SCHOOL OF ENGINEERING MATHEMATICS AND TECHNOLOGY

Improving Generalization and Robustness in Deep Reinforcement Learning for Algorithmic Trading: A Study of On-Policy vs. Off-Policy Methods

Aaron Tyler

MSc Financial Technology with Data Science.

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Thursday 5th September, 2024

Supervisor: Zining Wang

Dedication and Acknowledgements

I want to thank Zining Wang for his council throughout this process most notably when dealing with many challenges that arose throughout this projects development.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

Aaron Tyler, Thursday 5th September, 2024

Contents

List of Figures

List of Tables

Ethics Statement

A compulsory section

This project fits within the scope of the blanket ethics application, as review by my supervisor Zining Wang. I have completed the ethics test on Blackboard. My score is 15/15

Supporting Technologies

- I used VScode for implementing my python code
- I used OpenSSH with the VScode extension to run my code on a more computationally adept remote device
- I used GoogleColab to develop my graphs
- I used Microsoft Word for collecting diagrams before presenting them in my thesis
- I used Powershell to execute interactions with my python code existing within VScode

Chapter 1

Introduction

1.0.1 Algorithmic Trading

Algorithmic trading has become a cornerstone practice in modern financial markets, utilizing advanced computational models to automate decision-making, optimize trading strategies, and execute trades with precision and speed. As these financial markets have evolved, so too have the strategies used to navigate them. Algorithmic trading has seen a significant shift from more rule-based approaches to the integration of more dynamic, and adaptive techniques. Central to this evolution is the increasing interest into Deep Reinforcement Learning (DRL), a subset of artificial intelligence (AI) that enables trading systems to learn and optimize strategies through continuous interaction with market environments. The feasibility of applying DRL in trading has been made possible by some key developments in technology, including the increased availability of high-frequency market data, advancements in data processing, and the availability of more powerful computational resources. These innovations allow DRL models to handle large, complex datasets and adapt in real time, making them a promising alternative than traditional strategies to more effectively navigate periods of increasingly volatile and unpredictable financial markets.

1.0.2 Motivations for this Project

This thesis introduces and evaluates a new DRL model called Trading Proximal Policy Optimization (TPPO), designed to address the limitations faced by both traditional trading strategies and existing DRL models like the Trading Deep Q-Network (TDQN). Traditional models, such as Buy Hold (BH) and Mean Reversion (MR), rely on predetermined rules and are not necessarily equipped to handle more dynamic market conditions. These static strategies can often miss profitable opportunities or expose traders to unnecessary risk during times of high volatility. The TDQN model, a DRL-based algorithm developed by [18], represents an important step forward by moving away from static rules. Instead, TDQN uses DRL techniques to instead learn from its past market experiences to make better informed trading decisions during dynamic market conditions. As a key part of this research, TDQN serves as a benchmark for comparison having shown quite promising results in its implementation. Often beating classical strategies such as Trend Following (TF) and Mean Reversion (MR) when its potential for generalisation has been tested across a wide range of different stock markets.

Despite its strong performance, TDQN faces significant challenges, primarily due to its off-policy learning approach. By updating strategies based on past experiences rather than real-time interactions, TDQN often struggles to adapt swiftly to rapid market shifts. This reliance on representative past experiences increases the risk of overfitting, limiting the model's ability to generalize to unseen market conditions and leading to cases of sub-optimal performance in a more dynamic market environment.

TDQN still provides a promising case for DRL models in algorithmic trading as its highlighted some significant capabilities to outperform traditional strategies across a range of market environments. This success motivates the development of the proposed TPPO model, which aims to build on TDQN's strengths while addressing its weaknesses. TPPO focuses on adapting quickly to current market conditions, reducing the risk of overfitting and improving performance in volatile environments.

1.0.3 Proposed Solution

The Trading Proximal Policy Optimization (TPPO) model, developed in this thesis, offers a novel approach to overcoming the limitations of both traditional trading strategies and existing DRL models

like TDQN. TPPO employs an on-policy learning method, meaning that it continuously updates its strategy based on the most recent market interactions, allowing for real-time adaptability. This is a crucial improvement for navigating highly volatile market environments, where rapid changes require quick adaptations in decision-making.

While TDQN relies on past market experiences to inform its actions, TPPO focuses on learning directly from current market conditions. This shift helps avoid the challenges TDQN faced with overfitting derived from its use of unrepresentative past experiences, making the model more responsive to unexpected shifts in the trading environment. Additionally, TPPO uses Proximal Policy Optimization (PPO) gradient techniques, which ensure gradual, stable updates to its trading strategy. This approach strikes a better balance between flexibility and robustness, allowing TPPO to adapt to market changes without sacrificing reliability.

The aim of this research is to rigorously evaluate the proposed TPPO algorithm by investigating its overall performance against both traditional trading strategies and the existing DRL model TDQN we reference in the literature. This study looks to explore the effectiveness of TPPO when faced with a dynamic range of different market environments. The following chapters will detail the development, testing, and evaluation of TPPO, comparing its performance with that of TDQN and classical strategies like Buy Hold (BH).

The potential significance of TPPO extends beyond academic research, with direct relevance to real-world trading. As algorithmic trading continues to dominate financial markets, investors stand to benefit from more dynamic, adaptable trading strategies. TPPO's ability to respond quickly to market shifts could improve trading decisions, reduce risk, and provide a competitive edge in a rapidly evolving financial landscape. Providing foundations for more widespread implementations of DRL in algorithmic trading practices.

1.1 Aims and Objectives

The primary objective of this project is to develop and evaluate a novel Deep Reinforcement Learning (DRL) trading model, the Trading Proximal Policy Optimization (TPPO) algorithm, which has been specifically designed for this research. This model aims to address key challenges faced by existing DRL models like the Trading Deep Q-Network (TDQN). The project will assess TPPO's potential to outperform traditional algorithmic trading strategies and critically evaluate its effectiveness in dynamic, real-world market environments.

The specific aims of this project are:

1. **Develop the TPPO model** as an alternative to TDQN, focusing on overcoming key challenges such as adaptability to market regime shifts, reducing overfitting, and improving generalization in volatile, non-stationary financial markets.

2. **Test and evaluate both the TPPO and TDQN models** within a simplified trading framework, comparing their performance not only with each other but also with traditional strategies such as Buy & Hold (B&H) and Trend Following (TF). Key evaluation metrics include profitability, risk-adjusted returns (e.g., Sharpe ratio), and robustness.

3. **Critically analyze the feasibility of the TPPO model** for use in real-world algorithmic trading. This includes identifying any limitations in TPPO's generalization capabilities and discussing potential future directions for improving the model's robustness and adaptability in practical trading environments.

Chapter 2

Related Work

Within the field of algorithmic trading, much of the most innovative academic research and publications are often not publicly accessible. As explained by bailey2014, this is largely due to the highly competitive environment in which private firms operate, where proprietary research and cutting-edge developments are closely guarded secrets to maintain a competitive advantage in the industry. As a result, much of the progress in this area is not shared through academic publications, limiting the broader community to access and build upon the latest advancements in Algorithmic Trading. Most of the publicly available works in Algorithmic trading are mathematical and scientific models which do not exploit AI, rather these are more classical rule-based trading strategies. Despite this, the foundational work in algorithmic trading has been well-documented and continues to be an important resource influencing and developing modern practices. A notable example is the popular mean reversion classical trading strategy covered in detail in [3][4][14]. Still commonly employed for rigorous back testing and is being continually developed into more effective deep reinforcement learning (DRL) models [22][9].

A significant breakthrough in the development of algorithmic trading strategies was the pioneering work of [2] on optimal trade execution. The optimal trade execution problem proved to be a fundamental challenge in financial markets and the early life of algorithmic trading. It involved determining the best strategy to execute significantly large orders (buying or selling) in a way that would minimize the execution costs induced through price slippage and overall negative impacts large orders had on the market. Their use of dynamic programming to minimize trade execution costs provided a crucial development of closed-form solutions that addressed the trade-offs between execution costs and market impact. This work laid the groundwork for subsequent research focusing on optimizing trade execution, and has since significantly influenced the evolution of algorithmic trading.

As algorithmic trading matured some limitations of closed-form solutions became more apparent. Particularly given its reliance on strong assumptions about price dynamics which do not translate into real-world markets. These limitations in traditional closed-form solutions paved the way for the application of Reinforcement Learning (RL) into the field of algorithmic trading. Employed in an attempt to address the complexities of the optimal trade execution problem. Unlike static models, which rely on pre-defined rules and assumptions about market behavior, RL allowed the trading agent to continuously adapt to changing market conditions, making it particularly well-suited for the inherently volatile and uncertain nature of financial markets. Reinforcement Learning emerged as a promising approach to address the challenges faced by traditional models. Rooted in the principles of trial-and-error learning and decision-making, RL allows agents to learn optimal behaviors through interactions with their environment, receiving feedback on actions in the form of rewards or penalties. A seminal contribution to this field was made by [20] with the introduction of the Q-learning algorithm. Watkins' Q-learning is a model-free RL technique, meaning it did not require a model of the environment's dynamics to be implemented. This approach enabled an agent to learn the value (Q-value) of taking specific actions in given states, aiming to maximize cumulative rewards over time [20].

Building on the foundational principles of RL and algorithms like Q-learning, researchers began exploring the application of these methods in algorithmic trading. [15] were among the pioneers in this endeavor, presenting one of the first large-scale empirical applications of RL to the optimal trade execution problem. Their work demonstrated how RL could be leveraged to develop trading strategies that are not only adaptive but also capable of optimizing for execution costs in real-time. By allowing the trading agent to interact with the market environment, observe outcomes, and adjust strategies accordingly, RL introduced a dynamic approach to trading that contrasted sharply with the static nature of earlier

closed-form models. The success of early RL applications in trading set the stage for further advancements, including the integration of Deep Learning techniques with Reinforcement Learning. This fusion looked to solve the challenge of high-dimensional and sequential nature of financial data for time-series forecasting. Long Short-Term Memory (LSTM) networks, introduced by [6], has become a cornerstone tool in time series forecasting due to its ability to learn long-term temporal dependencies in sequential data. LSTMs have been widely applied in financial markets for predicting asset prices and market trends which are not captured within traditional mathematical models.[5] applied LSTMs to financial market prediction, achieving better accuracy compared to conventional models. Similarly, [1] developed a deep learning framework combining stacked auto-encoders and LSTMs for financial time series forecasting, further validating the efficacy of LSTMs in this domain.

Another approach to note which effectively tackled the challenges posed by high-dimensional state and action spaces within financial markets is a framework known as a Deep Q-Network (DQN). It extends the Q-learning algorithm by incorporating deep neural networks (DNNs) to approximate the Q-value function, which has shown great promise in handling complex decision-making scenarios in trading [13]. Deep neural networks, with their ability to model non-linear relationships and handle large amounts of data, are particularly well-suited for the financial markets, where the underlying data is often complex, noisy, and high-dimensional [11]. The incorporation of Q-learning in trading algorithms proved to exhibit exceptional generalisation and simplicity. Providing a robust framework for agents to learn optimal policies within unknown or complex trading environments such as financial markets. The evolution of deep reinforcement learning (DRL) has led to innovative research designed to tackle the unique challenges of algorithmic trading and provide increasingly profitable models. The study by by [18] produced a notable contribution developing the Trading Deep Q-Network (TDQN) algorithm. Their work presents a novel approach based on DRL to solve the algorithmic trading problem of determining the optimal trading position at any point across various security markets. This study is inspired by the works of [13] on the Deep-Q Network an architecture well adapted to more effectively deal with partially observable feature spaces within markets and their stochastic nature. The TDQN algorithm is developed to maximize the Sharpe Ratio as its reward function, a widely used performance indicator that balances returns against risks in trading activities. By utilising artificial trajectories generated from a limited set of historical stock market data the reinforcement learning agent can be trained on a partially observable feature set. These artificial trajectories provided necessary generalisation of the model across stochastic and unpredictable financial market data. To mitigate issues such as overestimation bias and training instability the framework also incorporates several enhancements necessary for a high performing trading algorithm. The first to note is the Double DQN technique introduced by [19] to reduce overestimation of action values, leading to more stable learning. Additionally, the study leverages the ADAM optimizer [8] to improve training stability and convergence speed. The inclusion of the Huber loss function [7] further stabilizes the training process by reducing the impact of outliers, which is particularly important in the noisy and volatile environment of financial markets. This newly proposed algorithm represents a significant advancement in the field of algorithmic trading, building on the foundations of RL and DRL techniques to address the specific challenges that exist trading in financial markets. While off-policy algorithms such as Deep Q-Networks (DQN) have shown significant promise in various domains, including algorithmic trading, they are not without limitations. Off-policy algorithms, face several inherent challenges that may limit its effectiveness in the highly dynamic and stochastic environment of financial markets. [18] highlighted these issues stem from poor observability, over fitting, and the sensitivity of DQN models to market regime shifts. Given the challenges associated with TDQN, on-policy algorithms like Proximal Policy Optimization (PPO) present a compelling alternative for algorithmic trading. Unlike off-policy methods, on-policy algorithms update the policy based on the agent's current experiences rather than relying on past data stored in a replay buffer. This continual updating of the policy using fresh data allows the agent to adapt more effectively to the current market environment, improving generalization and reducing the risk of over fitting [16]. The PPO architecture does so by leveraging the benefits of both an actor and critic network. With the actor network responsible for selecting actions (i.e., making trading decisions), while the critic network evaluates the actions by estimating the expected cumulative reward. This setup not only helps in reducing the variance of policy updates by providing a more stable estimate of the value function but also allows for more direct and efficient optimization of the trading strategy [10]. The TDQN algorithm represents a significant advancement in the application of deep reinforcement learning to algorithmic trading. However, despite the promising results and innovations introduced, deep reinforcement learning models like TDQN still face challenges in effectively handling partially observable environments, which are common in financial markets. This highlights the need for further research to improve the adaptability and robustness of DRL models in dynamic and complex market conditions.

Chapter 3

Contextual Background

3.0.1 Financial Markets: An Overview

Financial markets have long been the backbone of global economies, providing platforms for the exchange of various financial instruments, including stocks, bonds, commodities, and currencies. These markets play a crucial role in the allocation of resources across different sectors of the economy, facilitating investment, raising capital, and promoting economic growth. The efficient functioning of financial markets hinges on the principle of price discovery, where prices of securities reflect all available information, ensuring that resources are allocated to their most productive uses.

However, financial markets are also characterized by their inherent complexity and volatility. Prices of financial assets are influenced by a wide range of factors, including economic performance, investor sentiment, geopolitical events, and market speculation. This complexity makes predicting market movements a highly challenging task, which has spurred extensive research into developing models and strategies that can better understand and navigate these markets.

3.0.2 Trading Environment

Modern financial markets operate using complex mechanisms that facilitate the buying and selling of financial securities. One crucial aspect of market operations is the use of continuous double auctions (CDA) [17] to form the basis for price determination and trade execution. In a CDA, orders from buyers and sellers are matched continuously based on price priority and time of submission, forming the core of market liquidity.

At the heart of this process is the limit order book (LOB), a pair of digital records structured to contain all buy (bid) and sell (ask) orders submitted to the market for a particular security. These digital records provide a real-time snapshot of market depth, showing the number of shares available for trade at various price levels. This LOB is typically structured hierarchically with the highest bids at the top of one record and the lowest asks at the top of the other; collectively presenting the best prices available for immediate trade. The most competitive price in the order book is traded immediately if the necessary liquidity (bids) exists and can be matched. Each order is made publicly visible, offering essential transparency, fairness, and efficient price discovery across all financial markets. This is crucial for a market to exhibit effective price determination according to the true dynamics of demand and supply within the market.

The continuous nature of the double auction architecture creates a very fast-paced and highly competitive market environment, where traders must be responsive to exploit short-lived opportunities in the Limit Order Book (LOB). This competitive setting is further complicated by the fact that financial markets are only partially observable. Market participants lack access to all the information influencing prices, introducing significant uncertainty into decision-making. Consequently, traders must strike a delicate balance between acting quickly to seize market opportunities and exercising restraint to avoid premature trade execution under poorly visible market conditions.

3.0.3 Evolution of Algorithmic Trading

Algorithmic trading, the use of computer programs to execute trades based on predefined strategies, has become increasingly prevalent in modern financial markets. The rise of algorithmic trading has been driven by several factors, including the need for higher trading efficiency, further ability to better

exploit short-term market inefficiencies, and the desire to reduce transaction costs. In particular, high-frequency trading (HFT), a subset of algorithmic trading, has gained prominence for its ability to execute large volumes of trades at incredibly high speeds, often within microseconds. The competitive nature of algorithmic trading has led to an arms race among market participants, where the speed and sophistication of algorithms become key determinants of success.

3.0.4 Challenges in Trading

Algorithmic trading is inherently complex given the need to both capture the volatility and predict future price movements across financial markets. These challenges are exacerbated by the presence of adversarial market participants, such as other competing high-frequency traders (HFTs), who can also significantly influence market dynamics through rapid trade execution. Moreover, the impact of black swan events, which are rare and unpredictable occurrences such as the financial crisis in 2008, can lead to further extreme market shifts, adding another layer of difficulty for building effective trading algorithms that rely on pre-defined rules.

In this environment, the role of information is paramount. Trading algorithms must process vast amounts of data, including market prices, volumes, and any available external information such as economic indicators and news sentiment, to make informed decisions. However, traditional algorithmic trading systems often struggle to adapt to new information or sudden market changes, leading to sub-optimal performances during market downturns and more volatile periods.

3.0.5 The Role of Artificial Intelligence in Modern Trading

In response to these challenging high-frequency trading environments with limited observability of necessary information, the integration of Artificial Intelligence (AI) into trading systems has emerged as a promising solution. AI, with its ability to concurrently learn and adapt to new information, offers a an adaptive new method to overcome the rigidity of rule-based algorithms. Machine learning (ML), a subset of AI, has been particularly impactful in this regard, enabling the development of models that can identify more complex patterns in financial data, make more robust predictions, and better optimize trading strategies.

Deep learning (DL), a more advanced subset of machine learning (ML), excels at processing high-dimensional data and extracting insights from large, complex datasets, surpassing the capabilities of traditional ML approaches. This more advanced technique operates using artificial neural networks, which are inspired by the way the human brain operates. These networks consist of layers of interconnected nodes, or "neurons," that work together to process and analyze data. This technology has opened up new possibilities for developing adaptive trading strategies, with the ability to process high-dimensional data and vast datasets much more effectively than traditional machine learning trial and error-based strategies. The application of DL in trading is still in its early stages, but the potential benefits are significant. These technologies promise to create more robust and flexible trading strategies that can better handle the complexities of financial markets, including the ability to adapt to new market conditions, mitigate risks, and improve overall trading performance.

3.0.6 Concerns for Algorithmic Trading and the Future

Algorithmic trading has become a fundamental component of modern financial markets, accounting for a significant share of the trading volume on major exchanges. By leveraging speed, efficiency, and precision, algorithmic trading has not only revolutionized trade execution but also influenced key market dynamics such as liquidity, volatility, and price discovery.

However, as financial markets grow in complexity, so too do the challenges of algorithmic trading. One major concern is the potential for these systems to amplify market volatility, especially during periods of stress. Additionally, the opaque nature of some algorithmic strategies—particularly those used by high-frequency traders—has sparked debates around market transparency and regulatory oversight. In response to these concerns, there is a growing demand for more advanced, adaptive algorithmic strategies capable of navigating complex and uncertain market environments. The integration of artificial intelligence (AI) and deep reinforcement learning (DRL) into trading systems represents a promising step toward creating strategies that are more robust and flexible in the face of evolving market conditions.

Chapter 4

Technical Background

4.1 Deep Learning and Deep Reinforcement Learning

Deep Learning (DL), a subset of artificial intelligence, has proven to be an invaluable tool in the analysis and interpretation of high-dimensional data, such as financial market data. At the core of DL are Deep Neural Networks (DNNs), which consist of multiple layers of interconnected nodes (neurons) that are capable of learning complex patterns and representations from vast datasets. This capability makes DNNs particularly well suited for tasks like financial time series forecasting, where identifying subtle patterns can be crucial for predicting market trends.

However, a key issue in trading is that the outcomes of actions (such as buying or selling a security) depend not just on the current state of the market but also on future states, which are influenced by past decisions. Deep learning models, which are optimized for one-step predictions, do not account for the long-term consequences of actions. For example, a model might predict a rise in stock price, but it lacks the framework to determine when to buy, hold, or sell in a way that maximizes long-term profit. This limitation can lead to suboptimal trading strategies that fail to consider the future impact of decisions made in the present.

Reinforcement Learning (RL), a popular machine learning technique and another subset of AI, alternatively addresses the challenge of making decisions over time. Unlike supervised learning, where models are trained on static datasets, RL involves an agent interacting with the environment to maximize cumulative rewards over time. This approach makes RL especially well-suited for applications in dynamic and uncertain environments, such as financial markets, where the future is inherently unpredictable.

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$$

Where:

- r_i is the reward at time i.
- γ is the discount factor, reflecting the preference for immediate rewards over future rewards.

In RL, the goal is to maximize a cumulative reward R_t over time, where the reward is typically a function of returns adjusted for risk.

Deep Reinforcement Learning (DRL) combines the discussed strengths of deep learning and reinforcement learning, enabling the development of models that can learn and optimize complex policies directly from high-dimensional input spaces. In DRL, the deep-neural networks from deep learning serve as the function approximator, estimating the expected rewards of different actions in various states of the environment. This capability allows the DRL agent to make informed decisions even in complex, high-dimensional spaces where traditional RL methods might struggle.

The integration of DRL into various domains, including algorithmic trading, has shown significant promise by enabling the development of adaptive trading strategies that learn from market interactions and optimize decisions in real-time. This adaptability makes DRL a powerful approach to navigating the complexities of financial markets. But how does leveraging the benefits of reinforcement learning in our deep learning architecture change the algorithmic trading problem?

4.2 Algorithmic Trading as a Reinforcement Learning Problem

Algorithmic trading involves sequential decision-making, where each action—such as buying, selling, or holding—affects future outcomes. Traditional algorithmic trading strategies rely on predefined rules based on historical data and technical indicators to make these decisions. While these strategies can be effective, they often lack the adaptability needed to respond to sudden market changes.

In contrast, Reinforcement Learning (RL) offers a dynamic approach by modeling trading as a Markov Decision Process (MDP). In this framework, a trading agent interacts with the market environment, making decisions to maximize long-term rewards. At each time step t, the agent observes the market state s_t, selects an action a_t, and receives a reward r_t, with the objective of maximizing the cumulative reward over time:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \mid \pi\right]$$

Where:

- π^* is the optimal trading policy,
- r_t is the reward received at time step t,
- γ is the discount factor, which balances immediate versus future rewards.

Reinforcement learning excels in complex environments where information is often opaque, intricate, and only partially observable. By constantly learning from the outcomes of its actions, an RL-based trading agent can adapt its strategy to changing market conditions, improving performance over time and returning valuable information from its interactions with the environment.

As we advance into Deep Reinforcement Learning (DRL), which combines RL with deep learning, the discussed adaptability of a RL agent can be even further improved by allowing the agent to handle high-dimensional input data, such as price history, technical indicators, and market sentiment aswell as the data returned from their frequent interactions with the market. This makes DRL particularly useful in financial markets, where data is abundant and nuanced patterns need to be learned.

In this context, DRL-based trading strategies offer a powerful alternative to traditional rule-based approaches. Unlike rigid policies, DRL agents continuously update their strategies based on real-time interactions with the market, enabling them to better respond to volatility and regime shifts.

4.2.1 On-policy vs. Off-policy Reinforcement Learning Algorithms

In Reinforcement Learning (RL), there are two main types of algorithms: on-policy and off-policy. These algorithms differ in how they interact with the environment, particularly in how they learn and update their decision-making policies.

On-policy algorithms learn directly from the actions they take in real-time. As the agent interacts with the environment, it continuously updates its policy based on the immediate rewards and experiences from the current episode. This allows the algorithm to adapt quickly to changes, making it highly responsive to dynamic market conditions.

Off-policy algorithms, on the other hand, learn from past experiences rather than the actions being taken in real-time. By using an experience replay buffer, these algorithms store and reuse historical data, allowing them to evaluate and improve policies not directly tied to current actions. This approach makes off-policy methods more sample-efficient, as they can repeatedly learn from past experiences to refine their decision-making strategy.

4.3 Challenges of Off-Policy Algorithms in Trading

Off-policy algorithms like Deep Q-Networks (DQN) have shown promise in various applications, but they face specific challenges when applied to algorithmic trading. The Trading Deep Q-Network (TDQN), discussed in the literature, is subject to similar limitations. These challenges include:

- **Limited Market Observability and Sensitivity to Regime Shifts:** Financial markets are inherently complex and only partially observable. Off-policy algorithms, such as TDQN, rely heavily on stored past data from experience replay buffers, which may not fully capture the current state

of the market. This limitation is exacerbated during significant regime shifts, where the market dynamics change, and the historical data no longer reflects the present environment. Consequently, these algorithms struggle to adapt to market regime shifts or accurately interpret the current state when data is incomplete or noisy.

- **Over fitting and Performance Variability:** Off-policy models are prone to overfitting on the historical training data, leading to strategies that perform well during backtesting but fail to generalize to real-world scenarios. Historical data often contains noise, anomalies, or outdated conditions, which causes the learned strategies to be ill-suited for unseen market environments. This reliance on non-representative data leads to significant performance variance, where the algorithm may perform inconsistently when exposed to real-time market conditions that differ from those in the training dataset.

These challenges highlight the need for more adaptable and robust methods in algorithmic trading, particularly in environments characterized by high uncertainty and volatility.

4.4 Advantages of On-Policy Algorithms and Actor-Critic Networks

In contrast, on-policy algorithms like the new proposed Trading Proximal Policy Optimization (TPPO) offer a more suitable framework for algorithmic trading in complex and dynamic environments. An on-policy method updates the policy based on the agent's current actions and rewards, allowing it to adapt more effectively to the current market conditions. TPPO uses an actor-critic network to conduct its on-policy functionality:

Actor Network: Responsible for selecting actions (i.e., making trading decisions) based on the current policy. The policy $\pi_\theta(a_t \mid s_t)$ represents the probability of taking action a_t given the state s_t, where θ denotes the parameters of the neural network (i.e., the weights and biases). The goal of the actor network is to maximize the expected cumulative reward $J(\theta)$ over time, which can be defined as:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

Where: r_t is the reward at time step t, γ is the discount factor, which balances the importance of immediate rewards versus future rewards,T is the final time step,$\mathbb{E}_{\pi_\theta}$ denotes the expectation with respect to the policy π_θ.

The actor network updates its parameters θ by performing gradient ascent on $J(\theta)$, typically using the policy gradient theorem, which gives the update rule:

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

Where $A(s_t, a_t)$ is the advantage function, which estimates how much better or worse an action a_t taken in state s_t is compared to the average action in that state. The gradient $\nabla_\theta J(\theta)$ guides the updates to the actor network, improving the policy over time.

Critic Network: The Critic Network evaluates the actions taken by the Actor Network by estimating the value function $V(s_t)$, which represents the expected cumulative reward starting from state s_t and following the policy thereafter. This value function provides a baseline against which the advantage of the selected action is computed. The Critic Network's feedback helps to reduce the variance of the updates and stabilize the learning process, making the Actor Network's policy improvement more efficient.

The value function estimated by the Critic Network can be represented as:

$$V(s_t) = \mathbb{E}[r_t + \gamma V(s_{t+1}) \mid s_t = s]$$

Where: r_t is the immediate reward received after being in state s_t,$V(s_{t+1})$ is the value of the next state s_{t+1}, indicating the expected cumulative reward if the policy continues to be followed.

This actor-critic architecture reduces the variance in policy updates by leveraging the critic network to provide more accurate value estimates, which act as a baseline for the advantage function. This helps stabilize the learning process by preventing overly large policy updates that could destabilize the model. By doing so, it allows for more direct and efficient optimization of the trading strategy, as the actor network can focus on selecting actions that maximize expected rewards while the critic network refines these choices based on more reliable feedback. This synergistic interaction leads to more consistent and robust performance in dynamic and volatile trading environments.

Chapter 5

Methodology and Design

5.1 Introduction to Methodology

This project looks to expand the research space for DRL algorithms by conducting a constructive and unbiased investigation into novel trading algorithms that leverages cutting edge advancements in Artificial Intelligence (AI) and Reinforcement Learning (RL). As highlighted by the work of [18], the importance of maintaining an unbiased approach in financial research is critical, particularly in a field where there is often a tendency to emphasize positive results, sometimes at the expense of a rigorous scientific methodology. Therefore, this research rigorously evaluates the feasibility of the proposed algorithms to become widely adopted in a competitive trading environment across a wide array of security markets.

Our contribution is the development of a novel on-policy DRL algorithm, Trading Proximal Policy Optimization (TPPO), designed to address the optimal trading problem. TPPO specifically targets challenges like poor market observability and frequent regime shifts, which have proved particularly challenging for the off-policy Trading Deep Q-Network (TDQN) algorithm proposed by [18].

5.1.1 Trading Problem Formalisation

Algorithmic Trading, an automated trading process denoted by a set of mathematical rules is the commonly accepted definition and one we adopt in this paper. Although, throughout academic publications this widely used term is often partitioned into two more specific definitions to ensure distinct meanings are portrayed when discussed in literature. As many authors like to define trading decisions as quantitative trading, and trading execution as algorithmic trading. However, for generality of this paper these terms are synonymous encompassing the entire automated trading process.

To provide an unbiased and constructive investigation of the newly proposed TPPO algorithm and provide comparative analysis with the TDQN algorithm [18] we have chosen to focus on the comparison of trading activities at the level of individual stocks, rather than within a managed portfolio. This decision looks to provide better grounds for rigorous analysis into the feasibility and performance of considered algorithms within the trading environment, revealing more concrete conclusions regarding their future development and adoption into algorithmic trading practices.

5.2 Design of Trading Algorithms

This chapter details the design and development of the Deep Reinforcement Learning (DRL) algorithms that form the core of this research. Focusing on both the theoretical considerations and practical decisions that guided the construction of these algorithms, aiming to address the challenges of algorithmic trading in dynamic financial markets.

5.2.1 Simplified Trading Framework: Acknowledgement

In this project we acknowledge and integrate the simplified trading framework originally developed by [18] as the foundation for our investigation. Designed to facilitate an unbiased scientific analysis of trading algorithms within a homogenized trading environment. Their framework intentionally strips

[1]This simplified framework is based on the work of Theate and Ernst: https://github.com/ThibautTheate/An-Application-of-Deep-Reinforcement-Learning-to-Algorithmic-Trading/tree/main

the true trading environment of unnecessary complexities that could potentially obscure results and limit the capabilities of effective critical analysis, particularly when investigating the feasibility of Deep Reinforcement Learning (DRL) algorithms against traditional trading models.

The predominant adaptations of their simplified trading environment includes a reduced observation space and action space with a discretisation of the trading process into quantifiable time steps. These will be more thoroughly discussed in the following section. This reduction was essential to create a more simplified and controlled environment where the core mechanics of algorithmic trading strategies could be tested without the influence of extraneous variables. This study utilises their proposed simplified framework to ensure true focus on the essential dynamics of trading strategies and the introduction of new competing DRL algorithms.

5.2.2 Time-series Discretisation

Algorithmic trading operates as a continuous process that is not constrained by fixed time intervals, with trades occurring in response to real-time market conditions rather than at predefined moments. This fluidity makes many aspects of trading difficult to quantify, as the frequency of trading activities can vary significantly depending on the algorithm's strategy and the market dynamics. To therefore study the algorithmic trading problem a time discretisation is necessary to perform effective time-series analysis across trading activities. We quantify discrete time-steps t, $t+1$, $t+N$ to model transitions between time-steps throughout the timeline.

These time steps are linked to the trading frequency targeted by the trading agent. For the scope of this project the timeline is constrained to daily trading frequencies based on both technical limitations and sourcing of effective high-frequency historical data spanning the required time-period of our investigation.

5.2.3 Observing the trading environment: RL Agent

In the context of algorithmic trading, the trading environment represents the entire financial ecosystem in which the trading agent operates, encompassing the stock market, economic factors, and other relevant influences. One of the significant challenges in algorithmic trading is the partial observability of the market environment. The trading agent does not have access to all relevant information, as some data may be inherently hidden, such as confidential company information, adversarial market participant strategies or black-swan events (such as the financial crisis 2008). Therefore the RL agent only has access to a limited set of observations at each time step t, which it must use to make informed trading decisions. These observations, denoted as O_t, typically include the agent's current trading position, standard market data such as OHLCV (Open-High-Low-Close-Volume) prices, and potentially derived technical indicators like moving averages or RSI. Additionally, broader economic indicators (e.g., interest rates), sentiment analysis from news and social media can be incorporated to enhance the agent's understanding of the market state. However, this notably limited but crucial set of observations guides the agent's actions within the complex and dynamic trading environment.

To address these challenges, traders often enhance the observation space of their algorithms by leveraging the sequential nature of reinforcement learning (RL). An RL agent uses not only the current observation O_t but also previous observations of interactions, capturing the temporal dynamics of the market. This allows the agent to incorporate historical observations into its strategy, improving its ability to predict future movements and make more profitable decisions in a constantly changing market environment.

In mathematical terms, if we denote the sequence of observations up to time t as $O_{t-N:t}$, the agent's task is to use this sequence to make an informed decision at each time step:

$$a_t = \pi(O_{t-N:t})$$

where π represents the policy or strategy the agent follows, mapping past and present observations to a specific action a_t. By processing this sequence, the RL agent can better identify patterns and predict future market behaviors, thus improving performance despite the inherent partial observability of the trading environment. This approach enables the agent to execute actions based on the current and historical observations in alignment with its policy π.

5.2.4 Reduced Observation Space

While leveraging historical observations allows the agent to make more informed decisions, not all data is necessary for effective decision-making. In this study, we simplify the observation space by focusing

on key market indicators. This reduction streamlines the decision-making process, allowing the agent to operate efficiently in real-time without sacrificing essential information. Specifically, the observation space is reduced to include only the classical OHLCV (Open-High-Low-Close-Volume) data to determine the state s_t of the time step t. The observation space O_t is reduced, comprised of the current state of the agent's trading position together with a series of historical OHLCV data, small o_t. This can be visualized in the following expression:

$$o_t = \left\{ \sum_{t=t-N}^{t} \left(p_t^O, p_t^H, p_t^L, p_t^C, V_t\right) | P_t \right\}$$

The expression represents the reduced observation space o_t at time t, which includes the sum of prices (Open, High, Low, Close) and volume V_t, observed over a period N given the trading position P_t at time step t. By focusing on this reduced yet essential set of observations, the study ensures that the agent's decision-making is grounded in the most pertinent data, thereby simplifying the analysis while retaining the ability to capture key market dynamics.

5.2.5 Trading actions: RL agent

The next fundamental question that the trading agent must address is whether to trade, how to trade, and in what quantity. These decisions are critical, as they directly impact the agent's exposure to market risk and, ultimately, the performance of the trading strategy. The actions can be formalized as:

$$a_t = \pi(s_t)$$

Where s_t represents the state observed at time t (reduced observation space o_t), and π denotes the policy function that maps states to actions. The potential action a_t can take different forms: buying, selling or holding a position.

The actions selected by the agent affect two main components of the portfolio: the cash balance and the number of shares held. Assuming trades occur near the market close price p_t, the updates to these components can be described by the following equations:

$$\text{Cash Balance: } C_{t+1} = C_t - p_t \times n_t \times a_t$$

$$\text{Share Quantity: } S_{t+1} = S_t + n_t \times a_t$$

Where n_t represents the number of shares transacted, which could be positive (buy) or negative (sell), depending on the action a_t.

5.2.6 Constraints on Actions

To ensure the trading environment is both realistic and sustainable for the RL agent two primary constraints are imposed with regards to the actions they take. Ensuring the agent maintains a balanced trading strategy, preventing it from making unsustainable trades.

- **Cash Constraint**: The agent's cash balance C_t must remain non-negative throughout all trading steps t. This limits the maximum number of shares the agent can purchase at any given time.

$$C_t \geq 0 \quad \forall t$$

- **Risk Management Constraint**: When the agent holds a negative position (i.e., has shorted shares), it must maintain sufficient cash to cover potential losses in case of adverse price movements. The maximum relative change in price, denoted by ϵ, is estimated by the agent before trading begins. This parameter ensures the agent can always cover its position if the market shifts unexpectedly.

$$C_t \geq \epsilon \times S_t \times p_t \quad \forall t$$

5.2.7 Reduced Action Space

Considering the inherent complexities and computational challenges involved in determining the optimal trading action, the simplified trading framework narrows the action space to focus on essential decisions. By streamlining the decision-making process, the agent is limited to two key actions:

Maximize Shareholding (Long): The RL agent maximizes its shareholding by converting as much cash as possible into shares. Mathematically, this involves converting the maximum amount of cash C_t into share value S_t, constrained by the current market price p_t.

Minimize Shareholding (Short): The RL agent minimizes its shareholding by converting as much share value as possible into cash. Here, the agent liquidates its shareholding, reducing the number of shares owned to zero or even taking a negative position (short-selling), where allowed.

This reduction simplifies the problem by focusing the agent's decisions on key strategic moves, either fully committing to a long position or exiting/shorting the market entirely. The choice between these actions is informed by the current market state and the agent's policy.

The modifications to the observation space, action space, and time discretization are key considerations for implementing the simplified trading framework we use in this study. These simplifications are necessary to isolate the key elements of trading strategies and to rigorously evaluate the performance and feasibility of the proposed DRL algorithms without the confounding effects of extraneous variables. In doing so, we create a clear, focused environment that is conducive to the scientific investigation of DRL techniques, facilitating more robust and generalizable conclusions devoid of any issues of "black-box" processes commonly cited when analysing complex DRL models.

5.3 Deep Reinforcement Learning Algorithms

The Trading Deep Q-Network (TDQN) algorithm has played a crucial role in advancing the application of deep reinforcement learning in algorithmic trading. Demonstrating a promising performance in the simplified trading environment, TDQN leverages a deep Q-network to approximate the action-value function, incorporating techniques such as experience replay and target networks to enhance stability in the inherently volatile and stochastic financial markets. In this study, TDQN serves as the benchmark for evaluating our proposed TPPO model. Before delving into the specifics of TPPO and its performance evaluation, we will first provide a concise overview of the TDQN architecture.

5.3.1 Introduction to Trading Deep Q-Network (TDQN): Architecture

TDQN uses a deep neural network (DNN) to approximate a Q-value function, which predicts the expected reward for each action in a given market state. The DNN architecture includes:

- **Input Layer**: Incorporates historical OHLCV (Open, High, Low, Close, Volume) data and the current trading position, capturing essential temporal dependencies in financial data.
- **Hidden Layers**: Composed of fully connected layers with leaky ReLU activations, allowing the model to capture non-linear relationships in market data. Regularization techniques like dropout and L2 regularization help prevent overfitting.
- **Output Layer**: Provides Q-values for possible trading actions (buy, sell, hold), guiding decision-making based on the highest predicted value.

A key design choice for the TDQN, an off-policy implementation, was an experience replay mechanism. This technique stores past interactions between the agent and the trading environment in a replay memory. During training, a mini batch of these stored experiences is randomly sampled to update the network. This approach breaks the correlation between consecutive experiences, allowing the model to learn more effectively.

The ϵ-Greedy policy is employed to balance exploration and exploitation during the learning process. Initially, the agent selects actions randomly to explore the trading environment (high ϵ), but as training progresses, ϵ is gradually reduced, allowing the agent to increasingly exploit its learned policy by selecting actions with the highest Q-value.

5.3.2 Training and Improvement: TDQN

The Trading Deep Q-Network (TDQN) algorithm follows a structured process to develop and refine its trading strategy. This process involves a combination of techniques aimed at stabilizing training, improving decision-making, and balancing exploration with exploitation. A key element in TDQN's training is the use of experience replay, where past interactions between the agent and the trading environment are stored in a replay memory M with capacity C. By randomly sampling mini-batches from this memory, the algorithm breaks the correlation between consecutive updates, allowing for more stable learning and preventing the model from overfitting to short-term market patterns.

At each time step t, the TDQN agent selects actions using an ϵ-Greedy policy, a method designed to balance exploration (trying new actions) with exploitation (choosing the best-known action). As training progresses, the value of ϵ is gradually reduced, shifting the focus from exploration to exploiting the learned trading policy.

Training occurs at regular intervals, determined by a parameter T', where a mini-batch of experiences is sampled from M to update the main deep neural network (DNN). To further improve stability and performance, the training process uses the ADAM optimizer, along with gradient clipping based on the Huber loss to mitigate the effect of outliers, which are common in volatile market data. Additionally, TDQN employs a target DNN, updated periodically to match the main DNN, which helps reduce variance and ensures more stable Q-value estimates during training.

The architecture also incorporates Double DQN, a technique that separates action selection from action evaluation, thereby reducing overestimation bias. Finally, the use of the Huber loss function helps stabilize the learning process by reducing the influence of extreme market movements, which can otherwise lead to instability in training.

5.3.3 Critical Evaluation of TDQN and Proposed Developments

The TDQN framework, while having proven to be effective in its use for algorithmic trading, has also revealed some notable limitations when faced with dynamic, partially observable market environments. These limitations arise from the off-policy nature of TDQN, which relies heavily on past experiences rather than real-time market data. As a result, the model can struggle to quickly adapt to sudden shifts in market conditions, limiting its ability to generalize and perform consistently across different trading environments. This reliance on historical data can lead to overfitting and reduced effectiveness when applied to volatile or non-stationary markets, highlighting the need for more adaptable approaches.

Given these challenges, it becomes evident that a more responsive and adaptive approach is necessary to better handle the inherent complexities of financial markets. This leads to the implementation of my proposed Trading Proximal Policy Optimization (PPO), an algorithm designed to overcome the key limitations observed in TDQN.

5.4 Trading Proximal Policy Optimization (PPO) Architecture

Trading Proximal Policy Optimization (TPPO) offers a promising solution by employing an on-policy learning method coupled with an actor-critic architecture. Unlike TDQN, which processes batches of historical experiences and struggles with real-time adaptability, TPPO continuously updates based on real-time market interactions. This allows for more rapid adaptation to shifting conditions, reducing the risk of overfitting and improving the model's generalization across various market regimes.

The actor-critic architecture in TPPO combines the strengths of both policy gradient methods and value-based techniques, resulting in a more stable and efficient framework for optimizing trading strategies. By stabilizing policy updates and focusing on real-time observations, TPPO could prove to be a more suitable choice for dynamic and non-stationary financial markets, where timely and flexible decision-making is crucial. This section explores the PPO architecture in greater detail, highlighting its application in the context of this project and comparing its effectiveness against TDQN.

Actor-Critic Framework

One of the key distinctions between the Trading Proximal Policy Optimization (TPPO) algorithm and TDQN is TPPO's dual-network architecture, consisting of an Actor and a Critic. This dual-network design addresses the challenge of partial observability in financial markets, aiming to improve decision-making in our Deep Reinforcement Learning (DRL) algorithm.

Actor Network: The Actor network is responsible for selecting trading actions by modeling the policy $\pi_\theta(a|s)$, where a represents the action, and s is the current market state. Unlike TDQN, where the policy is indirectly optimized through value-based methods, the Actor directly optimizes its policy by focusing on maximizing the expected cumulative rewards, providing a more responsive and adaptive approach to dynamic market conditions.

Critic Network: The Critic network evaluates the actions selected by the Actor by estimating the value function $V_\phi(s_t)$, which predicts the expected return from the current state. This feedback loop between the actor-critic allows the Actor to adjust its strategy based on the Critic's assessment, leading to more informed and precise policy updates. This contrasts with TDQN's approach, where the action-value is updated in a single network, potentially limiting the depth of feedback available.

PPO further enhances decision-making through the use the discussed advantage function $A(s_t, a_t)$. This addition helps quantify how much better an action is compared to the expected value of the state. The advantage function is computed as follows:

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

This additional layer of evaluation provides more nuanced policy updates than TDQN, which relies on single-step action-value estimates. By incorporating the advantage function, PPO can make more refined decisions, particularly in environments where rewards are sparse or delayed—a common scenario in financial markets.

Training Modifications

The proposed TPPO algorithm introduces several innovative mechanisms that differ from the TDQN framework to ensure stability and efficiency during training. TPPO employs a clipped surrogate objective function to limit the extent of policy updates in each iteration. This is a key difference from TDQN, where policy updates can sometimes be large and destabilizing. The clipping mechanism in PPO ensures that the updates to the policy are gradual, leading to a more controlled and stable learning process.

Generalized Advantage Estimation (GAE) is another crucial modification for TPPO, enhancing the calculation of the advantage function. GAE helps reduce the variance in the policy gradient estimates while maintaining accuracy, leading to more stable policy updates. By introducing a flexible parameter () that balances the trade-off between bias and variance, GAE smooths the learning process and improves stability. This is particularly beneficial in financial markets, where noisy and inconsistent rewards can disrupt learning and cause the model to fit to noise rather than essential trends in the data. GAE allows TPPO to make more reliable updates in these conditions, reducing the sensitivity to short-term fluctuations while retaining responsiveness to long-term trends, which is critical in environments with frequent market regime shifts.

Handling Partial Observability

Given the partial observability of financial markets, Long Short-Term Memory (LSTM) layers are integrated into both the Actor and Critic networks of the TPPO algorithm. While on-policy algorithms like TPPO typically focus on real-time data and do not inherently retain memory of past states, the addition of LSTM layers allows TPPO to capture and leverage sequential data. This enables the model to retain historical market information over time, which is crucial in financial trading, where patterns and trends evolve, and current conditions are often influenced by previous states.

By incorporating LSTM layers, TPPO gains the ability to learn temporal dependencies in market data while maintaining the benefits of on-policy learning. This hybrid approach allows TPPO to adapt more effectively to market regime shifts, offering a significant advantage over off-policy algorithms like TDQN, which generally rely on replay buffers but may struggle with real-time adaptability.

The LSTM structure enables TPPO to make more informed decisions based on past observations, improving its ability to predict future market behavior. By retaining long-term dependencies through LSTM layers, TPPO maintains its on-policy learning advantage while enhancing its memory capacity. This makes it better suited to dynamic market conditions, where frequent regime shifts can challenge models that lack sequential memory, such as TDQN.

5.4.1 Training and Improvement: Novelty of TPPO

The Trading Proximal Policy Optimization (TPPO) algorithm employs a structured training process to develop an adaptive and robust trading strategy. In each training iteration, the agent interacts with the

trading environment by selecting actions based on the Actor network's policy, while the Critic network evaluates the value of these actions, providing feedback on the expected rewards and transitions.

Unlike TDQN, which relies on a replay buffer and off-policy learning, TPPO uses a hybrid approach with a main focus of its on-policy learning approach, directly updating the policy from its latest actions. The clipped objective function is employed to restrict the magnitude of policy updates, ensuring stable learning, even in highly volatile markets. TPPO also uses Generalized Advantage Estimation (GAE), which improves the accuracy of policy updates while reducing variance, allowing for smoother learning across different market conditions.

To better handle sequential data and improve decision-making, TPPO integrates LSTM layers in both the Actor and Critic networks. These layers enable the model to retain and leverage historical market information, making it better suited to detect patterns and respond to temporal dependencies in market data—a key advantage over TDQN's feed forward architecture.

The training process further incorporates gradient clipping to address potential issues like exploding gradients in volatile conditions, while the Adam optimizer ensures efficient and reliable convergence. Additionally, the inclusion of an entropy coefficient in the loss function encourages exploration, helping the agent avoid overfitting and promoting the discovery of new strategies in changing market environments. By balancing stability with adaptability, TPPO can develop a more responsive trading strategy, making it better equipped to navigate the complexities of financial markets compared to the TDQN framework.

5.5 Performance Metrics

In evaluating the effectiveness of deep reinforcement learning (DRL) models compared to classical trading strategies, it is essential to consider more than just profitability. A robust model must consistently perform well across a wide variety of market conditions while demonstrating stability and generalizability. This includes assessing not only how a model performs during training but also how well it adapts and maintains its performance during testing in real-world scenarios. The balance between profitability, risk, and robustness is crucial for determining a high-performing trading model.

One of the most universally accepted measures of this balance is the Sharpe Ratio. This metric goes beyond simple profitability, factoring in the risk taken to achieve those profits. In the context of DRL models, where the goal is to generate not just high returns but also high risk-adjusted returns, the Sharpe Ratio enables a direct comparison with classical strategies. It helps to determine if the added complexity and adaptability of DRL models result in better trading performance under real-world conditions. The Sharpe Ratio S is given by:

$$S = \frac{\bar{R} - R_f}{\sigma_R}$$

Where: $\bar{R}$ is the average return of the trading strategy,R_f is the risk-free rate of return, representing a baseline level of return with zero risk, such as that from government bonds,σ_R is the standard deviation of the trading strategy's returns, representing the volatility or risk associated with the returns.

This metric is particularly relevant for DRL models, where the objective extends beyond achieving high returns to maximizing risk-adjusted returns. The Sharpe Ratio provides a clear, quantitative means of comparing DRL models with classical strategies, evaluating whether the added complexity and adaptability of DRL approaches result in superior real-world trading performance. In this context, maximizing the Sharpe Ratio ensures that the trading strategy is not only profitable but also stable, offering a consistent balance between risk and reward.

5.6 Performance Assessment

This section outlines the methodology used to evaluate the performance of algorithmic trading strategies within a simplified trading environment. The focus begins with an assessment of both the TDQN and the proposed TPPO algorithms. To ensure a comprehensive evaluation, we extend this analysis by comparing their performance to classical trading strategies such as Buy & Hold and Trend Following. The evaluation is conducted across diverse market conditions, including both volatile and stable environments, as well as varying trading frequencies, to provide a more robust and realistic assessment of the models' adaptability and effectiveness.

5.7 Testbench: Main Objective

To ensure an unbiased and comprehensive performance assessment, we propose a testbench consisting of 29 securities from diverse global markets. As shown in Table 5.1, these securities were carefully selected to provide a representative cross-section of different sectors and regions. Adjustments were made to the original testbench environment outlined by [18] to account for recent market changes. For example, Twitter was delisted from the NYSE after rebranding to X, Facebook rebranded as Meta Platforms under the new ticker META, PetroChina delisted its American Depositary Shares (ADS) while remaining on the HKSE (0857.HK), and Shell consolidated its A and B shares into a single listing (SHELL.AS).

Sector	**American**	**European**	**Asian**
Trading Index	Dow Jones (DIA) S&P 500 (SPY) NASDAQ (QQQ)	FTSE 100 (EZU)	Nikkei 225 (EWJ)
Technology	Apple (AAPL) Google (GOOGL) Amazon (AMZN) Meta (META) Microsoft (MSFT)	Nokia (NOK) Philips (PHIA.AS) Siemens (SIE.DE)	Sony (6758.T) Baidu (BIDU) Tencent (0700.HK) Alibaba (BABA)
Financial Services	JPMorgan Chase (JPM)	HSBC (HSBC)	CCB (0939.HK)
Energy	ExxonMobil (XOM)	Shell (SHELL.AS)	PetroChina (0857.HK)
Automotive	Tesla (TSLA)	Volkswagen (VOW3.DE)	Toyota (7203.T)
Food	Coca Cola (KO)	AB InBev (ABI.BR)	Kirin (2503.T)

Table 5.1: Performance Assessment Testbench Stocks (The data sourced from Yahoo Finance [21])

For this study, we selected an eight-year period from January 1st 2012, to January 1st 2020, aligning with the time frame used in [18] TDQN algorithm study. This allows us to directly compare the performance of our model with their TDQN architecture. The chosen period provides a representative sample of market conditions, while avoiding black-swan events like the 2008 financial crisis, which could negatively impact model training stability. By maintaining consistency with the previous research, we ensure that our findings are comparable. The trading horizon is divided into training and testing phases with an 75-25 split, ensuring sufficient data for model learning and evaluation in dynamic market environments.

5.7.1 Trading Strategies

Classical trading strategies continue to play a significant role in modern financial markets, often serving as backtesting instruments or are still continually being integrated into newer algorithms. To provide a well rounded performance assessment of the newly proposed DRL algorithms, we employ a sample of classical strategies to serve as an additional benchmark. Specifically, we utilize a combination of two passive and two active strategies. The passive strategies, Buy and Hold (B&H) and Sell and Hold (S&H), maintain fixed positions over the entire time horizon with no reactive adjustments to market changes. On the other hand, the active strategies Trend Following (TF) and Mean Reversion with moving averages (MR) dynamically adjust their positions throughout the trading period. Trend Following reacts to significant market trends, while Mean Reversion bets against the market in weak trends by selling and then anticipating a return to the mean to buy the stock again. In this study, we evaluate six trading strategies across 29 securities to examine the robustness of the newly proposed DRL algorithms TPPO amidst a diverse range of market conditions. This comprehensive approach ensures that the models are tested not only against each other but also against well-established classical trading techniques.

5.8 Hyper parameter Tuning: Implementation and Rationale

In developing robust and effective deep reinforcement learning (DRL) algorithms to compete with the historically effective classical trading strategies, hyperparameter tuning plays a critical role. The process involves optimizing various parameters which govern the learning process; ensuring that the algorithm can generalize well across different market conditions. In the basis of this study the DRL algorithm must generalise well across 29 securities with largely differing market volatility and data complexities. So effective hyperparameter tuning is paramount to ensure the success of the proposed TPPO model. This

section details the methodology used for hyperparameter tuning in this project, focusing on the rationale behind the selection of seeds and the choice of Volkswagen as the initial stock for testing.

Model Initialisation and challenges of data

The hyperparameter tuning process for the proposed TPPO model began with initialising a baseline of trading hyperparameters. These parameters were sourced based on those described in the TDQN architecture, which had already demonstrated promising generalizability across various testbench stocks. Despite the success of the TDQN algorithm with this parameterization, its success would not translate to the proposed TPPO model given its divergent architecture. Hence these parameters and their results would be used to solely provide an initial direction for hyperparameter tuning.

5.8.1 Robustness Testing Across Multiple Stocks and Seeds

To initially investigate the ability of the TPPO model to generalise across different stocks and market conditions, a representative sample of 10 diverse stocks was selected for initial testing. Simulations were conducted for each stock across six different random seeds to account for variability in model performance due to DNN random weigh initializations. This method provided insights into the model's ability to generalize across various market conditions, with the basic sourced TDQN parameters. This process offered crucial insights into the TPPO model's performance across different securities, highlighting potential challenges for hyperparameter tuning.

The results showed significant variability in the models performance, particularly when different seeds were used. Stocks like Tesla and Apple, with relatively straightforward historical data, were more consistently captured and effectively predicted. In contrast, more volatile and complex stocks such as Alibaba and Volkswagen presented substantial challenges, leading to inconsistent outcomes across different seeds.

5.8.2 Navigating Random Weight Initialization and 29 Stocks

Policy gradient methods like PPO update the policy based on the gradient of expected rewards with respect to the policy parameters. This process is highly sensitive to the initial weight configuration because the entire policy (which determines the actions) is directly dependent on these weights. If the initial weights are poorly set, the policy might start with suboptimal actions, which can lead to poor exploration and exploitation balance, making it difficult for the model to recover and find an optimal trading strategy. The variability in performance underscored a fundamental challenge in deep reinforcement learning (DRL) model development, the impact of random weight initialization. In DRL, neural network weights are randomly initialized at the start of each training run. This randomness can significantly influence the learning trajectory and outcomes, even when all other factors remain constant. For the TPPO model, this issue was particularly pronounced due to its use of policy gradient methods. TPPO updates its policies based on the gradient of expected rewards with respect to the policy parameters, a process that is highly sensitive to the initial weight configuration. Since the policy, which dictates the actions, directly depends on these weights, poor initial settings the TPPO model exhibited a higher tendency to overfit to complex market data, frequently converging to suboptimal local minima. This issue was particularly pronounced in simulations with Alibaba and Volkswagen, where the model over learned the nuances of the volatile training periods, leading to poor generalization to broader market behavior. The hyperparam-

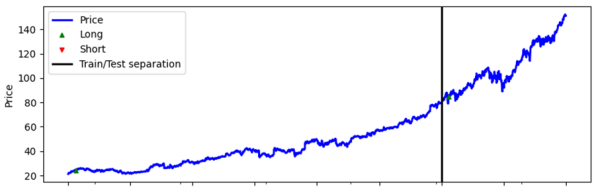

Figure 5.1: Simple Bull Market

eter tuning process faced a significant challenge in ensuring the model could effectively generalize across the diverse market conditions of the 29 selected stocks. As the tuning process had to ensure the model effectively balanced adaptability and robustness, allowing it to perform consistently well across varying levels of volatility, trends, and market dynamics. Figures 5.1, 5.2, 5.3 illustrate the significant diversity of stock data that must be considered in effective hyperparameter allocation. Figure 5.1 shows a simple

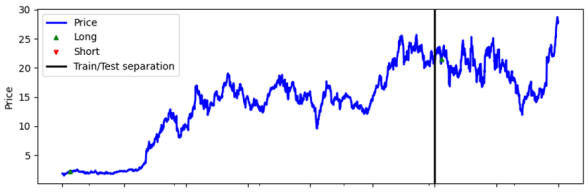

Figure 5.2: Complex Bull Market

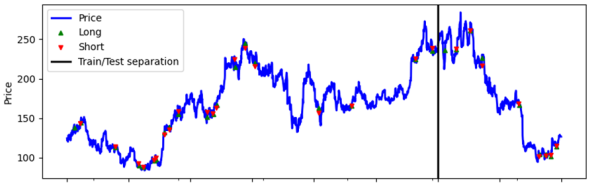

Figure 5.3: Market experiencing significant regime shifts

bullish market, marked by a consistent upward trend with low volatility. Figure 5.2 presents a more complex bullish market, where significant fluctuations add complexity to pattern recognition and trend forecasting despite the overall upward movement. Finally, figure 5.3 represents a market with substantial regime shifts, characterized by sudden changes in trends and volatility, requiring the model to be highly adaptable and responsive.

These diverse market conditions presented a significant challenge for the tuning process. The model needed to learn from complex patterns to prevent underfitting while also avoiding the risk of overfitting to simpler, more stable data. Striking this balance was essential to ensure consistent performance across all market scenarios, maintaining both generalizability and precision.

5.8.3 Volkswagen as a Case Study for Hyperparameter Tuning

In the context of this research, the selection of the stock and seed combination for hyperparameter tuning was driven by the need to initially address the most challenging scenarios within the trading environment. After testing the initial hyperparameters derived from the TDQN architecture, Volkswagen emerged as a particularly insightful case among the stocks tested. Despite its complexity, the stock yielded promising results in approximately half of the seed iterations, producing positive Sharpe ratios. However, the other half of the seeds tested returned negative Sharpe ratios, highlighting significant instability caused by variations in weight initialization. Seed 6 was identified as the most unstable, with a Sharpe ratio of -0.837, a substantial maximum drawdown, and other poor performance metrics. These indicators suggested that the model struggled significantly under this seed's initialization conditions, making it an ideal candidate for becoming the main focus of the hyperparameter tuning process.

5.8.4 Hyperparameter Tuning Process

The hyperparameter tuning process started with a focused effort to stabilize the model's performance, which was highly sensitive to initialization conditions. Seed 6 of the Volkswagen stock, identified as particularly unstable became the primary focus for tuning due to its challenging nature. The objective was to identify a combination of conservative hyperparameters that would enhance stability during both the training and testing phases to balance both overfitting and underfitting for this specific seed-stock pairing. Once promising hyperparameter combinations were identified, they could be tested across other, better-performing seeds to determine whether they provided consistent stability across different weight initialisations and also other stocks in the testbench. This methodology aimed to achieve an effective balance between adaptability and generalization, ensuring robust performance across various stock-seed market conditions.

Given the constraints of computational resources, an exhaustive grid search across the entire hyperparameter space was impractical. Instead, a random search was conducted, sampling 100 different combinations of hyperparameters. This approach was deemed sufficient to explore the parameter space

effectively while managing resource limitations. The considered hyperparameters and their respective values ranges were as follows:

Hyperparameter	**Values**
Learning Rate	[0.00001, 0.0001, 0.0005, 0.001]
Dropout Rate	[0.0, 0.1, 0.3, 0.5]
Weight Decay	[1e-5, 1e-4, 1e-3]
Entropy Coefficient	[0.1, 0.01, 0.001, 0.0001]
Gamma	[0.4, 0.7, 0.9, 0.99]
Batch Size	[32, 64, 128]

Table 5.2: Hyperparameter Search Space

These hyperparameters in table 5.2 were chosen for their crucial roles in controlling the model's learning process, including the pace of learning (learning rate), regularization (dropout rate and weight decay), exploration-exploitation trade-off (entropy coefficient), and discounting of future rewards (gamma). The batch size also played a significant role in determining how the model processed data during training.

By conducting this random grid search, we generated around 100 combinations from the parameter space, providing substantial coverage of potential optimal candidates. Each of these candidates was rigorously tested on the Volkswagen Seed 6 stock-seed initialization. This process yielded a list of the highest-performing combinations, identified by their Sharpe ratio values, which were then selected for further testing in the more expansive trading environment.

Sensitivities in DRL model initialisation

However, when applying these top-performing hyperparameter combinations back within the main trading environment, a critical sensitivity was discovered. Despite using identical frameworks, certain combinations failed to replicate their previously strong performance when used in the broader main testing environment. The only plausible difference between the tuning and main environments was the initialization of the model's neural network weights, which alone seemed to have caused significant variations in performance across the same stocks and seeds. When conducting the tuning process, the "tuning environment" would re-initialise the PPO environment with the necessary hyperparameter combination, inadvertently creating a new neural network initialisation of network weights for each of the random seeds. This discrepancy highlights the crucial role of neural network weight initialization in DRL model performance. Even with identical hyperparameters, random weight initialization can lead to different learning paths and therefore extremely varied outcomes.

This sensitivity to weight initialization indirectly served as a critical filter. It allowed for an effective narrowing down of the list of previously viable hyperparameter candidates. With combinations that failed to maintain consistent performance across the tuning environment being eliminated from consideration. This process highlighted that only those hyperparameter sets capable of achieving robust performance regardless of initialization could be deemed truly generalizable. As a result, the focus shifted to a now smaller list of hyperparameters combinations that seemingly demonstrated resilience to variations in weight initialization, ensuring a more reliable and consistent performance across diverse market conditions.

Refining Hyperparameter Selection Based on Broader Performance Metrics

Given the discrepancies observed between the tuning and main environments, it became clear that relying solely on the Sharpe ratio to identify a list of "top performers" was not effective. While the Sharpe ratio offered valuable insights into risk-adjusted returns, the variability in model performance across different initializations underscored the need for a more thorough evaluation process to select hyperparameters capable of consistent performance and would prove valuable to include in further testing.

To address this, the focus expanded to include a broader range of performance metrics, such as the Sortino ratio, maximum drawdown, and volatility. The Sortino ratio, which distinguishes between harmful volatility (downside risk) and overall volatility, offered a more nuanced assessment of risk-adjusted returns, making it a crucial addition. Maximum drawdown was essential for understanding the largest observed loss from those hyperparameters from peak to trough, and an extra consideration for volatility provided insights into the stability and consistency of returns. By incorporating these additional metrics we could more effectively determine a new list of top hyperparameter candidates for further testing . This

multi-faceted evaluation approach ensured that the selected hyperparameters were more likely to perform consistently and reliably across different market conditions and weight initializations, providing better chances for finding an optimal combination of hyperparameters which would provide effective tuning for the model to effectively generalise across all the considered stocks.

Final Validation Across Multiple Stocks and Seeds

With the refined set of hyperparameter combinations in hand, the next crucial step was to validate their generalizability across a broader range of market conditions. This final verification process was designed to ensure that the consistency observed from the tuning environment to the main environment were not merely isolated successes from experiencing two cases of favorable initializations across both environments. The testing involved evaluating the top performing hyperparameter combinations on four distinct stocks, each representing different market behaviors and volatility profiles. Additionally, the tests were conducted across three different random seeds, which introduced necessary variability in the initialization of the model's neural network weights. By expanding the scope of the testing in this way, the objective was to confirm that the hyperparameters could maintain their effectiveness regardless of the specific market conditions or initial model configurations.

Stock	Seed	Profit & Loss (P&L)	Sharpe Ratio	Max Drawdown Duration
Volkswagen	5	13722	0.385	110 days
	6	13722	0.385	110 days
	7	13722	0.385	110 days
FTSE 100	5	542	0.088	173 days
	6	542	0.088	173 days
	7	542	0.088	173 days
Nokia	5	-30260	-0.408	208 days
	6	-31659	-0.440	208 days
	7	10748	0.313	236 days
ExxonMobil	5	-1509	0.055	63 days
	6	-1509	0.055	63 days
	7	-1509	0.055	63 days

Table 5.3: Optimal Hyperparameter Combination: Performance metrics across different seeds for selected stocks

The results presented in Table 5.3 reflect the performance of the TPPO model using the best-found hyperparameter combination after extensive testing. For the representative stocks considered, the selected hyperparameters delivered consistent Profit Loss (PL), Sharpe Ratios, and Maximum Drawdown Durations across multiple seeds, validating the model's ability to achieve robustness and generalizability. This stability suggests that the hyperparameter configuration effectively allowed the TPPO model to perform reliably across different market conditions and random weight initializations.

While the results for Nokia showed some variability—particularly with negative Sharpe ratios and P&L for seeds 5 and 6—these outcomes underscore the TPPO model's sensitivity to random weight initialization, even with the most conservative hyperparameters. Although my further analysis in chapter 6 looks to examine the specific results observed in seed 6, I have also considered the possibility that these outcomes may have been influenced by favorable or unfavorable initializations. The variability between seeds highlights how subtle differences in starting conditions can impact the model's performance, even when conservative hyperparameters are applied. While the hyperparameter tuning process aimed to mitigate this sensitivity, it did not completely eliminate the models sensitivities to these random initialisations, as seen in Nokia's results. Nonetheless, this variability does not undermine the overall success of the hyperparameter tuning process. Rather, it highlights the complexity of certain market dynamics and provides important insights into the generalization capabilities of TPPO in real-world trading scenarios, particularly when compared to the alternative TDQN algorithm in the following sections.

The next phase of the study will focus on applying this optimal hyperparameter configuration across the full testbench of 29 securities continuing with seed 6, providing a more comprehensive evaluation of the TPPO model's real-world performance and resilience in diverse and challenging financial environments.

Chapter 6

Critical Evaluation

6.1 Testbench Results

6.1.1 Introduction

This section explores the effectiveness of the proposed TPPO (Trading Proximal Policy Optimization) algorithm by rigorously evaluating its performance against the TDQN (Trading Deep Q-Network) algorithm and other classical trading strategies. The evaluation is grounded in the performance assessment methodology described in (Chapter 5). Through this comparative analysis, we aim to uncover insights into the robustness of these Deep Reinforcement Learning (DRL) algorithms, particularly in their ability to navigate significant market regime shifts and the partial observability when trading in financial markets.

In the following subsections, we first examine individual stocks where the TPPO algorithm underperformed. These case studies will investigate the specific conditions under which the TPPO algorithm struggles and how these results compare to the TDQN implementation.

Next, we provide a comprehensive comparison across the entire testbench of results. This evaluation will necessarily contextualize the overall performance trends of TPPO and TDQN across diverse market conditions.

Finally, we analyze the stocks where both models converged to similar results. This analysis is visualized through a scatter plot, which will serve as a basis for a critical discussion on the implications of model convergence. Specifically, we will explore how such convergence impacts the objectives of algorithmic trading, along with the benefits and challenges introduced by the inherent randomness in DRL models.

6.1.2 Divergent Stock Performances: Coca-Cola and Baidu

When investigating the performance of TPPO across the various stocks, Coca-Cola (KO) and Baidu (BIDU) emerged as significantly poor performing outlier cases. These case studies are extremely informative regarding challenges faced by the TPPO model in achieving robust generalization across different market environments.

Coca-Cola (KO)

For Coca-Cola, the TPPO algorithm yielded a Sharpe ratio of -0.404 during the training period, indicating a negative risk-adjusted return even when the model was exposed to favorable market conditions. This performance deteriorated further during the testing phase, where the Sharpe ratio dropped to -0.871, signifying a clear failure to generalize from the training data to real-world market conditions. These consistently low Sharpe ratios highlight the model's inability to exploit the upward market trend that characterized Coca-Cola's stock during both the training and testing periods, ultimately resulting in a capital loss of $34,020 across the two-year testing period.

The price and capital performance data for Coca-Cola (Figure 6.1) reveals a steady decline in the model's capital throughout the testing period, despite a clear upward trajectory in the stock price. This suggests that the TPPO algorithm under fitted the data, largely due to its overly cautious configuration, including a low learning rate and limited exploration during training. These conservative settings likely caused the model to misinterpret short-term fluctuations as noise, preventing it from capturing the

broader upward trend. As a result, the model poorly traded against market signals, leading to substantial capital losses and missed opportunities during both the training and testing phases.

In contrast, the TDQN algorithm significantly outperformed TPPO, achieving a Sharpe ratio of 1.068 in the testing phase, reflecting a much better risk-adjusted return. TDQN's simpler and more stable architecture, combined with its off-policy nature and use of experience replay, allowed it to generalize more effectively from training to testing. This enabled TDQN to balance risk and return more efficiently, avoiding the underfitting issues that plagued TPPO. The added flexibility and stability of TDQN made it far better suited to handle Coca-Cola's straightforward but sustained upward market conditions.

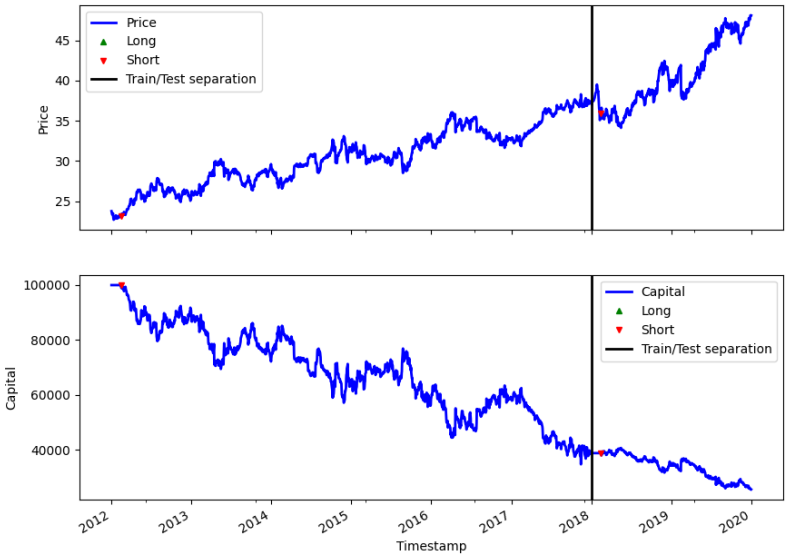

Figure 6.1: Price vs Capital Performance for Coca Cola (TPPO model)

Baidu (BIDU)

The TPPO algorithm also under performed in Baidu's market, with a Sharpe ratio of -0.886 during testing, compared to 0.414 during training (performance breakdown available in appendix). Unlike in Coca-Cola's case, this under performance is attributed to overfitting. The model over-adapted to volatile market conditions during training, learning patterns that were highly specific to the training data. When market dynamics shifted in the testing phase, TPPO failed to generalize, continuing to make frequent, unprofitable trades based on outdated patterns as seen in figure 6.2.

The conservative hyperparameters, such as a low learning rate and limited exploration, contributed to this overfitting by making the model excessively cautious and overly responsive to short-term fluctuations making it unable to adjust to new trends. The sharp decline in Sharpe ratio from training to testing reflects this lack of adaptability. While such hyperparameters aim to prevent excessive risk, they can also hinder the model's ability to adjust to real-world market shifts, indicating the need for stock-specific tuning and more flexible hyperparameter strategies.

In contrast, the TDQN model, with a modest Sharpe ratio of 0.080 during testing, showed better generalization. Its simpler architecture, despite also using conservative settings, allowed it to maintain stability and avoid both overfitting and underfitting, enabling more consistent performance across the testing phase's volatile conditions.

6.1.3 Implications for Hyperparameter Selection

These examples highlight the delicate balance required in hyperparameter selection, especially for a more complex model like TPPO that needs more intensive tuning to generalize across a diverse set of

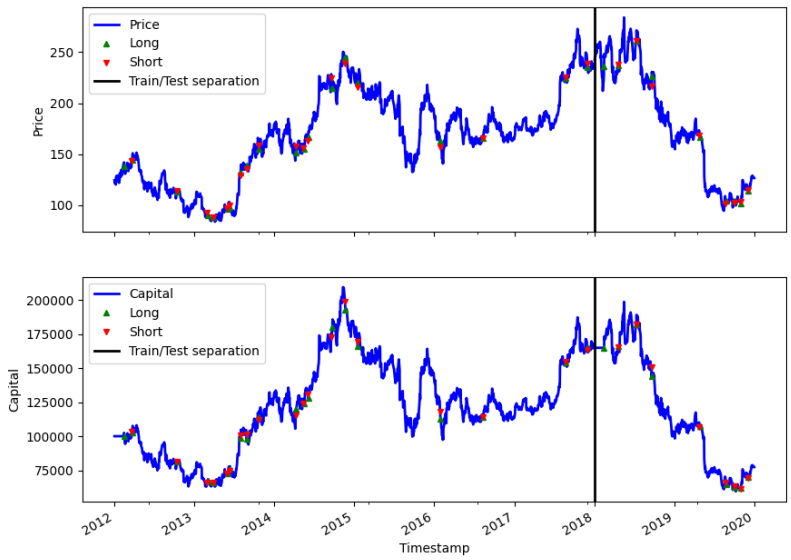

Figure 6.2: Price vs Capital Performance for Baidu (TPPO model)

stocks. The conservative hyperparameters selected for TPPO, intended to prevent both overfitting and underfitting, may have restricted the model's ability to effectively learn and adapt to key market trends. This led to issues such as underfitting, as seen with Coca-Cola, or overfitting, as observed with Baidu, where the model was overly sensitive to learning specific volatile patterns in the training data. The challenge lies in selecting hyperparameters that strike a balance between overfitting and underfitting, allowing TPPO to adapt more robustly to varying market conditions

In contrast, the TDQN model, with its simpler design and more focused learning strategy, avoided these pitfalls, delivering better overall performance by balancing risk and return more effectively. This suggests that while TPPO's approach aimed for broad generalization, the model's complexity proves to add extra difficulties for balancing the chosen hyperparameters. Therefore the TPPO model may need further refinement to achieve robust performances across every considered security market's conditions in this investigation.

In conclusion, while TPPO sought to generalize effectively across all stocks, the results reveal the inherent challenges in tuning hyperparameters for a complex model. In cases like Coca-Cola, the conservative tuning led to underfitting, causing the model to miss clear trends, whereas in Baidu, the same settings resulted in overfitting, where the model became overly sensitive to volatile data. This demonstrates the difficulty of creating a one-size-fits-all solution for market environments where conditions across stocks vary significantly.

6.1.4 Comprehensive Testbench Results: Sharpe Ratio Comparison

To provide a broader context beyond individual case studies, Table 6.1 presents the Sharpe ratios for all testbench stocks across the various strategies (BH, SH, TF, MR, TDQN) and compares them with the results obtained from the TPPO algorithm.

Besides having some outlier cases the results from the testbench provide a compelling case for the potential employment of TPPO models in real-world algorithmic trading environments. Compared to classical methods such as Buy and Hold (B&H) or Trend Following (TF), TPPO demonstrated a superior ability to achieve higher Sharpe ratios and overall profits, indicating better risk-adjusted returns across a diverse range of stocks. For traders and financial institutions considering the implementation of these models, the results imply a promising opportunity to enhance trading strategies by leveraging the adaptability and more sophisticated learning capacity of DRL.

As presented in table 6.1 the average Sharpe ratio for TPPO across all testbench stocks was 0.380,

Stock	B&H	S&H	TF	MR	TDQN	TPPO
Dow Jones (DIA)	0.684	-0.636	-0.325	-0.214	0.684	0.684
S&P 500 (SPY)	0.834	-0.833	-0.309	-0.376	0.834	0.834
NASDAQ 100 (QQQ)	0.845	-0.806	0.264	0.060	0.845	0.845
FTSE 100 (EZU)	0.088	0.026	-0.404	-0.030	**0.103**	0.088
Nikkei 225 (EWJ)	0.128	-0.025	-1.649	0.418	0.019	**0.128**
Google (GOOGL)	0.570	-0.370	0.125	0.555	0.227	**0.570**
Apple (AAPL)	1.239	-1.593	1.178	-0.609	**1.424**	1.239
Meta (FB)	0.371	-0.078	0.248	-0.168	0.151	**0.456**
Amazon (AMZN)	0.559	-0.187	0.161	-1.193	0.419	**0.560**
Microsoft (MSFT)	1.364	-1.390	-0.041	-0.416	0.987	**1.364**
Nokia (NOK)	-0.408	0.565	1.088	1.314	**-0.094**	-0.440
Philips (PHIA.AS)	1.062	-0.672	-0.167	-0.599	0.675	**1.063**
Siemens (SIE.DE)	0.399	-0.265	0.525	0.526	**0.426**	0.410
Baidu (BIDU)	-0.699	0.866	-1.209	0.167	**0.080**	-0.886
Alibaba (BABA)	0.357	-0.139	-0.068	0.293	0.021	**0.357**
Tencent (0700.HK)	-0.013	0.309	0.179	-0.466	-0.198	**-0.043**
Sony (6758.T)	0.794	-0.655	-0.352	0.415	0.424	**0.868**
JPMorgan Chase (JPM)	0.713	-0.743	-1.325	-0.004	**0.722**	0.713
HSBC (HSBC)	-0.518	0.725	-1.061	0.447	0.011	**0.725**
CCB (0939.HK)	0.026	0.165	-1.163	-0.388	**0.202**	-0.001
ExxonMobil (XOM)	0.055	0.132	-0.386	-0.673	**0.098**	0.055
Shell (RDSA.AS)	0.488	-0.238	-0.043	0.742	0.425	**0.488**
PetroChina (PTR)	-0.376	0.514	-0.821	-0.238	0.156	**0.514**
Tesla (TSLA)	0.508	-0.154	-0.987	0.358	**0.621**	0.508
Volkswagen (VOW3.DE)	0.384	-0.208	-0.361	0.601	0.216	**0.385**
Toyota (7203.T)	0.352	-0.242	-1.108	-0.378	0.304	**0.354**
Coca Cola (KO)	1.031	-0.871	-0.236	-0.394	**1.068**	-0.871
AB InBev (ABI.BR)	-0.058	0.275	0.036	-1.313	**0.187**	-0.058
Kirin (2503.T)	0.106	0.156	-1.441	0.313	**0.852**	0.104
Average	**0.369**	**-0.202**	**-0.331**	**-0.056**	**0.404**	**0.380**

Table 6.1: Sharpe Ratios for Various Stocks across Different Strategies and Your Results

slightly below the 0.404 achieved by TDQN. While this difference may appear modest, it highlights crucial distinctions between the two models, particularly in their sensitivity to varying market conditions and their ability to generalize across diverse financial environments. As shown in the testbench table, the better-performing DRL model for each stock iteration has its sharpe ratio value highlighted in bold and dark green. Interestingly however, TPPO outperformed TDQN in 14 cases, while TDQN had superior results in only 12. This outcome could be attributed to TPPO's increased complexities and ability to capture more intricate patterns in the data. Evidently when TPPO correctly aligns with market trends, it can deliver significantly better results than TDQN notably in stock-specific cases such as Nikkei 225, Microsoft, Philips and HSBC. However, this increased complexity over TDQN also evidently leads the model to increased risks of overfitting or missing broader market trends in simple datasets which we evidently highlighted in stocks like Baidu and coca cola, which would explain the lower overall Sharpe ratio. This comparison highlights the trade-off between TPPO's ability to capture intricate market dynamics and TDQN's superior robustness and consistency, making the latter a more reliable choice for broader market generalization.

6.1.5 Analysis of Sharpe Ratios and Model Performance

An interesting trend to highlight in the testbench results from Table 6.1 is the similar Sharpe ratios produced by TDQN and TPPO across a wide range of stocks, despite their architectural differences. [12] conducted a comparable study, testing various Deep Reinforcement Learning (DRL) algorithms across 19 securities. They found that while advanced DQN models showed some improvements over basic DQN, they were generally outperformed by PPO-based approaches in terms of effectiveness, stability, and applicability to optimal trade execution. The robustness of PPO models, particularly in handling sparse

rewards and maintaining consistent performance across diverse stocks, was emphasized. The convergence between TDQN and TPPO in our results remains a concerning factor. This suggests that despite the theoretical and empirical advantages of TPPO as highlighted by [12], the convergence raises important questions about the effectiveness of our project's framework and the overall approach to model tuning.

The scatter plot in Figure 6.3 illustrates this alignment in Sharpe ratios across the different stocks. Many data points cluster close to the line of equality (y = x), indicating that the models frequently yielded comparable results. While this might suggest a degree of robustness in the models, it also indicates potential issues with how these results were achieved.

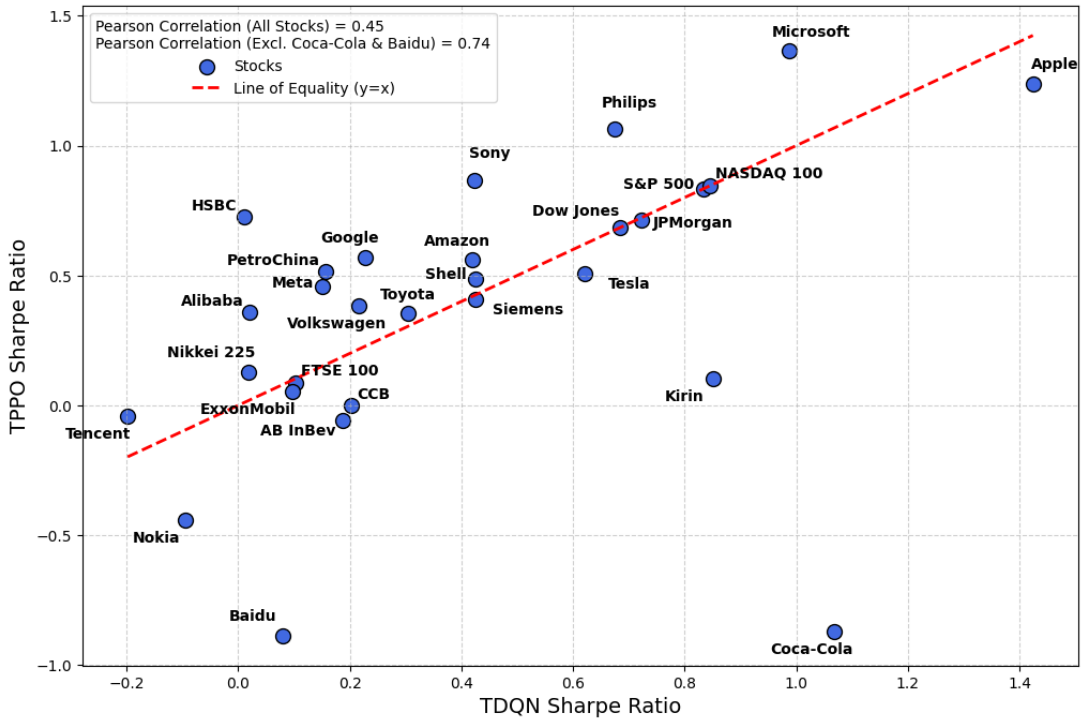

Figure 6.3: Comparison of TDQN and TPPO Sharpe Ratios by Stock

The Implications of Correlation

The Pearson correlation coefficient across all stocks was r=0.45, indicating a moderate correlation between the two models' performances. However, this coefficient seems to mask key concerns evident in the underlying testbench results. The previously highlighted similarity in results between TPPO and TDQN may conclude that the models have converged on similar strategies, despite their differing architectures. This is very evident from the identical results across the first three currency markets, where both DRL models matched the passive Buy Hold (BH) strategy. This convergence likely stems from the conservative hyperparameters, causing the models to default to the most accessible but suboptimal strategies such as B&H. When outliers like Coca-Cola (KO) and Baidu (BIDU) are excluded, the correlation increases to r=0.74, further reinforcing the alignment between the models. This suggests that the framework's constraints may be limiting both models' ability to explore alternative strategies, particularly for the more complex TPPO model.

Convergence to Local Minima

It is common for DRL models to converge on similar, nuanced trading strategies for stock-specific cases that are highly profitable, as observed in NASDAQ, SP 500, JPMorgan, and others in our testbench. However, the consistent convergence seen among lower Sharpe ratio stocks—and even stocks where both models produce negative Sharpe ratios—raises concerns.

As illustrated in Figure 6.4, the Sharpe ratios for several stocks in the TPPO algorithm, including the four examples shown, indicate a clear learning plateau during both the training and testing phases. This suggests that the conservative hyperparameter tuning applied to prevent overfitting and promote

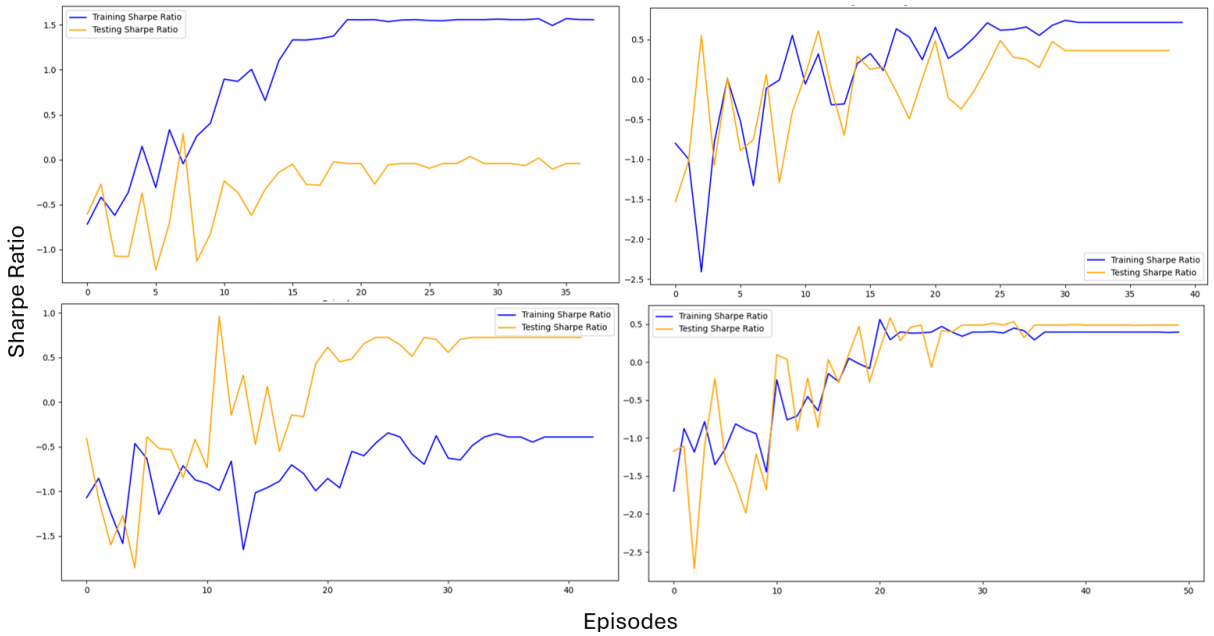

Figure 6.4: Training and Testing Performance of a Sample Batch of Stocks (Tencent, AliBaba, HSBC, Shell [Left-to-right and Up-to-down])

generalization may have inadvertently restricted the models' capacity to explore and adapt to more complex market dynamics. While this conservative approach maintained stability, it also constrained the models' ability to develop more sophisticated trading strategies, often defaulting to simpler, suboptimal strategies that achieved consistent but limited performance outcomes. For the TPPO model, this issue is particularly pronounced due to its complexity. The model's architecture, designed to capture intricate market dynamics, requires more careful tuning. Without such tuning, the complexity of TPPO increases its susceptibility to converging on local minima. This highlights the delicate balance needed when tuning advanced models like TPPO to ensure they leverage their complexity effectively without becoming trapped in suboptimal strategies.

Methodological Constraints and Their Impact

The fact that two distinct models like TDQN and TPPO produced similar results suggests that the constraints of the hyperparameter tuning framework and the heightened focus on generalization may have inadvertently limited the models' ability to fully leverage their unique strengths. The imposed hyperparameter tuning, designed to prevent overfitting, likely diminished their exploratory capabilities that are crucial for the effectiveness of DRL models. Consequently, both models tended to settle on safer, more generalized strategies which, while consistent, may not have fully exploited the potential of each model's architecture.

This realization highlights a key limitation of the project's approach. The similarity in outcomes, despite the different theoretical foundations of TDQN and TPPO, suggests that the framework may have been overly restrictive, leading to an unintended convergence of results. This raises important questions about the extent to which the framework and tuning process supported or hindered the models' ability to adapt and perform optimally across diverse market conditions.

6.2 Critical Analysis

6.2.1 Hyperparameter Tuning and its Trade-offs

The primary objective of this project was to develop models capable of robust and consistent performance across a diverse set of stocks, as measured by high overall Sharpe ratios. To achieve this, a rigorous hyperparameter tuning process was implemented, aiming to generalize well across various market conditions. While this approach effectively met the project's defined goals, it also exposed significant limitations when considering the results for real-world applications in algorithmic trading. The emphasis on broad

generalization required a hyperparameter selection process that often prioritized stability over adaptability. This "one-size-fits-all" approach led the models to converge on accessible but suboptimal solutions, known as local minima. This process inadvertently diminished the DRL models' capacity for exploration, limiting their ability to adapt dynamically to significant market regime shifts—a crucial requirement in real-world trading.

6.2.2 Generalization vs. Specialization in DRL

For instance, in the case of Coca-Cola (KO), the TPPO model's conservative hyperparameter tuning led to a lack of effective trading activity given its significant underfitting. This came as the model notably misinterpreted the short-term fluctuations as noise, preventing it from capturing the straightforward upward trend in the market. Despite the stock's consistent performance, the model missed opportunities to enter profitable trades, largely due to its overly cautious settings that hindered adequate learning from market signals. Conversely, in Baidu's (BIDU) case, the model exhibited overfitting, where it became overly sensitive to the highly volatile patterns in the training data. As a result, the TPPO model's performance deteriorated when market conditions shifted, failing to generalize to the new market environment. This contrast between Coca-Cola's underfitting and Baidu's overfitting highlights the model's difficulty in balancing trade-off between learning too little and over-learning specific patterns, which negatively affected its ability to adapt to diverse market conditions. These cases underscore the critical gap between the project's objectives and the real-world application of these models. The ability to dynamically adapt to specific market conditions and capitalize on unique opportunities is often more valuable than striving for uniform performance across a wide range of scenarios, especially in environments as volatile as financial markets.

6.2.3 Model Simplification and the Risk of Convergence

The observed convergence between the TDQN and TPPO models towards similar local minima can also be partly attributed to the simplifications made in the project's design. By abstracting away real-world complexities such as transaction costs, liquidity constraints, and slippage, the models operated in a more controlled and idealized environment. This likely reduced the differentiation between the models' performances, as they were not required to navigate the complexities that would otherwise drive distinct strategies. Eventually leading them to cases with almost identical trading strategies and performances.

While these simplifications were necessary to focus on the core objectives of the project and prevent creating an opaque black box process, they also reveal a limitation in the findings. In practical algorithmic trading, these omitted factors play a critical role in shaping the success of a strategy. The convergence observed in the simplified environment might not hold in a more complex, real-world scenario where these factors would require the models to differentiate their approaches more significantly. For example, in real-world trading, models must account for the impact of large trades on market prices, transaction costs, and the varying liquidity of assets. These factors could force the models to adopt more diverse strategies, potentially leading to less convergence and greater differentiation in performance.

6.2.4 Real-World Application of TPPO

In real-world trading, the necessity for models to maintain flexibility and adapt to specific market nuances cannot be understated. The limitations observed in the performance of the TPPO model, driven by a hyperparameter tuning process that favored stability over adaptability, underscore a critical tension in DRL applications: the balance between achieving robustness across diverse conditions and preserving the model's capacity for innovation in response to distinct market events.

Despite these challenges, the TPPO model showed promise, particularly in its superior performance over TDQN in specific instances, outperforming TDQN in 14 out of 29 stock cases. This suggests that TPPO's more complex architecture has the potential to capture and exploit more intricate market patterns when the conditions are right. However, the difficulty in tuning hyperparameters for such a complex model means that its real-world potential remains somewhat unproven. The inability to fully optimize these hyperparameters likely prevented TPPO from consistently demonstrating its capabilities across the entire testbench. Future iterations of the model should aim to achieve a better balance between generalization and specialization. By implementing a more nuanced hyperparameter optimization strategies, TPPO could better capitalize on the flexibility and adaptability necessary to succeed in the complex and dynamic landscape of real-world financial markets.

Chapter 7

Conclusion

7.1 Thesis Overview

This thesis set out to develop and evaluate a novel Deep Reinforcement Learning (DRL) model, the Trading Proximal Policy Optimization (TPPO) algorithm, aimed at achieving robust generalization across diverse market conditions and addressing the challenges faced by the comparative TDQN algorithm. Through rigorous testing, the TPPO model demonstrated its potential to outperform traditional strategies such as Buy Hold (BH) and in certain scenarios, the Trading Deep Q-Network (TDQN), validating the newly proposed DRL model's capability to capture complex market dynamics.

However, the emphasis on broad generalization revealed significant challenges in this study. The considered TDQN and TPPO models exhibited tendencies to converge to local minima, particularly due to conservative hyperparameter tuning, often leading to suboptimal performance in dynamic market conditions. This trade-off between stability and adaptability underscored both the complexity of effectively tuning DRL models in real-world trading and the limitations of the employed simplified framework to properly test the capabilities of the proposed DRL models.

7.1.1 Redefining the Methodology: Including Insights from Lin and Beling

While the methodological framework used in this thesis provided a controlled environment for model evaluation, it also imposed constraints that limited the models' exploratory capabilities. The conservative hyperparameter tuning approach, aimed at preventing both overfitting and underfitting, contributed to convergence issues and highlighted the need for a more robust framework to rigorously test DRL algorithms.

A key lesson from this research is the necessity to rethink and adapt the methodology for employing DRL-based algorithmic trading. [12] approach presents a compelling case for more flexible and adaptive tuning, specifically focusing on a representative stock before attempting generalization. This strategy could better balance exploration and exploitation within specific market conditions without the immediate pressure to generalize.

After optimizing for a representative scenario, the model's generalization capabilities could then be rigorously tested across various stocks and market environments. This adjusted methodology could mitigate challenges such as models converging to local minima when primarily tuned for broad generalization. By first establishing a strong, representative stock-specific foundation, future research could more effectively assess and enhance the model's ability to generalize, leading to the development of more robust and reliable DRL models for true investigations of their feasibility for real-world trading applications.

An important factor to consider in this evaluation is the sensitivity DRL models have to random weight initialization. The analysis in this thesis is based on the results from seed 6, which offered valuable insights into the model's behavior. However, different initializations can lead to variability in performance outcomes, as demonstrated by the Nokia results, where different seeds produced noticeably distinct results even while considering overly conservative hyperparameters. This suggests that while the TPPO model's hyperparameters were tuned for general stability, its performance still remained slightly sensitive to initialisation conditions. This highlights a key oversight in the potential of conservative hyperparameters in suppressing the randomness present in DRL models which was imposed within this

methodology. Techniques such as weight regularization or smarter weight initialization strategies (e.g., Xavier or He initialization) would be more effective and could help mitigate this sensitivity and reduce performance variability across different initializations. Incorporating these strategies into the adapted methodology would enhance the model's robustness, ensuring more consistent performance and reducing the likelihood of suboptimal initializations affecting the model's ability to converge on optimal trading strategies.

Following the adaptation of the research methodology, future work could focus on refining the models based on their demonstrated generalization capabilities. Once the model's performance has been rigorously evaluated across various market environments, adjustments could be made to further the adaptability of the proposed DRL models. A few of these considerations are discussed in the following section.

7.2 Future Directions for enhancing the DRL models

Redefining the methodology opens up several avenues for future research to address challenges faced by DRL models, such as partial observability and market regime shifts:

- **Advanced Model Architectures:** Integrating more advanced architectures, such as transformers, could further improve the DRL model's abilities to handle temporal patterns in market data, thereby enhancing its capacity to generalize across diverse market conditions following a representative tuning process.
- **Incorporating Realistic Market Dynamics:** Future research should include factors like transaction costs, liquidity constraints, and market impact in the evaluation framework. These additions would provide a more nuanced understanding of the model's performance and enhance the real-world applicability of DRL models like TPPO.
- **Specialized Models:** Developing a portfolio of sub-models tailored to specific stocks or market conditions could optimize performance by balancing generalization with specialization. This approach could mitigate the risks of model convergence to local minima observed in the current study.
- **Improved Model Robustness:** Introduce robustness testing during the training phases by simulating extreme market conditions in the training data. This will help ensure the model can manage vulnerabilities, particularly those associated with market regime shifts.
- **Adaptive Learning Strategies:** Implementing adaptive learning strategies that dynamically adjust the model's parameters in response to changing market conditions could enhance the model's effectiveness across diverse environments.
- **Live Testing and Longitudinal Studies:** To ensure the robustness and adaptability of DRL models in real-world scenarios, future research should include live testing and longitudinal studies. These studies would provide valuable insights into the models' long-term performance and adaptability.

7.3 Final Thoughts

This thesis has contributed significantly to algorithmic trading by developing and evaluating the TPPO model, demonstrating both the potential and challenges of implementing deep reinforcement learning (DRL) into algorithmic trading. The research highlights the need for a more refined methodological framework that can more effectively test and optimize DRL models. Key considerations include achieving a balance between generalization and specialization in hyperparameter tuning, as well as addressing the impact of weight initialization in neural networks. Future research should prioritize tailoring models to specific market conditions before generalizing, adopt adaptive learning strategies, and incorporate more realistic market dynamics. These steps will be essential for advancing the robustness and effectiveness of DRL algorithms and developing more reliable trading strategies.

Bibliography

[1] Wenbin Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long short-term memory. *PLOS ONE*, 12(7):e0180944, 2017.

[2] Dimitris Bertsimas and Andrew W. Lo. Optimal control of execution costs. *Journal of Financial Markets*, 1(1):1–50, 1998.

[3] Ernest P. Chan. *Quantitative Trading: How to Build Your Own Algorithmic Trading Business.* Wiley, 2009.

[4] Ernest P. Chan. *Algorithmic Trading: Winning Strategies and Their Rationale.* Wiley, 2013.

[5] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[7] Peter J. Huber. *Robust Estimation of a Location Parameter*, pages 492–518. Springer, New York, NY, 1992.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[9] Petter N. Kolm and Gordon Ritter. *Machine Learning in Financial Markets: A Guide to Contemporary Practice.* Routledge, 2019.

[10] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, pages 1008–1014, 2000.

[11] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[12] Siyu Lin and Peter A. Beling. An end-to-end optimal trade execution framework based on proximal policy optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 4548–4554. International Joint Conferences on Artificial Intelligence Organization, 2020.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[14] Rishi K. Narang. *Inside the Black Box: A Simple Guide to Quantitative and High Frequency Trading.* Wiley, 2009.

[15] Yevgeniy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 673–680, 2006.

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[17] Eric Smith, J. Doyne Farmer, László Gillemot, and Supriya Krishnamurthy. Statistical theory of the continuous double auction. *Quantitative Finance*, 3(6):481–514, 2003.

[18] Thibaut Théate and Damien Ernst. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632, 2021.

[19] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2094–2100, 2016.

[20] Christopher Watkins. Learning from delayed rewards. In *PhD Thesis, University of Cambridge*, 1989.

[21] Yahoo Finance. Historical stock price data for multiple companies. https://finance.yahoo.com/. Accessed: August 15, 2024.

[22] H. Zhang and J. Li. Recent advances in deep reinforcement learning for financial applications. *International Journal of Financial Studies*, 9(3):44–57, 2021.

Appendix A

Appendix

- Simulation transcript for Coca Cola and Baidu including their derived training and testing performance assessments:

```
Running simulation for stock Baidu with seed 6...
    Initializing TradingEnv with marketSymbol: BIDU, startDate: 2012-1-1, endDate: 2018-1-1, fre
    2 and 117
    Initializing PPO with the following hyperparameters:
    Learning Rate (Actor): 0.001, Learning Rate (Critic): 0.001, Dropout Rate: 0.5, Weight Decay
    Initializing TradingEnv with marketSymbol: BIDU, startDate: 2018-1-1, endDate: 2020-1-1, fre
    Testing environment initialized with data shape: (503, 11)
    Testing environment initialized successfully.

    Training finished.
    Plotting training and testing performance...
    Performance plot saved.
    Displaying final performance metrics on training environment...
    Sharpe Ratio: 0.41422620864259146

       Performance Indicator      PPO

        Profit & Loss (P&L)      64896

          Annualized Return       10.91%

       Annualized Volatility     34.66%

           Sharpe Ratio          0.414

           Sortino Ratio         0.620

         Maximum Drawdown        53.51%

     Maximum Drawdown Duration  312 days

           Profitability         61.54%

     Ratio Average Profit/Loss   0.927

             Skewness            0.060

    Sharpe Ratio: -0.8859521664067459

       Performance Indicator      PPO
```

```
    Profit & Loss (P&L)      -53119

     Annualized Return      -39.20%

   Annualized Volatility     35.59%

       Sharpe Ratio          -0.886

       Sortino Ratio         -1.154

     Maximum Drawdown        70.05%

 Maximum Drawdown Duration  351 days

       Profitability         47.06%

 Ratio Average Profit/Loss   0.318

         Skewness            -0.509

Simulation for stock Baidu with seed 6 completed.

Initializing TradingEnv with marketSymbol: KO, startDate: 2012-1-1, endDate: 2018-1-1, frequ
2 and 117
Initializing PPO with the following hyperparameters:
Learning Rate (Actor): 0.001, Learning Rate (Critic): 0.001, Dropout Rate: 0.5, Weight Decay
Initializing TradingEnv with marketSymbol: KO, startDate: 2018-1-1, endDate: 2020-1-1, frequ
Testing environment initialized with data shape: (503, 11)
Training finished.
Plotting training and testing performance...
Performance plot saved.
Displaying final performance metrics on training environment...
Sharpe Ratio: -0.4044551637540351

   Performance Indicator       PPO

    Profit & Loss (P&L)      -61178

     Annualized Return       -18.17%

   Annualized Volatility     28.87%

       Sharpe Ratio          -0.404

       Sortino Ratio         -0.659

     Maximum Drawdown        65.26%

 Maximum Drawdown Duration  1477 days

       Profitability          0.00%

 Ratio Average Profit/Loss    0.000

         Skewness             1.087

Sharpe Ratio: -0.8705995524150402
```

Performance Indicator	PPO
Profit & Loss (P&L)	-34020
Annualized Return	-20.71%
Annualized Volatility	21.33%
Sharpe Ratio	-0.871
Sortino Ratio	-1.277
Maximum Drawdown	37.07%
Maximum Drawdown Duration	407 days
Profitability	0.00%
Ratio Average Profit/Loss	0.000
Skewness	0.858

Simulation for stock Coca Cola with seed 6 completed.