethical Guideline Proposal

CHAPTER 9. CONCLUSION

9.1 Summary of work

This project started an effort to use reinforcement learning (RL) in stock trading. Its main goal was to find the working trade methodology by feature enhancement and reward shaping.

The project started with an analysis of the initial set of features, which mostly consisted of simple price information. We knew that such a small sample had some problems, so we increased the number of columns used in the input features to include all the available columns from the market data. This made the model's understanding of how markets work much better. Several more improvements, like adding technical markers like MACD, RSI, and Bollinger Bands, made it easier for the model to spot market trends and momentum. This change made it clear that the model design needed to be more complicated in order to handle the more complicated data well. After that, candlestick patterns were added to see if they could give real-time views into how the market thinks, but this had to be done carefully so as not to fill the model with too much noise.

The first reward function was very simple; actions were directly linked to instant price changes. We added a behavioral reinforcement reward to encourage more complex trading behaviors. At first, we used a static weight to punish losses more than wins. Seeing that a static method had some minor problems, we improved it to a dynamic weight that changed based on how volatile the market was. This made the model more sensitive to risk. Then, we move our focus to risk handling, firstly with the threshold method. After many tries and dynamic changes, efforts to manage risk through profit thresholds still didn't work. We changed the approach and dealt with risk by limiting the number of trades to avoid overconfidence problem. This worked well and was easy to put into action. Furthermore, a complicated market trend-following strategy that was meant to look like financial strategies didn't work very well. This shows how important it is to start simple and add more complexity over time when shaping rewards.

To see how robust the methodology we created were, we tested them on a wider range of stocks from different industries. This cross-stock validation showed that both the methods for feature enhancement and reward shaping worked well in a variety of market conditions. This showed how flexible the model is and proved that our approach can be used in most situations.

The results of the project show that RL could be useful in stock trading, but they also show how important it is to be very careful with design and make both feature engineering and award design more complicated over time. We can better handle the complicated nature of the financial markets if we start with simple ideas and build on them over time. This will pave the way for future trade models that are more advanced and reliable.

9.2 Final Thoughts

As I wrap up this Final Year Porject, I'm amazed at how much I've learned about reinforcement learning and how it can be used in stock trading. This journey has been both exciting and hard, pushing me to go beyond my comfort zone and learn more about a complicated subject.

There were times when doubt set in, like when models didn't work as planned or when the markets seemed too complicated to understand. But because of these problems, after tackling them, feel particularly rewarding. They taught me not only how algorithms and the environment work, but also how important it is to be patient and keep going even when things are unclear.

Last but not least, I'm feeling very satisfied and ready to take on new tasks. After working on this project, I have a better idea of both the opportunities and problems that lie ahead in the fields of technology and finance, which will help me make a useful contribution in the future.

CHAPTER 10. REFERENCE

[1] Investopedia, "What's the most expensive stock of all time?" Investopedia. https://www.investopedia.com/ask/answers/081314/whats-the-most-expensive-stock-all-time.asp

[2] Antonio Briola et al., "Deep Reinforcement Learning for Active High Frequency Trading," ArXiv (2021)

[3] Silver, D. et al., 2018, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," Science.

[4] M. V. Otterlo and M. Wiering, "Reinforcement Learning and Markov Decision Processes," in Reinforcement Learning: State-of-the-Art

[5] Tidor-Vlad Pricope, "Deep Reinforcement Learning in Quantitative Algorithmic Trading: A Review," ArXiv

[6] Y.-H. Chang and M.-S. Lee, "Incorporating Markov decision process on genetic algorithms to formulate trading strategies for stock markets,"

[7] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction,"

[8]D. Silver et al., "Mastering the game of Go with deep neural networks and tree search,"

[9] J. Schulman et al., "Proximal Policy Optimization Algorithms,"

[10] R. Lowe et al., "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments,"

[11] Y. Deng, Z. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading,"

[12] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem,"

[13] M. Kearns and Y. Nevmyvaka, "Machine learning for market microstructure and high frequency trading,"

[14] Z. Xiong, X.-Y. Liu, S. Zhong, H. Yang, and A. Walid, "Practical deep reinforcement learning approach for stock trading,"

[15] S. Bajpai, "Application of deep reinforcement learning for Indian stock trading automation,"

[16] Hongyang Yang, Xiao-Yang Liu, Shanli Zhong, and Anwar Elwalid, "Deep reinforcement learning for automated stock trading: an ensemble strategy,"

[17] Konstantinos Saitas Zarkias; "Reinforcement Learning for Financial Trading Using Price Trailing"

[18] Koratamaddi, P., Wadhwani, K., Gupta, M., & Sanjeevi, S. (2021). "Market Sentiment-Aware Deep Reinforcement Learning Approach for Stock Portfolio Allocation."

[19] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita, "Adaptive stock trading strategies with deep reinforcement learning methods,

[20] AminHP, "gym-anytrading," GitHub, [Online]. Available: https://github.com/AminHP/gym-anytrading].

[21] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction

[22] Jordan Ayala, M. García-Torres, José Luis Vázquez Noguera, Francisco Gómez-Vela, and F. Divina, "Technical analysis strategy optimization using a machine learning approach in stock market indices,"

[23] "Mastering Candlestick Patterns for Successful Crypto Trading," Altfins, [Online]. Available: https://altfins.com/knowledge-base/mastering-candlestick-patterns-for-successful-crypto-trading/.

[24] Yiming Liu and Zheng Hu, "The Guiding Role of Reward Based on Phased Goal in Reinforcement Learning,"

[25] Cheng Wang, Patrik Sandås, and P. Beling, "Improving Pairs Trading Strategies via Reinforcement Learning,"

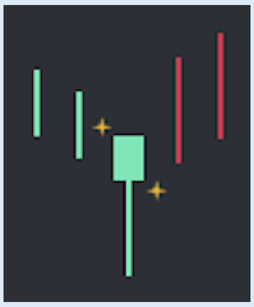
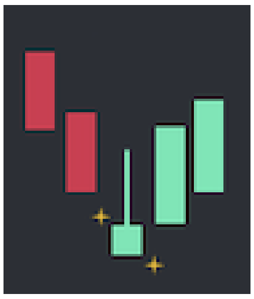
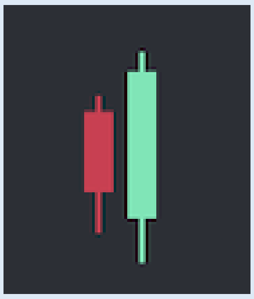
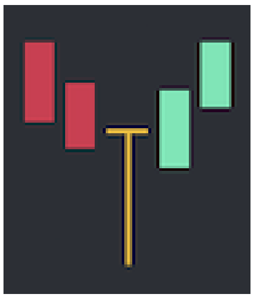
[26] John Schulman, F. Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal Policy Optimization Algorithms,"

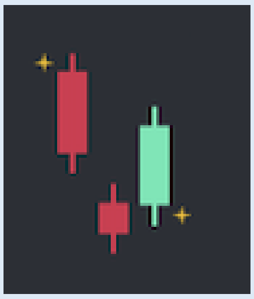
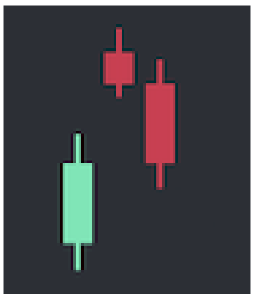
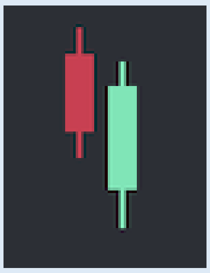
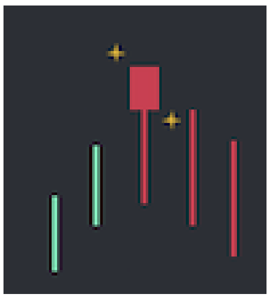
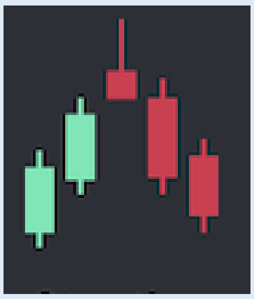
CHAPTER 11. APPENDIX

Technical Indicators	Description	Usefulness to RL agent
MACD	Measures the momentum by subtracting two moving averages.	Helps in identifying trend reversals and momentum, guiding entry and exit points.
EMA	Provides a moving average weighted more towards recent prices.	Useful for tracking price trends more closely, allowing for timely reactions to price changes.
RSI	Oscillator that measures the speed and change of price movements in a range from 0 to 100.	Indicates overbought or oversold conditions, suggesting potential reversal points.
Stochastic Oscillator	Momentum indicator comparing a particular closing price of a security to a range of its prices over a certain period of time.	Useful for predicting price turning points by comparing the closing price to its price range.
Williams %R	Momentum indicator that measures overbought and oversold levels, similar to the Stochastic Oscillator.	Helps determine optimal points for trade entries and exits by identifying market extremes.
OBV	Combines volume and price to show how money may be flowing into or out of a stock.	Indicates bullish or bearish moves; useful for confirming trends or spotting divergences.
MFI	Uses both price and volume to measure trading pressure and identify potential reversals.	Provides insights into market sentiment and potential points of price reversal based on flow of money.
ATR	Measures market volatility by decomposing the entire range of an asset price for that period.	Helps in adjusting the trading strategy according to market volatility, potentially improving risk management.
Bollinger Bands	Consists of a middle band being an N-period simple moving average, along with upper and lower bands	Useful for identifying periods of low or high volatility, as well as potential price targets

	that are 2 standard deviations away from the middle band.	and trend continuations or reversals.

Table 11: Full Technical Indicators Details

Name	Candlestick Pattern	Description
Hammer		A bullish reversal pattern that occurs during a downtrend. The long lower shadow indicates that sellers drove prices down during the session, but buyers were able to push the prices back up close to the open.
Inverted Hammer		A variation of the hammer and can also be a bullish reversal pattern. It shows that buyers attempted to push the price up, but sellers brought it back down, yet the closing price was still higher than the opening price.
Engulfing Pattern		A two-candle pattern where a small candle is followed by a large candle that "engulfs" the body of the previous candle. A bullish engulfing pattern may indicate a price bottom and a bearish engulfing a top.
Doji		Represents indecision in the market. The opening and closing prices are virtually equal. A Doji after a series of up or down candles can signal a reversal in the trend.

Morning Star		A bullish reversal pattern consisting of three candles: a large bearish candle, a small-bodied candle, and a large bullish candle. It signals the end of a downtrend.
Evening Star		The bearish counterpart to the Morning Star. This pattern is a reversal indicator at the end of an uptrend, marked by a tall bullish candle, a smaller candle, and a large bearish candle.
Piercing Line		A bullish pattern at the end of a downtrend or during a pullback within an uptrend, indicated by a down candle followed by a larger up candle that closes above the midpoint of the first candle's body.
Hanging Man		Appears at the end of an uptrend and warns of a potential reversal downward. It resembles the hammer but occurs at the top of an uptrend.
Shooting Star		A bearish reversal pattern that occurs after an uptrend. It looks like an inverted hammer but is formed when the price opens at its low, rallies, but closes near its open.

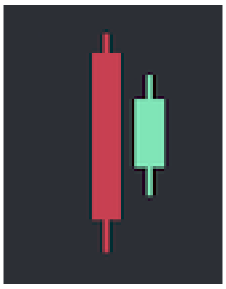
Harami		A two-candle pattern in which a small candle's body is completely contained within the range of the previous body. It suggests a possible reversal, with a bullish harami occurring after a downtrend and a bearish harami after an uptrend.

Table 12: Full Candlestick Pattern Details