TRADING ON GAMBLING MARKETS: A REINFORCEMENT LEARNING APPROACH

George Maksour

Supervisor: Dr Rohitash Chandra

School of Mathematics and Statistics
UNSW Sydney

May 2023

Submitted in partial fulfillment of the requirements of the degree of
Bachelor of Science with Honours

Plagiarism statement

I declare that this thesis is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere.

I acknowledge that the assessor of this thesis may, for the purpose of assessing it:

- Reproduce it and provide a copy to another member of the University; and/or,
- Communicate a copy of it to a plagiarism checking service (which may then retain a copy of it on its database for the purpose of future plagiarism checking).

I certify that I have read and understood the University Rules in respect of Student Academic Misconduct, and am aware of any potential plagiarism penalties which may apply.

By signing this declaration I am agreeing to the statements and conditions above.

Signed: George Maksour Date: 20-4-23

Acknowledgements

I would like to thank the University of New South Wales, Sydney for giving me this excellent opportunity to write my undergraduate thesis at their School of Mathematics and Statistics.

I would like to express my gratitude towards my supervisor, Dr Rohitash Chandra, for always believing in my capabilities. His constant guidance and support have inspired and motivated me. I also want to thank Dr Sahani Pathiraja for her constant help and technical knowledge of my subject matter, her knowledge really helped me get on the right track.

Last but not least, I would like to thank my parents, my girlfriend and my friends for their constant support throughout the project. Without the help and contribution of the above-mentioned people, this project would not have been a success.

George Maksour, 20 April 2023.

Abstract

As market-style gambling is a rapidly growing field, there is potential to translate and improve predictive financial systems with many novelties. Deep reinforcement learning has become[e prominent in solving sequential decision problems, such as trading. The purpose of this thesis is to develop a framework for DRL agents to converge on a gambling exchange and generate profits by addressing the problem of automated racing trading. Through the evaluation of deep reinforcement learning models in pre-match markets in thoroughbred and greyhound races, this objective is achieved. We evaluate actor-critic, deep-q networks, policy-proximal optimisation and actor-critic with experience replay and reinforcement learning models, and report on convergence and profitability to compare the agents. We create a unique framework to set up an environment for learners mimicking a live trading environment. We find that the policy proximal optimisation algorithm is the most efficient learner for converging on a policy in these markets. In terms of risk-adjusted returns, deep Q-networks perform best.

Contents

List of Tables

List of Figures

Acronyms

A2C Actor-Critic
ACER Actor critic with experience replay
CVAR conditional value at risk
DQN Deep Q-Network
DRL Deep reinforcement learning
EMA Exponential moving average
FFNN Feed forward neural network
LSTM Long short-term memory
MACD Moving average convergence divergence
MDP Markov decision process
MLP Multi-layer perception
PPO Proximal policy optimisation
WOM Weight of money

Chapter 1

Introduction

1.1 Motivation

Market-style gambling has existed since the early 2000s, with the start of the *Betfair* exchange [20]. This form of gambling is a unique concept where users set odds for markets and trade on these prices. This unique democratization of bookmaking has led to interest from traders, retail bettors, bookmakers, and hedge funds [74], and has become akin to trading in financial markets [99]. Recent popularity can be attributed to API integration, in conjunction with the significant amount of (un)structured data related to sports events that are produced at present [40], which allows for the application of machine learning and statistical models [45]. The rise in interest in these markets has led to the problem of devising profitable trading strategies. These fall into two main categories. The first, identifying the inefficiencies in prices that are offered by other punters [12], and taking on odds that have a positive expected value. This is evident across a number of events including horse-racing [98, 11, 39], greyhound racing [78], tennis [81] and soccer [89]. Alternatively, a trader can make trades on price fluctuations before an event occurs, allowing a trader to arbitrage their stake and guarantee a profit before the event occurs. This alternative has recently become more popular in soccer [17] and racing [24]. Trading on price fluctuations requires the trader to continually make sequential decisions, making predictions on the direction of each proposition.

In recent years there has been a surge in the application of deep reinforcement learning Deep reinforcement learning (DRL) [50] to trading [108]. A significant area of progress in DRL is the ability to solve sequential decision problems. DRL has shown a significant capability for learning certain sequential decision-making strategies including for video games [61], robotics [56, 47], and financial markets [88, 31]. Given the unsolved task of continual trading on markets and an unused model befitting the problem, this thesis applies the task of market-style gambling to deep reinforcement learning. In doing so, analyse the objectives of convertibility, profitability and improvements on previous methods. Achieving these can give great insight into generalising gambling trading, developing the input space and live trading.

1.2 Gambling exchanges

A betting market is an example of a simple financial market [99]. Gambling exchanges are a relatively new concept in the world of gambling. Unlike traditional bookmakers who set odds and take bets from customers, betting exchanges facilitate peer-to-peer betting. It is found that these betting exchanges exhibit better

predictive power [28] and better odds for punters. The most popular of these exchanges is *Betfair*. For each betting event, there exists a set of propositions that a user can bet on. A user can place a back bet at certain odds, which represents a bet that will be successful if the outcome occurs. However, a lay bet represents the opposite, taking a price that is successful if the outcome is not in the proposition's favour. A user can place a number of bets across a single market at any price with any volume or take another user's odds, which represent the opposite side of the event. As seen in Table 1, there exists a range of back and lay prices and stated volumes at each price. These odds will fluctuate for different reasons including algorithms, market makers and retail (casual) bettors. As seen in Figure 1.1 market makers provide liquidity and algorithm and retail bettors will exchange odds more often as the event comes closer to occurring.

Proposition		Best Back Price	Best Lay Price	
West Ham	3.64 ($32)	3.66 ($74)	3.7 ($66)	3.72 ($47)
Draw	3.84 ($35)	3.88 ($57)	3.94 ($43)	3.96 ($12)
Newcastle	2.3 ($20)	2.31 ($102)	2.33 ($82)	2.34 ($56)

Table 1.1: Example of exchange layout, odds represented in European style and volume in brackets

The market is not perfectly efficient, as there are a set of biases that bettors have in odds setting and odds taking [33]. These lead to inefficiencies in price which drive the market, and is important to our problem of trading on price because it gives significance to movement within the market. As suggested by [19] the search for market inefficiencies must be automated as manual processes are too slow at finding and exploiting these opportunities. An important bias is that bettors overreact to noisy outcomes and put too much weight on the past, as opposed to recent performance when forming expectations about future outcomes [15, 49]. In multiple codes including football [13], NFL [27] and horse-racing [57]. Further, a well-studied bias in betting markets is the long-shot bias [82]. The long-shot bias refers to the tendency for people to overestimate the likelihood of rare or unlikely events occurring while underestimating the likelihood of more common events happening. This bias is observed in tennis [1] and is important for our research in horse-racing [101]. Moreover, the bias increased over time,f a result consistent with attributions for success and failure that rewarded intuitive choosing [79]. These inefficiencies allow for movements of price within the market so that directional trading can occur, allowing us to explore the application of reinforcement learning on this problem within the market.

1.3 Deep reinforcement learning for trading

1.3.1 Trading on gambling markets

Whilst there is much research surrounding artificial intelligence in betting exchanges, little is known about trading on price fluctuations. The closest area of research that can be made to the objectives of this thesis is deep reinforcement learning in financial markets. This is due to the similarities between the financial

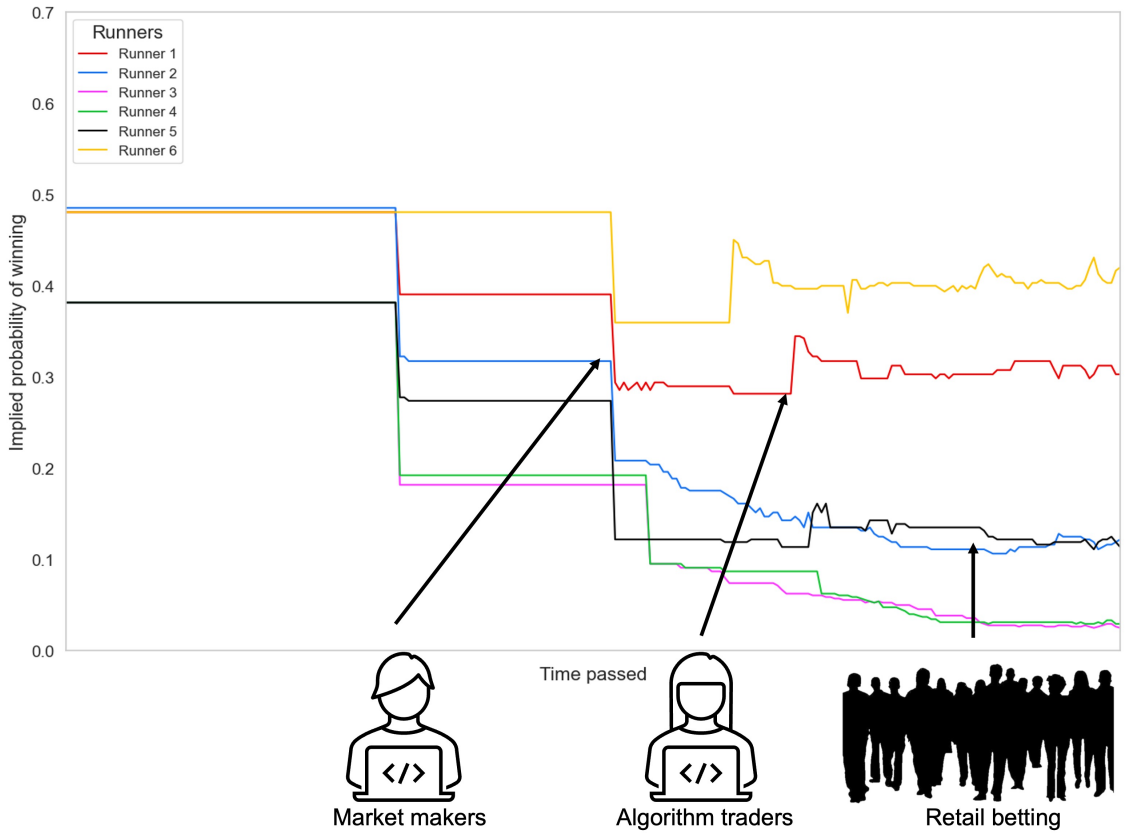


Figure 1.1: Example of Betfair greyhound prices with 5-second intervals representing the best back prices of multiple racers in a greyhound field

and gambling markets. Within the applications of AI in gambling, there are many papers that focus on the pricing of outcomes. This involves using statistical imputation of price and placing bets that yield positive expected value across sports [106, 58]. These papers focus on taking the correct odds but do not focus on trading on odds. The only study conducted on price-trading with AI [24] uses a Cartesian genetic programming and artificial neural network model on the racing markets. However, this problem falls short of using a static predictive model to apply price inputs of N minutes and predict direction. The purpose of our thesis is to build on this, allowing a model to sequentially trade throughout the last 20 minutes of a market's opening.

1.3.2 Trading on financial markets

Outside of market-style gambling papers with similar method formulation are DRL-applied financial papers. As there are many similarities between trading in the financial markets and market style-gambling. Both markets, have the same signals for volume and price and can be evaluated within the same context [35]. Similarly, the state actions for sell, hold and buy respectively can be transferred to lay, back, and hold. Therefore, when creating a Markov decision process the state, action and reward space are transferable. There are many papers within the financial deep reinforcement learning field as summarised by [59]. The methods within this thesis differentiate themselves across four different categories: data, state space, action

space and rewards. Papers on deep reinforcement learning markets in financial trading differ in policy and Markov decision process (MDP). We compare multiple policies of Long short-term memory (LSTM) and Feed forward neural network (FFNN). Analysing the policy decisions, many deep reinforcement learning papers have used LSTM models over FFN within their deep reinforcement learning due to the time-series-based nature of the price of stocks [102] and [110]. They generally find LSTMs outperform an FFN due to the memory property of the model in time series data.

Furthermore, there are different proposed methods within data time stamps. The research predominately analyses equities with hour-interval time stamps, filtering out noise in price. There also exists research in interval structure between 15 seconds and 1 minute [84, 96]. Each paper explores profitable agents in these smaller intervals. We propose intervals smaller at 5 seconds. This would work differently in market-style gambling, as markets tend to be more deterministic. Additionally, the range of time periods used in training these models vary. The range of trading times varies between 61 days to 14 years [93] on specific assets, including ETFs [43], portfolios of equities [88] and cryptocurrencies [54]. There are similarities between financial trading and market-style gambling, however, the research does not provide a sufficient framework of the learning environment to comprehensively teach a gambling market agent.

Many of MDPs features are transferable from financial trading to market-style gambling. Within the action space, the majority of research follows the discrete set buy, sell, hold [91] or buy, sell [83]. These discretized actions limit the models to a 2 or 3-dimensional action space, and train on equities and currencies individually rather than using a portfolio system. The state space is similar to market-style gambling, and most papers use price, volume and some of their derivatives across time to convey information to the learners. Many financial indicators are relevant to gambling markets, such as Moving average convergence divergence (MACD), Exponential moving average (EMA) and on-balance volume, and research shows that these indicators have been successful in a learner's profitability [103, 104], making these state spaces useful for traders making decisions on gambling markets. Further, the reward spaces are a mixture of financial returns and ratios. An effective approach was [91] using profit and loss, which this thesis explores. However, other approaches that are included and should be explored in future works are the Sharpe ratio, Sortino ratio and log cumulative return [97, 103, 102]. These papers give insight into the framework and methodology for building a trading system and transferring it to market-style gambling.

1.4 Aim and objective of the study

As market-style gambling is a rapidly growing field, there is potential to translate and improve predictive financial systems with many novelties. This research will create a novel DRL framework to generate profit on a gambling exchange that builds on previous techniques [24] to trade in a new market. We investigate the convergence and profitability of these learners within racing markets and compare

these learners providing insight into performance. This is done by treating the fluctuation of racers in an event as an equity that can be traded. In achieving this, we design and present a novel framework that includes state and reward spaces unique to market-style gambling. The designed framework further presents new computational, time and data challenges that are not dealt with in the financial markets for deep reinforcement learning. We find that learners in market-style gambling environments can generalise optimal decisions to make profitable decisions in the environment, even though the stochastic nature of trading implies some randomness to payoffs. This thesis contributes to the research on deep reinforcement learning by investigating its utility in further applications. This allows for future work to be done on testing DRL agents in a live setting facing further issues including adverse selection, limited volume, and latency. Creating a more generalised agent that could effectively be used as a trader.

1.5 Approach

The problem of trading on market-style gambling can be viewed as a sequential decision problem to which we apply DRL agents. To investigate trading on the price, we select WIN racing markets 20 minutes prior to the event beginning. The attraction to horse-racing is the large volumes and smaller spread in the market, which causes susceptibility to heavier fluctuations. The agent will have access to price and volume information, including novel representations of this information, and will choose to either lay or hold these propositions within the market. The reward will be derived from the profitability of each model. Using an open-sourced Python implementation, we apply five different DRL algorithms, including Proximal policy optimisation (PPO) [77], Deep Q-Network (DQN) [61], Actor-Critic (A2C) [60] and Actor critic with experience replay (ACER) [95]. By comparing this representation across models, it can be determined which models are best suited for a trading environment.

Chapter 2

Review of techniques

2.1 The Markov decision process

Considering the stochastic nature and the sequential decision problem of trading on gambling markets, we formulate the problem as a MDP [69]. The idea is to assign different aspects of the data to the state space tuple. The notation will set out the way to quantify a learner's interaction with a trading environment. An MDP for reinforcement learning [85] is defined as a 5-tuple (S, A, T_a, R_a, γ);

1. S is a set of legal states of the environment.
2. A is a finite set of actions.
3. $T_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action a in state s at time t will transition to state s‘ at time $t + 1$ in the environment. This is also known as the trace function.
4. $R_a(s, s')$ is the reward received after a transition from state s to s'.
5. γ is the discount factor representing the difference between the future and present rewards.

A policy π in the Markov decision process answers how different actions in different states should be chosen. In a limit to the computational capacity of collecting data, the MDP will be represented as a discrete-time MDP. The policy π is a conditional probability distribution that for each possible state specifies the probability of each action $\pi : S \to p(A)$. For a particular probability distribution, the policy is written as $\pi(a|s)$. The measure of success of a policy is the expected cumulative discounted future reward known as the value of the state. The value of policy π is defined as;

$$V^{\pi}(s) = \mathbb{E}_{\tau \approx p(\tau_t)} \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t = s \right], \tag{2.1.1}$$

Every policy has a unique associated value function $V^{\pi}(s)$. The goal of the MDP is to find the optimal policy based on value, when there is a large range of actions and states, as is the nature of trading. Policy and value can be estimated and optimised through different policy and value evaluation methods, as it is computationally infeasible to tabulate the probabilities for every state and reward within a trading environment. This thesis will define a set of states, actions and reward structures to deal with the task of trading.

2.2 Deep policy evaluation

To simplify the environment throughout the course of market trading the research will compare two policy approximators, the Multi-layer perception (MLP) [68] and the LSTM model. These approximators will be compared across the applicable models on the same trading data.

2.2.1 Multi-layer perceptron

An MLP is made up of linked layers of nodes. This creates a series of layers that transforms a group of inputs into an output using a process of vector multiplications. The main goal of an MLP is to approximate a global function with respect to parameters θ and the data x where θ are the weights. As such the formula for approximating y is represented as,

$$f(x) = b + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n, \tag{2.2.1}$$

where b represents a bias term. This is represented in matrix notation as,

$$f(x) = \Theta^T X \quad \text{where} \quad \Theta = \begin{bmatrix} b \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}, \tag{2.2.2}$$

The parameters are randomly set and are updated in the process of training through back-propagation given a predefined error function E,

$$\theta^{t+1} = \theta^t - \eta \frac{\partial E(X, \theta^t)}{\partial \theta}, \tag{2.2.3}$$

The novel aspect of an MLP is its ability to approximate non-linear functions, this is done through the use of non-linear activation functions such as ReLu [2], hyperbolic tangent and logistic function. The activation function σ is represented by,

$$f(x) = \sigma(\Theta^T X), \tag{2.2.4}$$

The use of an MLP in DRL is to generalise the complex environment of trading and distilling information for the agent. In simplifying the environment the agent

can approximate the optimal solution improving computational speed and training times.

2.2.2 Long-short term memory

The LSTM is similar to the MLP however the modification is allowing for previous data to be used as part of the inputs. Looking over previous inputs and using them in generalisation gives an LSTM the property of memory. The memory property is helpful in a trading environment as a trader will make sequential time-based decisions.

The LSTM contains three gates, an input, output and forget gate, respectively represented,

$$\begin{aligned} i_t &= \sigma(w_i[h_{t-1}, x_t] + b_i), \\ o_t &= \sigma(w_o[h_{t-1}, x_t] + b_o), \\ f_t &= \sigma(w_f[h_{t-1}, x_t] + b_f), \end{aligned}$$

Where h_{t-1} is the output of the previous block and w_t is the weight of the respective gate. The output of these gates is compiled as,

$$\begin{aligned} \tilde{c}_t &= tanh(w_c[h_{t-1}, x_t] + b_c) \\ c_t &= f_t c_{t-1} + i_t \tilde{c}_t \\ h_t &= o_t tanh(c^t) \end{aligned}$$

where $\tilde{c}_t$ is the candidate for cell state at timestamp (t) and c_t is the cell output. The LSTMs memory property is useful for remembering time series data used in trading. It will be used to approximate the Q-value function for the A2C and ACER agents.

2.3 Reinforcement learning

Reinforcement learning distinguishes itself from other types of model learning as it learns through constant interaction with an environment. The framework of RL is based on an MDP. This is important in a market-style gambling environment as sequential decision-making is imperative in placing trades on an exchange. The objective of reinforcement learning is to find the optimal action for each state maximising future rewards [5]. There are innumerable applications of these learners, such as trading, however have never been applied to market-style gambling. There exists learners that utilise pixels on the screen to play ATARI [61], or robots that interact in 3 dimensions [56]. Some earlier papers on sequential decision tasks were successful but were restricted in their use to low dimensional problems due to stability, memory and computational complexity [6, 47, 87]. However, deep learning has recently been proven as a good generalization method in machine learning and allows scalability, and now has become the standard of speech recognition [22], visual object recognition/detection [94]. This allows the problem of market-style of trading to be dealt through a new lens of powerful function approximators and

sequential decision agents. The reinforcement learning algorithms can be divided into two key sections: model-free and model-based. Within this thesis, model-free algorithms are used, as an explicit representation of a market-style environment cannot be created.

2.3.1 Model-free

Model-free reinforcement learning approaches do not use an explicit representation of the environment's dynamics but rely on updates to estimates of values or policies. In model-free reinforcement learning, the agent learns through trial and error, by taking actions in the environment and receiving a reward signal. The goal is to find a policy, or a mapping from states to actions ($s \rightarrow a$), that maximizes the expected cumulative reward. One common approach in model-free reinforcement learning is Q-Learning, which estimates the value of taking a specific action in a specific state. The agent updates its estimates of Q-values based on observed rewards and expected future rewards. Another approach is policy gradient methods [100], which directly estimate the optimal policy by updating the policy parameters through gradient ascent.

The reinforcement learning algorithm maximizes a function $V(s)$. The agent performing will choose action a_t in-state s_t to maximize the long-term return of all current states under the policy. Based on the notion of the Q-function of a policy $Q^{\pi}(s, a)$ (also known as Q-learning) measures the discounted sum of rewards obtained from state s by taking action a first and following policy π thereafter. We define the optimal Q-function $Q^{*}(s, a)$ as the maximum return that can be defined by the Bellman optimality equation,

$$Q^{*}(s, a) = \mathbb{E}[r + \gamma max_{a'} Q^{*}(s', a')], \tag{2.3.1}$$

The Bellman equation is optimised through an iterative update as follows:

$$Q_{i+1}(s, a) = -\mathbb{E}[r + \gamma max_{a'} Q_i(s', a')], \tag{2.3.2}$$

where $Q_i \rightarrow Q^{*}$ as $i \rightarrow \infty$. However, when calculating the optimal policy across actions and spaces that are large there is a significant decrease in the computational performance of the model. Therefore these methods are used to approximate these values over a 2D market-style space efficiently and effectively.

2.3.2 Model-based

Model-based reinforcement learning approaches, on the other hand, explicitly represent the environment's dynamics and use planning algorithms to determine the optimal policy. In model-based reinforcement learning, the agent starts by constructing a model of the environment, which can be in the form of a transition function that maps states and actions to the next states, and a reward function that maps states and actions to rewards. The agent can then use this model to

perform planning, either by searching for a sequence of actions that maximizes the expected reward or by evaluating different policies and choosing the one with the highest expected reward.

2.4 Deep reinforcement learning types

Deep reinforcement learning utilizes deep neural networks as function approximators for the value function or policy. This allows for more complex decision-making in larger state and action spaces. As market-style trading deals with complex states it is necessary to use approximators to deal with these more complex time-dependent spaces. DRL solves the problem of storing tabular values for each V, Q and π, by transforming into θ parameterising functions V_θ, Q_θ and π_θ. The optimal θ is trained to predict each state's best action,

$$\theta^* = \text{argmax}_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right], \tag{2.4.1}$$

$$p_\theta(\tau) = p_\theta(s_1, a_1, \ldots, s_T, a_T), \tag{2.4.2}$$

Thus for every state, there exists an approximated policy for the action. There are two subsections of DRL we explored in this thesis including deep value-based reinforcement learning and value-based reinforcement learning.

2.4.1 Deep-value based

Deep value-based methods in reinforcement learning involve learning the optimal value function directly, while deep policy-based methods learn the optimal policy directly. Otherwise known as Deep Q-learning, a function approximator such as a deep learning model is trained with parameters θ to estimate Q values. Such that $Q(s, a; \theta) \approx Q^*(s, a)$. This can be done by minimizing a loss function at each i state s,

$$L_i(\theta_i) = E_{s,a,r,s' \approx \rho(.)}[(y_i - Q(s, a; \theta_i))^2] \text{ where,} \tag{2.4.3}$$

$$y_i = r + \gamma max_{a'} Q(s', a'; \theta_{i-1}), \tag{2.4.4}$$

y_i is called the temporal difference (TD) target and $y_i - Q$ is labelled as the TD error. The weights θ within the network are iteratively updated using the Bellman equation as follows,

$$\theta_{k+1} = \theta_k - \alpha \Delta_{\theta s' P(s'|s,a)}[(Q_\theta(s,a) - R)^2]|_{\theta=\theta_k}, \tag{2.4.5}$$

where $R = R(s, a, s')$. The most prominent type of value-based learning is the Deep Q-Network (DQN) [62] which will be used in this thesis. A DQN uses this by using an artificial neural network to approximate these values. It leads to faster training times and does not use ground truth data to train the Q-value estimator. However, the model has a tendency to overestimate Q-values, due to the use of one network for both collections of the estimation and ground-truth data for estimation [67].

2.4.2 Deep-policy based

Value-based methods can be more efficient for simple action spaces, while policy-based methods can handle more complex action spaces and provide a more direct approach to finding the optimal policy [67]. Policy-based DRL methods optimize the parameterized policy directly. The policy is also modelled with a parameterized function with respect to θ, $\pi_\theta(a|s)$. The value of the reward function depends on the policy and then an algorithm can be applied to optimize θ for the best reward. The reward function is defined as,

$$J(\theta) = \sum_{s \in S} d^\pi(s) V^\pi(s), \tag{2.4.6}$$

$$= \sum_{s \in S} d^\pi(s) \sum_{a \in A} \pi_\theta(a|s) Q^\pi(s,a), \tag{2.4.7}$$

where $d^\pi(s)$ is the stationary distribution of Markov chain for π_θ. The stationary probability for $\pi_\theta . d^\pi(s) = \lim_{t \to +\infty} P(s_t = s|s_0, \pi_\theta)$ is the probability that $s_t = s$ when starting from s_0 and following policy π_θ for t steps. Policy gradient is used to update θ and is a method of gradient ascent. Policy gradient is the calculation of $\nabla_\theta J(\theta)$. The policy gradient theorem omits the state distribution $d^\pi(.)$ to simplify the gradient calculation.

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_{s \in S} d^\pi(s) \sum_{a \in A} Q^\pi(s,a) \pi_\theta(a|s), \tag{2.4.8}$$

$$\propto \sum_{s \in S} d^\pi(s) \sum_{a \in A} Q^\pi(s,a) \nabla_\theta \pi_\theta(a|s), \tag{2.4.9}$$

There are many policy gradient algorithms that are useful in policy-based learning. In recent years the most prominent methods include proximal policy optimization (PPO) [77], Actor-Critic (A2C) [60], Deep Deterministic Policy Gradient (DDPG) [52] and Twin Delayed Deep Deterministic (TD3) [29]. Each of these addresses an issue within policy gradient derivation and will be reviewed in the results of this thesis.

2.5 Deep reinforcement algorithms

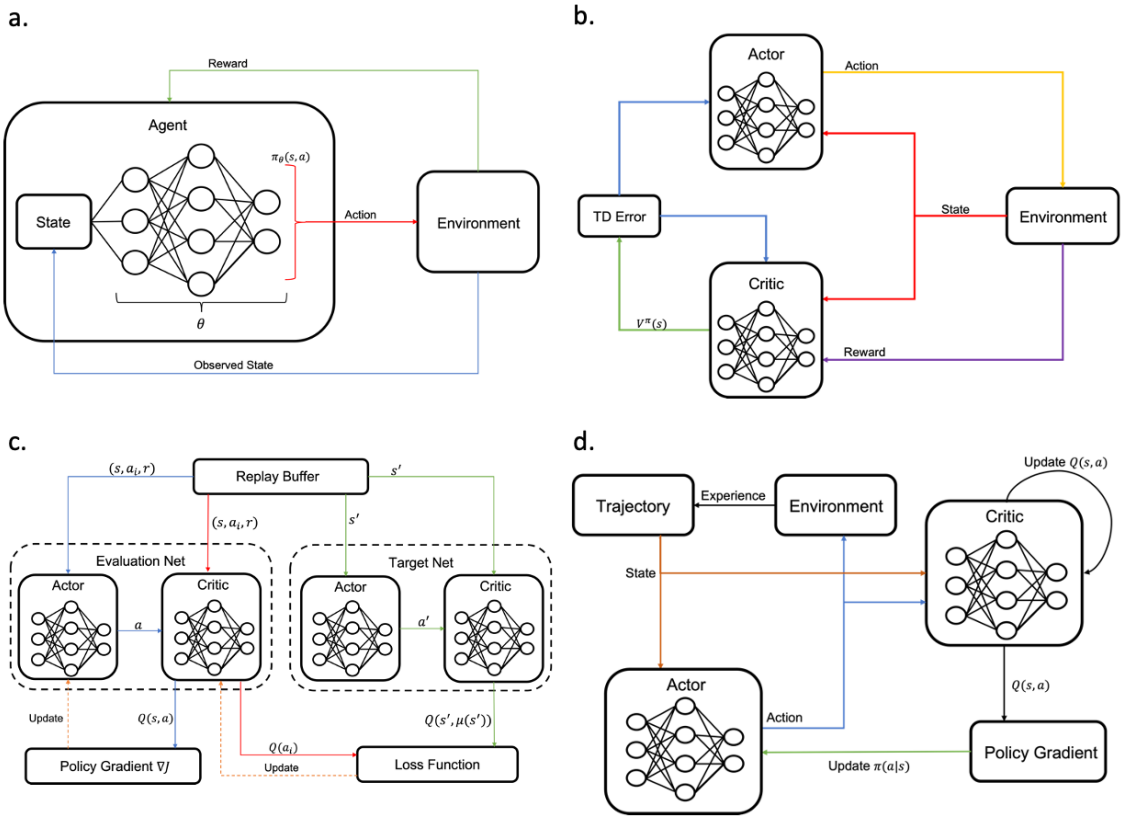


Figure 2.1: Model diagrams of used DRL agents for testing on market-style gambling. **Top left:** Deep Q Network, **Top right:** Actor-critic, **Bottom left:** Actor-critic with experience replay, **Bottom right:** Proximal policy optimisation

2.5.1 Actor-Critic

Actor-Critic, represented in Figure 2.1(a), is an algorithm that combines both policy-based and value-based methods. Simply, it has two components, an actor which determines the actions to take in the environment and, a critic which evaluates the value of taking those actions. In Actor-Critic reinforcement learning, the actor is implemented as a neural network that updates the policy distribution in the direction suggested by the critic. The critic is also implemented as a neural network that takes the state and action as input and outputs an estimate of the value of taking that action in that state. The critic provides a statistic called advantage which is how much better a specific action is compared to the average, general action at the given state. This is calculated through the difference between the Q-value and the state value (V value). As represented by,

$$A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t), \tag{2.5.1}$$

given the bellman optimality equation $Q(s_t, a_t) = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1})]$, the advantage can be rewritten as:

$$A(s_t, a_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t), \tag{2.5.2}$$

Therefore this can be substituted into the policy update equation (2.3.1) as:

$$\nabla_\theta J(\theta) \sim \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t), \tag{2.5.3}$$

The actor and critic are trained simultaneously, with the critic providing feedback to the actor by reinforcing positive and negative actions. This allows the actor to improve its policy over time and converge to the optimal policy. Advantages of Actor-Critic include its ability to handle continuous action spaces, as well as its ability to learn online and in real-time [34]. The disadvantages of Actor-Critic include its high sample complexity, as it often requires a large amount of interaction with the environment to learn a good policy [48]. Additionally, its use of neural networks can make it more challenging to interpret and debug compared to other methods. A2C has been used for sequence prediction [9], and trading [51].

2.5.2 Actor-Critic with experience replay

Actor-Critic with Experience Replay represented in Figure 2.1(c), is an off-policy deep reinforcement learning algorithm that combines the aforementioned is a sample-efficient actor-critic algorithm that uses experience replay, trust region policy optimization method to improve its performance [95]. Experience replay is a replay memory technique used in reinforcement learning where the agent's experience is stored at each time step $e_t = (s_a, a_t, r_t, s_{t+1})$ in a dataset pooled over many episodes into a replay memory. The advantage of this is tackling the problem of auto-correlation that leads to unstable training, by making the problem more like a supervised learning problem [36]. The modified algorithm achieves an increase in convergence speed and sample efficiency compared to both the on-policy actor-critic A2C and the importance-weighted off-policy actor-critic algorithm [86]. The ACER algorithm has been successful in video games, and 2D robotic motion planning [109].

2.5.3 Deep Q-Networks

Deep Q-Networks, represented in Figure 2.1(a), is a DRL algorithm that uses a neural network to approximate the optimal action-value function directly from the environment. The goal of DQN is to find the best action to take in a given state, so as to maximize the expected cumulative reward. As such we define it as Q-learning and given a policy π and function the Q-function is defined as;

$$Q^*(s,a) = \max_\pi \mathbb{E}\Big[R_t | s_t = s, a_t = a, \pi\Big], \tag{2.5.4}$$

In DQN, the neural network is trained to predict the expected reward for taking a specific action in a given state. The training process is done using Q-learning, an off-policy reinforcement learning algorithm, where the neural network is updated based on the difference between its predicted and target values as represented by,

$$L_i(\theta_i) = \mathbb{E}\big(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\big), \tag{2.5.5}$$

The target values are updated periodically using a separate target network, which is a slower-moving copy of the primary network. This helps to stabilize the training process and prevent over-fitting [61]. DQN also uses experience replay, where the transitions (state, action, reward, next state) are stored in a replay buffer [107] and used to update the network randomly, instead of after every step. This breaks the correlation between consecutive samples and improves the stability of the learning process. However, the slower updates mean the DQN takes longer to converge to an optimal solution. DQN has had success in difficult applications including portfolio management [32] and communications networks [55].

2.5.4 Proximal Policy Optimization

Proximal policy optimisation, as represented in Figure 2.1(d) is an algorithm that alternates between sampling data through interaction with the environment and optimising a "surrogate" objective function[77]. The surrogate function enables the creation of a different objective function that updates through multiple epochs of mini-batch cycles. The objective function is derived from the existing method of trust region policy optimisation that constrains the size of the policy update. Using the formula,

$$\max_\theta \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right], \text{ subject to } \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(.|s_t), \pi_\theta(.|s_t)]] < \gamma \tag{2.5.6}$$

where $\hat{A}_t$ is an estimator of the advantage function at time-step t. The policy uses a clipped objective [30] which removes the incentive of a learner to move r_t outside the interval $[1-\epsilon, 1+\epsilon]$ where $\epsilon \in [0, 1]$. Hence proposing,

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \Big[\min(r_t(\theta))\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \Big], \tag{2.5.7}$$

The PPO model has been mentioned to outperform other policy gradient methods in efficiency as it strikes a balance between sample complexity, simplicity and wall time [64]. However, much like other policy gradient methods PPO suffers from some drawbacks compared to value-based learning including inconsistent policy updates and high reward variance. Inconsistent policy updates tend to overshoot and miss the reward peak or stall prematurely whilst high reward variance is due to the nature of Monte-Carlo learning [72]. PPO has been used in successful in multiple fields with success in applications including solving stochastic lot-sizing problems [92] and lane-changing strategy in automated vehicles [105].

Chapter 3

Framework

3.1 Framework overview

The aim of this research is to apply modern techniques to a field that has not made use of them. There are no comparable results of this style of application of DRL. The process of this work is to create an environment that enables a deep reinforcement learning agent to learn to predict and trade on fluctuations within a market. The framework and theory for enabling this process are strictly outlined within this thesis with the aim to identify and explain the failure and success of DRL agents within this field. The computational framework for this methodology is presented in Figure 3.1. The Betfair API provides live data which will then be collected through timed and repeated calls every 5 seconds. This will be pulled through the betfairlightweight package. Each event will be stamped with a time and date. The data will be stored as a set of .csv files locally. The raw data is parsed through a pre-processor that applies selected environment features to the raw prices. The pre-processed data will be used to train 5 different DRL agents with 2 different policies. The diagnostics of each model will be recorded and the overall performance will be represented graphically.

3.2 Application of the Markov decision process

The novelty of gambling markets are the different actions, spaces and rewards that an agent can navigate within a trading environment. The state-space is developed from feature engineering the time series data to be represented within the context of an agent, inspired by previous works' feature engineering. In order to maximize the understanding of the agent within a market-style environment, historical and current price and volume information is taken into consideration. There are other quantifiable variables used in other papers that can be further assessed in future work including, total market percentage, fluctuations of other racers in the market and spreads.

3.2.1 Action space

The action space in an MDP is defined as the set of all legal possible moves a model can make. Within this work, the trader, inspired by [88] is uni-directional and will be only trading 'long' meaning that it will expect profit if the odds fluctuate upwards. Using the setup of [31], the trader will decide to either go long or hold, if the decision is to go long it will be able to choose a stake ratio of discrete divisors of 5 $A \in \{0.25, 0.5, 0.75, 1\}$. Once the trader has initiated a trade it will be until it chooses to go long again to close out the position. The position will be closed in full using the previously staked amount to close it out and a new position will be

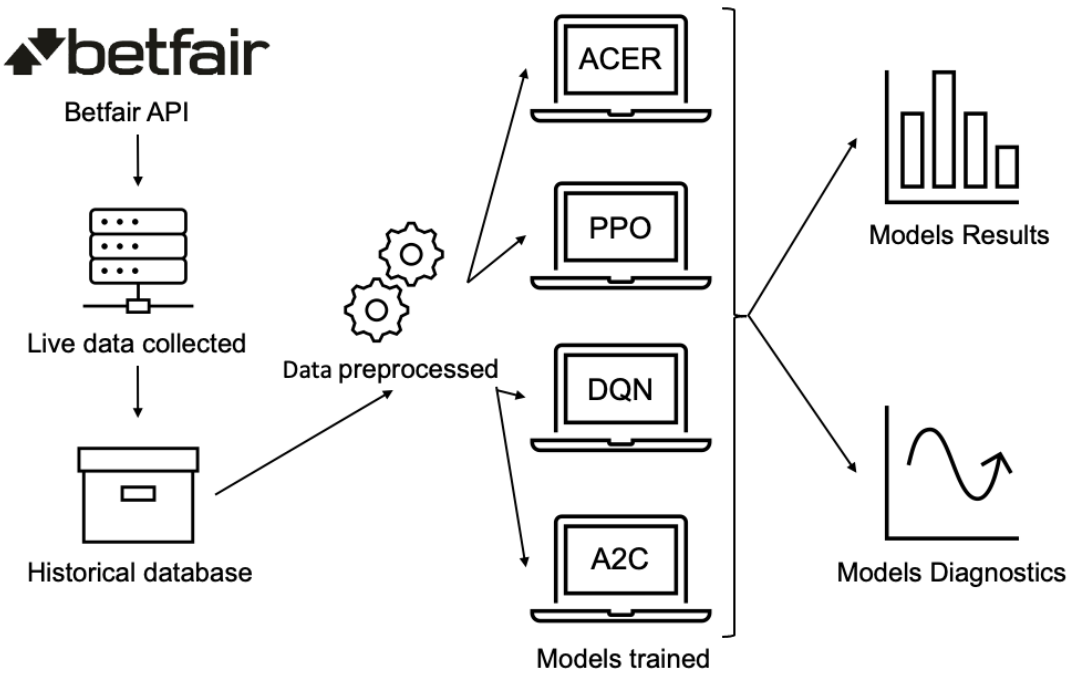


Figure 3.1: The framework of data flow within this trading system

opened. Each action will stop placing trades at the end of each episode (step 240) and will automatically be closed out and rewards are given based on the agent's last trade.

3.2.2 State space

The state space S for the model represents the environment it will interact with. The state space is considered the set of the feature inputs of the model. All of these observations are normalised as inspired by [4]. These feature inputs are drawn from [24] and other financial reinforcement learning papers:

1. *Step number:* The model will know the number of steps until the completion of training, the maximum number of steps is 240 as the episode length cannot exceed this.
2. *Time until jump:* The model will use the difference between the supposed event starting time and the current time. The figure will be calculated as the difference between the jump time and the current time. At jump time, when $t = 0$ if there are any outstanding trades the agent will be overridden, and trades will be closed and given a heavy feedback penalty for not closing before the jump.
3. *2min average probability:* This feature generates the average of the last 4 minutes of both back and lay probabilities from the current step. The method is as follows,

$$P_{\text{average}} = \frac{1}{N}\sum^{N}(\frac{P_{\text{back}} + P_{\text{Lay}}}{2}) \tag{3.2.1}$$

Where N is the number of time steps within 2 minutes. This input allows an agent to understand the current probabilities relative to where it has averaged historically.

4. *2min average price change:* Another feature will look at the difference between the means of two time periods of 2 minutes. Prices of racing events tend to either firm (price falls) or drift (price rises) close to the jump as all information about the racer is now known. The input allows the agent to understand the direction of the price within the last two minutes and can build a theory around the proposed future price based on the change.
5. *Exponential moving average:* To mitigate data noise and uncertainty, hand-craft financial features, e.g. EMA, MACD [63]. The EMA [42] is a method of synthesising the moving average but applying weighting to the more recent time events. Calculated recursively as,

$$\text{EMA}_t = \alpha P_t + (1-\alpha)\text{EMA}_{t-1} \tag{3.2.2}$$

and α is a hyper-parameter that can be tuned to accentuate recent time stamps as $\alpha \to 1$. This allows an agent to see the price historically with a focus on the recent direction of the market.

6. *Moving average convergence divergence:* MACD is used in DRL in market trading [23], it can be used in conjunction with the prices of an outcome. The MACD within this context is calculated as the difference between the last 2-minute EMA and the 5-minute EMA. The MACD is calculated for every interval and used for the model to identify movement trends.
7. *2min average probability with weight of money*
   The average price with the Weight of money (WOM) had returned with success in market-style gambling paper [24]. The statistic is altered to explore the relationship between price and WOM. The inclusion of volume penalises available volume that is far from the current price. The WOM is calculated as a ratio between the weighted volume available to back and lay.

$$W_{\text{back}} = \sum_{i=1}^{N} \frac{V_{P_{\text{back}(i)}}}{i} \text{ and } W_{\text{lay}} = \sum_{i=1}^{N} \frac{V_{P_{\text{lay}(i)}}}{i}$$

$$\text{WOM} = \frac{W_{\text{back}}}{W_{\text{back}} + W_{\text{lay}}} \tag{3.2.3}$$

where N represents the number of intervals. WOM is calculated at every interval with a look back of 2 minutes.

3.2.3 Reward space

The reward space allows for the definition of the model's payment structure, inspired by previous works [31], [91] the reward within this thesis is directly related to the profit. Given an action space of long at time t is defined as l_t and previous long at l_{t-1}. The stake S will be fractioned by action choice $\delta \in \{0.25, 0.5, 0.75, 1\}$ and a commission rate ρ set, thus the reward structure of time t is set as such,

$$r_t = \begin{cases} S\delta(1-\rho)(\frac{l_t}{l_{t-1}}) & \text{if } \frac{l_t}{l_{t-1}} > 1 \\ S\delta(\frac{l_t}{l_{t-1}}) & \text{if } \frac{l_t}{l_{t-1}} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.2.4}$$

where $\rho = 0.05$ if $\mathrm{R}_t > 0$ else $\rho = 1$. The rewards are discounted at the rate γ. Where,

$$\mathrm{R} = \sum_{i=1}^{N} \gamma^i r_{t+i}, \tag{3.2.5}$$

The discounted future rewards allow for an agent to plan. Where $\gamma = 0$ is a greedy policy and $\gamma = 1$ is a fully planned model. Within this thesis, γ will be set differently for each agent dependent on the hyper-parameter tuning of each model.

3.3 Implementation

3.3.1 Data and markets

The data used is 2000 Australian thoroughbred and greyhound events between November and December 2022. Within each event, only the win market will be explored with an average of 8 propositions. The data is collected by using the *Betfair* API service, calling for the prices. For each proposition, the data will be called at 5-second intervals 20 minutes from the beginning of the event. The pre-processing will separate each proposition whilst applying the feature inputs listed in section 3.2.2. The data is vectorised and used to set up the betting environment.

3.3.2 Computational features

The models are trained locally on a MacBook Pro, 2.3GHz Quad-core i5. The models are created using a stable-baselines3 [71] implementation in Python with a custom environment. The custom class is named BettorEnv and outlines the states, actions, data and rewards. Thus, completing the outline of the necessary resources required to replicate these models.

Hyper-parameter	Range	Reference
Actor/Critic network	(64,64,1)	[65]
Gamma	(0.8, 0.999)	[4]
Learning rate	(1e-10, 0.01)	[46]
Entropy coefficient	(1e-8, 0.1)	[7]
GAE lambda	(0.9, 1)	[92], [76]
Clip range	(0.1, 0.3)	[7]
Value function coefficient	(0.5, 1)	[7]
Max gradient normalization	(0.01, 1)	[25]
Rprop-epsilon	(0, 0.01)	[26]
Correction term	($1\exp^{-8}$, $1\exp^{-6}$)	[26]
Exploration fraction	(0, 1)	[8]
Tau	(0.001, 0.1)	[41]

Table 3.1: Range of DRL hyper-parameters and reasoning

3.3.3 Hyper-parameter tuning

As an objective measure of testing the performance of each learner in the trading environment, the hyperparameters of each agent will be chosen on a subset of the racing data. It is known that DRL is extremely sensitive to hyper-parameters [53]. Due to computational constraints and the wide range of continuous hyper-parameters, Bayesian optimisation is used [44]. The framework for the Bayesian optimisation is provided by an API framework in Python named Optuna [3]. The framework applies relational sampling between hyperparameters and an efficient pruning algorithm to arrive at optimal values. Each agent will be optimised with unique sets of hyperparameters, looking across each range of values. These parameters include learning rate, gamma and entropy coefficient and will be represented within the results section.

Trading with a neural network involved using a grid search of hyper-parameters measured on the first 30% of the training set after labelling is complete. Each combination of parameters is tested on the smaller subset and the model with the smallest entropy loss is chosen. These parameters will be selected as the final model and used to test against the DRL agents. Due to the smaller discrete combinatorial space of neural networks, a grid search can be used.

3.3.4 Machine learning trading

To understand the context of the profitability of DRL we compare the agent to the machine learning algorithm. This will give further insight into another algorithm. Choosing a neural network as it is used in [24] to trade on price directly. We emulate the trading process as close as possible to the DRL for the neural network. The data is divided in half for train and testing. We apply the grid search mentioned in 3.3.3 to the neural network. Table 3.3 depicts the exact values of the neural network that was the most profitable over the train set. We apply labels using the tested triple-barrier method in [21]. The agent's profitability will trade with a reward similar to equation (27) however the δ parameter is continuous between 0 and 1 reflecting the model's confidence of direction in the trade. In comparing the

	Model Name					
	A2C	A2C (LSTM)	ACER	ACER (LSTM)	DQN	PPO
Actor/Critic network	(64, 64, 2)	(64, 64, 2)	(64, 64, 2)	(64, 64, 2)	(64, 64, 2)	(64, 64, 2)
Gamma	0.907630814	0.926080116	0.908428004	0.994566878	0.989404125	0.916368073
Learning rate	0.001453703	3.26E-05	6.21E-05	0.000934552	0.021085628	0.000216744
Entropy coefficient	4.80E-05	4.85E-07	3.49E-07	0.092481572	-	4.62E-07
GAE lambda	-	-	-	-	-	0.906086827
Clip range	-	-	-	-	-	0.119458197
Max gradient norm	0.745136687	0.021688087	0.317711477	0.536255486	0.875090227	0.328598758
Rprop-epsilon	-	2.46E-05	0.606509166	0.000672379473	-	-
Correction term	-	-	-	-	-	-
Exploration fraction	-	-	-	-	-	-
Value function coefficient	0.020133023	0.527286098	-	2.55E-08		7.03E-07
Tau	-	-	-	-	0.115659586	-

Table 3.2: Agent rewards over multiple runs

Hyper-parameter	Value
Solver	Adam [46]
Hidden layer size	(5,5,3)
Activation	ReLu [2]
Learning rate	adaptive
Momentum	0.9

Table 3.3: Neural network selected hyperparameters

DRL agent and the neural network we apply the Sharpe and win-loss ratio. The implementation is done through Scikit-learn [66] in Python.

3.3.5 Assumptions

Considering that the agent will be trading on static data that has been collected and cleaned from *Betfair* there will be assumptions imposed on this experiment. Inspired from [16] these assumptions can be rationalised and then used as a framework for agents learning on pre-downloaded data.

1. *Sufficient Liquidity:* A proposition is considered liquid if it can be rapidly converted to cash, with little or no value loss [38]. When trading with the racing markets with 20 minutes to go we are dealing with the highest point of liquidity in the market. Further, the small stake size should always be covered by volume in the market.
2. *Zero Slippage:* Each trade can be executed with zero slippage, exactly at the back or lay odds. Slippage refers to the difference between a trade's expected execution price and the price at which the trade is actually executed [37]. We send orders during the live betting times taking odds at the best back or best lay odds, reducing slippage.
3. *Zero Market Impact:* The agent making trades does not influence the odds. The DRL agent bets small enough not to direct the market odds.
4. *Zero Latency Impact:* The delay to send orders has no influence on the execution prices. In a live context the agent has access directly to the market and considering the odds at time t is being parsed every 5 seconds there will be sufficient time to place money in a live context.

3.3.6 Evaluation metrics

The evaluation metrics chosen within this thesis are designed to encapsulate the three questions this thesis posits. These metrics are selected from related DRL papers that take on similar problems. Thus,

1. Convergence of models: The convergence of models will inspect specifically the policy gradient models. There will be a multitude of factors including value-function loss, policy gradient loss, entropy loss and advantage [10]. These figures will be mapped over the 5 million time steps and should aptly converge towards certain values indicating the model has learned a local/global optimum.
2. Profitability of models: The profitability of the model is directly tied to the reward structure of the agents. As aforementioned these are standard across all agents, key figures that will be tracked are the profit per episode, wins,

losses, Sharpe ratio and win-loss ratio. These metrics are brought forth in multiple financial DRL papers in assessing profitability. Inspired by [14], further evaluation metrics including value at risk and conditional value at risk (CVAR) [73]. These metrics will be indicative of the model learning over time.

3. Comparison against machine learning: The metrics will be reused from the profitability section focusing on Sharpe [88] and win/loss ratio as the statistics as the rewards will be compared against the two models.

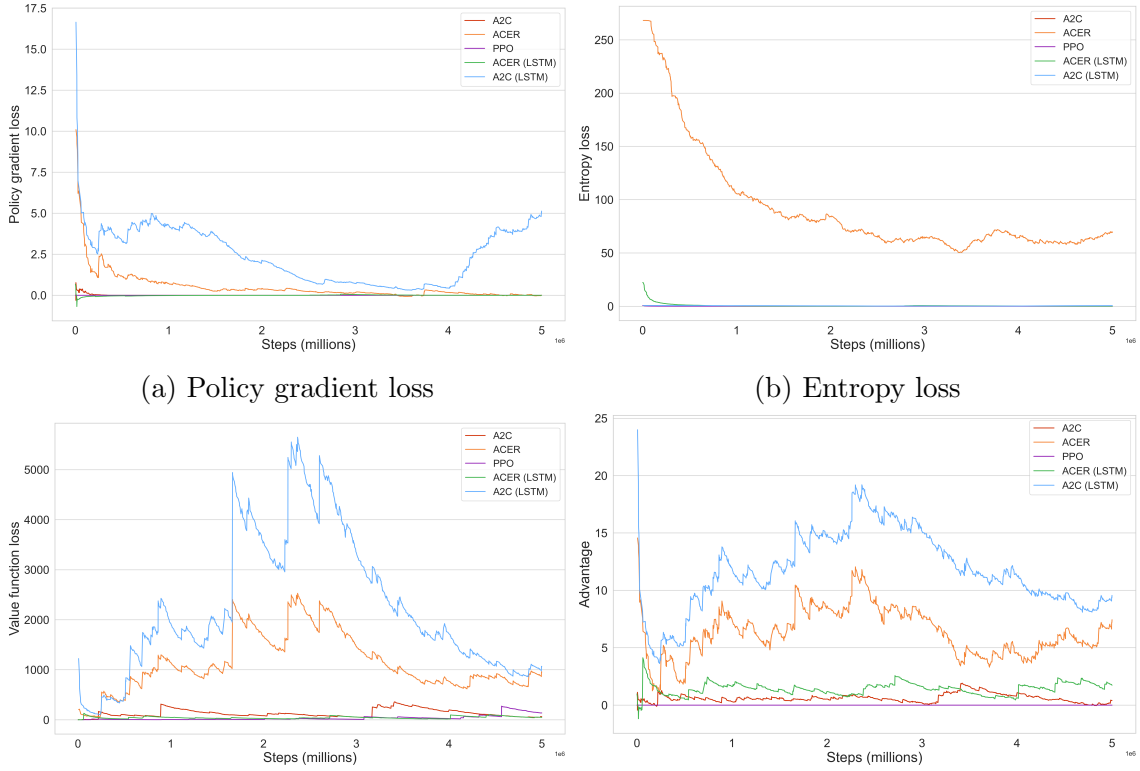
Chapter 4

Results

4.1 Model verification

We apply the entire dataset with both codes, greyhounds and thoroughbreds to the model. They are compiled data frames with each episode lasting a maximum of 240 steps. The outputs of each episode are logged through a tensor board and key evaluation metrics are visualised. The outputs are overlaid giving key comparisons amongst agents for learning. Figure 4.1 gives insight into the learning trends.

Figure 4.1(a) represents the policy gradient loss, correlating to how much the policy (process for deciding actions) is changing. Research states the loss should converge towards 0 indicating a policy has learnt optimal decisions for each action. For most agents the policy converges towards 0, the ACER agent converges last whilst A2C (LSTM) does not converge at all showing a clear upward trend between 4 and 5 million steps. Evidently, PPO converges the quickest within a few thousand



(a) Policy gradient loss

(b) Entropy loss

(c) Value function loss

(d) Advantage

Figure 4.1: Model diagnostics over a single run

steps. The entropy loss is a metric that represents the randomness of an agent's actions. If the policy aligns with the reward the entropy loss should decrease due to confidence in the policy increasing. As presented in Figure 4.1(b) there is a clear trend of entropy loss towards 0 by the half a million step for all except the ACER agent as the agent tends between 50 and 100 throughout the steps. Within Figure 4.1(c) the value function loss represents how well a model is able to predict the value of each state. A2C (LSTM) and ACER have a significant difference in value function loss with a trend towards 0. The other models tend to perform similarly converging towards 0, however, ACER (LSTM) outperforms in predictive performance consistently closer towards 0. The advantage is another critical factor in the analysis of a deep reinforcement learners diagnostic. Within Figure 4(d) the advantage is plotted across time. The advantage statistic represents how much is a certain action a 'good' or 'bad' decision given a certain state. Derived as

$$A(s, a) = \mathbb{E}[r(s, a) - r(s)], \tag{4.1.1}$$

advantage closer to 0 indicates a policy with more profitable decisions, we find that all actors converge towards 0 with the exception of the LSTM policies, as both do not have trends towards to - throughout the time steps. The other MLP agents converge around 0.

4.2 Model results

Using multiple replays of 500,000 time steps, the model results are calculated and the average and standard deviation are tabulated. We determine the output of agents and their profitability when trading on thoroughbreds and greyhounds. As rewards dictate the trader's profit, the most profitable trader is the trader with the highest net worth yield. As represented in Table 4.1 the most profitable trader is the ACER with an average net worth of 48.1 rewards per episode across 30 runs. We find that DQN is the least profitable at 45.99, however, over the number of wins statistic, the ACER (LSTM) is the most profitable with 114.18 wins per episode (240 steps), it also has the highest number of losses (55.49) explaining why it is not the most profitable per episode. The final profitability statistic is profit per trade. The ACER (LSTM) agent on average receives the most profit per trade with \$4.92 per trade whilst the DQN agent on average receives the lowest by a considerable margin at \$4.28 per trade. In finance, the Sharpe ratio determines the risk-adjusted rate of return giving an indication of the strength of the entire strategy and summarising the upside and downside. We find that the DQN has the highest Sharpe ratio of 0.4 with the lowest being ACER (LSTM) at 0.38. The win-loss ratio gives a great indication of the policy maximising upside and minimising downside. The DQN again outperforms the other models significantly with a win-loss ratio of 2.21, whilst the average of the other agents are 2.08.

	Model Name					
	A2C	A2C-LSTM	ACER	ACER-LSTM	DQN	PPO
$\mu_{\text{Net worth}}$	46.07 (10.93)	46.49 (10.8)	**48.1 (11.00)**	46.60 (11.04)	45.99 (10.72)	46.01 (10.93)
$\mu_{\text{Number of wins}}$	111.59 (6.24)	113.53 (6.27)	111.52 (7.03)	114.18 (6.32)	100.96 (7.41)	110.92 (6.46)
$\mu_{\text{Number of losses}}$	53.9 (5.05)	54.75 (5.07)	52.38 (5.53)	55.49 (5.2)	**45.63 (5.60)**	53.52 (5.16)
$\mu_{\text{Profit per trade}}$	4.85 (0.08)	4.89 (0.06)	4.6 (0.16)	**4.92 (0.08)**	4.28 (0.22)	4.83 (0.11)
$\mu_{\text{Sharpe ratio}}$	0.39 (0.05)	0.39 (0.04)	0.39 (0.05)	0.38 (0.04)	**0.4 (0.03)**	0.38 (0.06)
$\mu_{\text{Win-loss ratio}}$	2.07 (0.4)	2.07 (0.39)	2.13 (0.4)	2.06 (0.45)	**2.21 (0.36)**	2.07 (0.37)

Table 4.1: Agent rewards over multiple runs

To provide further context to these statistics, further model statistics are examined over time. A model's episode reward represents the cumulative rewards over a single episode, as shown in Figure 4.2. All models perform similarly at certain stages during the episode. The episode reward range for all models lies between 30 and 40. Throughout PPO, ACER (LSTM) and A2C perform better between 1 and 2 million time steps. However, all agents except A2C (LSTM) tend to converge towards the same reward.

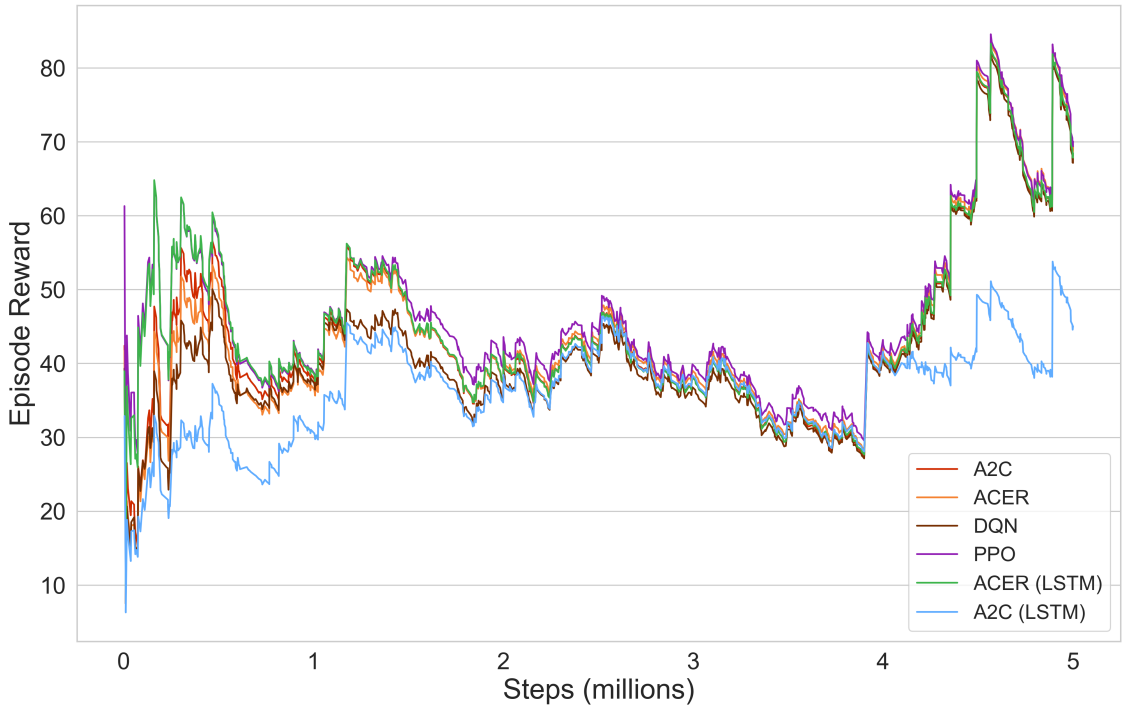
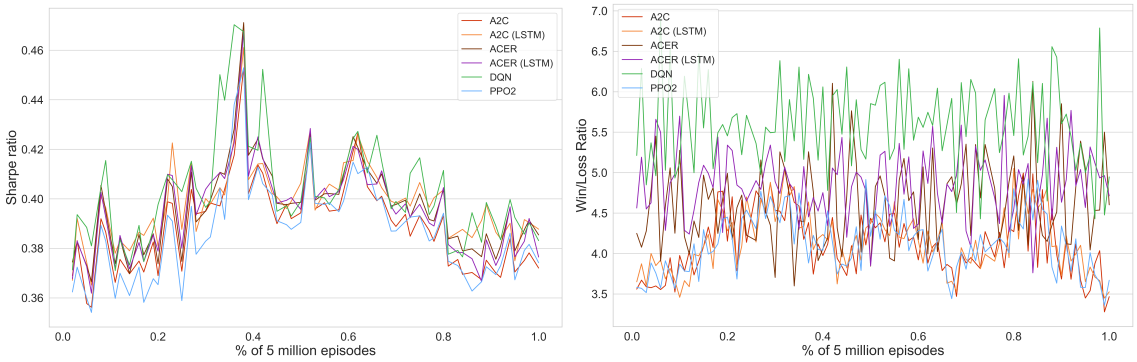


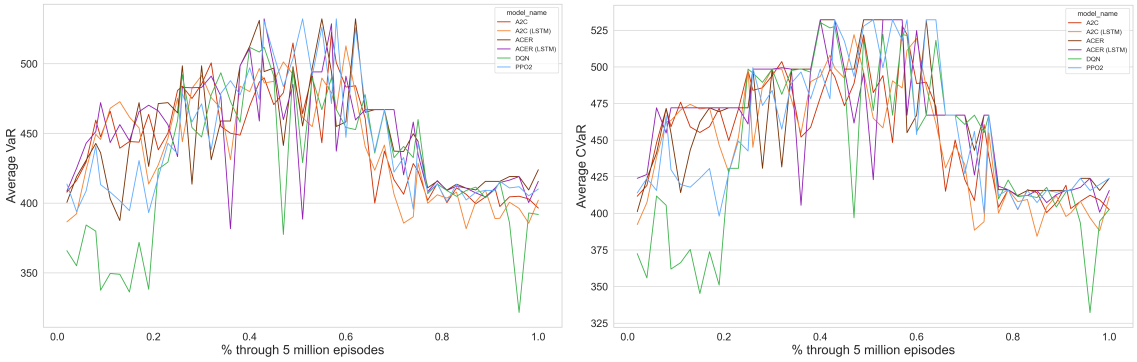
Figure 4.2: Episode rewards over a single run

As shown in Figure 4.3(a), the Sharpe ratio is measured over time. The overall Sharpe ratio does not improve over time and there is no clear trend within the graph. There is a fluctuation between 0.38 and 0.46 starting and ending within the same position. DQN consistently outperforms the other agents when comparing Sharpe ratios across models, whilst PPO tends to lie at the bottom of the model's Sharpe ratio. The Win v Loss ratio is plotted in Figure 4.3(b) over time, and its trend is linear, with the agents fluctuating around the starting value. Once more, the DQN clearly outperforms the other models with a win-loss ratio bounded between **1.5** and 2.5. All other models fall between 3.5 and 5.5.
Further, the value and conditional value of risk over time using net worth binned over time. Shown in Figure 4.4(a) is the value at risk over time. There is not a clear trend over time however, between the 50% and 100% of the episodes the VaR decreases consistently. The DQN outperforms the other models as it trends lower than the other agents, whilst there is no clear agent that performs the most poorly. There is also evidence of these trends in Figure 4.4(b), where the conditional value at risk is plotted over time.



(a) Sharpe ratio over runs (b) Win v Loss over time

Figure 4.3: Sharpe and Win-loss ratio throughout agent learning

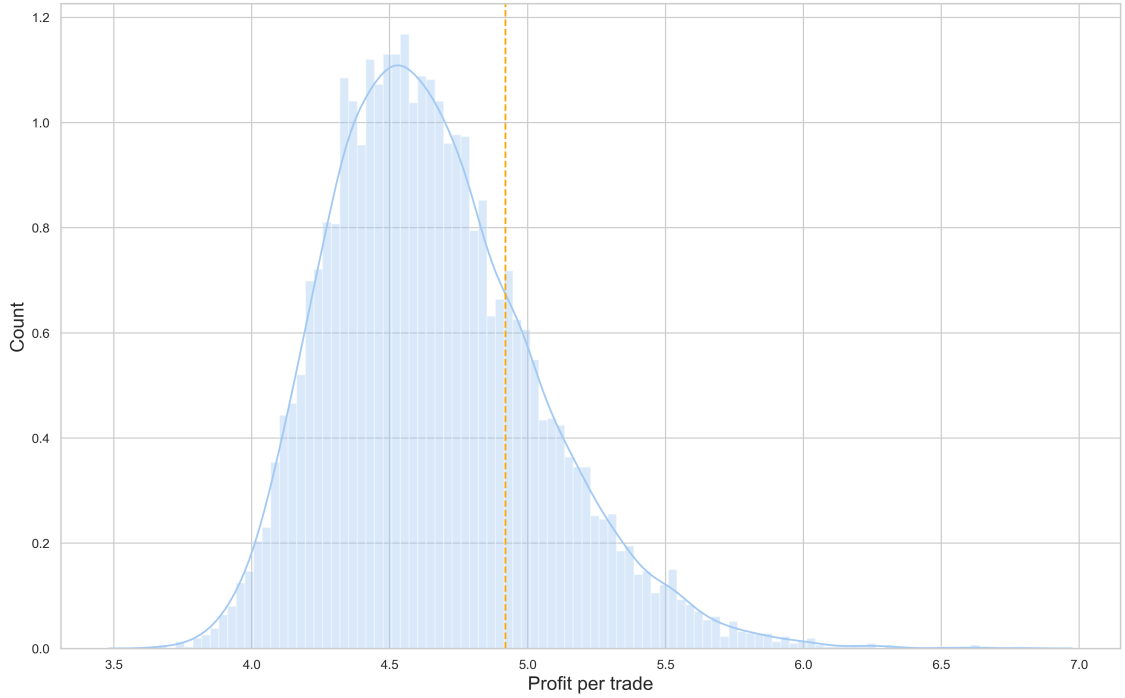


(a) Value at risk over time (b) Conditional value at risk over time

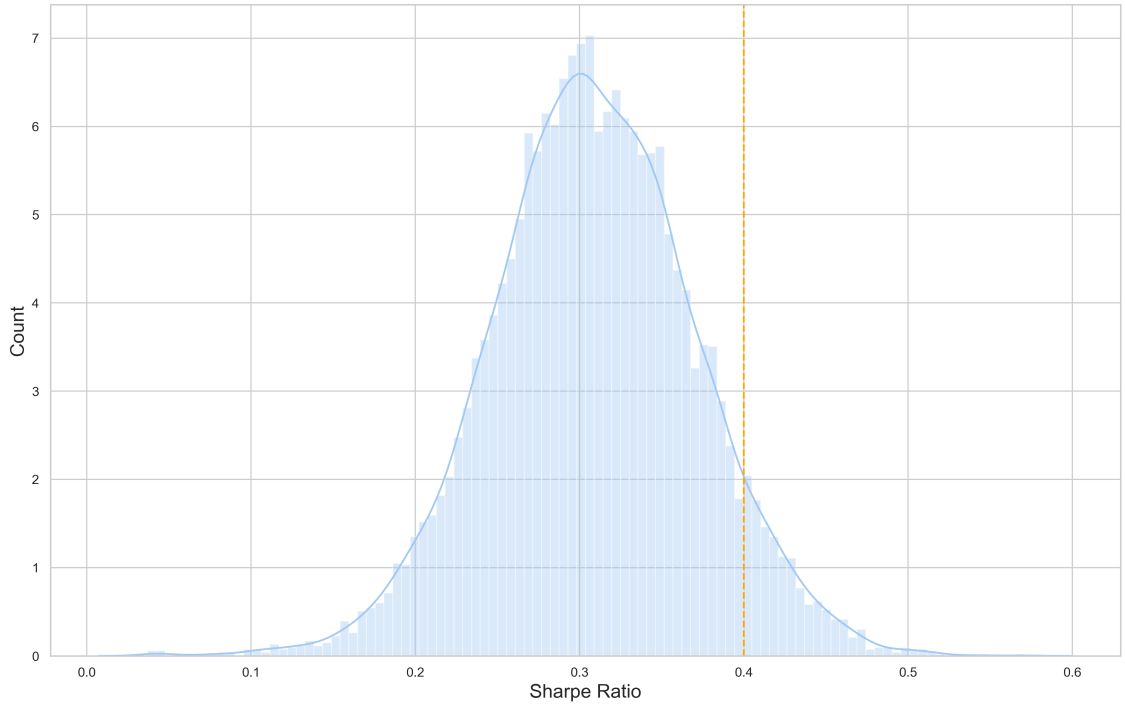
Figure 4.4: Conditional and unconditional value at risk over time where $\alpha = 0.05$

4.3 Comparison against machine learning algorithm

The neural network is run 1000 times using the framework suggested in section 3.34 to calculate the profit per trade and Sharpe ratio for each run. Figure 4.5(a) shows the neural network profit per trade distribution. Finding that the mean score is 4.68 per trade and a standard deviation of 0.378, however, all models except for DQN (4.28) and ACER (4.6) outperform in this metric through mean and a smaller standard deviation. The orange line indicates the best-performing trader within this metric ACER with an LSTM (4.92) which is less than one standard deviation from the neural network's mean. Further, Figure 4.5(b) displays the distribution of Sharpe ratios of the neural network. The mean shape score is 0.308 and the standard deviation is 0.06. The mean DQN score (the best-performing model) is 0.4 with a standard deviation of 0.03 outperforming the neural network model. The other models outperform the baseline learner, and the agents outperform supervised learners.



(a) Neural network profit per trade distribution with orange line indicating mean ACER (LSTM) Sharpe



(b) Neural network Sharpe ratio histogram with an orange line indicating DQN mean Sharpe ratio

Figure 4.5: Comparing the best RL learners with neural network

Chapter 5

Discussion

5.1 Result summary

We have found that most models tended to converge over the environment given in the context of sports gambling. A data pipeline involving the transfer of data from Betfair to a local machine and palatable for a DRL agent was adopted and then tested with these agents. We found that data on thoroughbred and greyhound odds could be used to form policies to trade in these markets. In these markets, most models tend to converge on policies to trade. With many policy-based algorithms, the randomness of decision-making approaches zero, as illustrated in Figure 4.1(a-d).

Using the policy proximal optimisation agent results in the best convergence performance. Demonstrated in Figure 4.1(a-d) as the most efficient in loss convergence and advantage. In line with the current research, PPO tends to converge faster than A2C and its derivatives [77]. Because PPO uses smaller policy updates, it is more likely to converge to an optimal solution directly. In addition, PPO avoids a "too large " step in its policy update, which will cause a policy to fall "off a cliff" and take longer to converge with an optimal policy [80]. Compared to A2C (LSTM) and ACER, we find that the other algorithms converge on policies at different rates over our set time horizon. As shown in Figure 4.1(b-d) ACER does not reduce the loss functions towards 0. ACER can be challenging to converge since it is an application of an off-policy algorithm to an on-policy agent (A2C) [75]. The lack of convergence of the A2C (LSTM) agent shown in Figure 4.1(a,c-d) highlights the agent's inability to converge on an optimal decision. Current literature suggests that for both models, the lack of convergence could suggest the models do not have enough episodes to train on, or the policy updates are too large at each episode [70] and thus not converging. To evaluate the long-term convergence rates of these agents, a larger dataset is required. The proof that some DRL agents can learn to converge trading policies on gambling markets implies they can interact allowing future work to be applied to teaching agents to trade. Through this diagnostic evaluation, it is clear that many of the models can converge policies and value functions in learning about racing markets.

In addition, this thesis examines whether an agent can be profitable in an uncertain market. Our findings indicate that agents can be, and there are two distinctions between the most profitable and the best traders. As shown in Table 4.1 ACER is the most profitable agent, and this is due to a multitude of reasons. Trading is a continuous action space problem, where the agent must decide how much to buy or

sell. Trading environments can be volatile, and ACER's ability to handle large state spaces makes it more robust to changes in the market. In trading, the agent must balance exploration with exploitation, exploring new strategies while exploiting those that have proven to be successful using ACER's actor-critic architecture, these two goals can be effectively balanced [75]. The implications of ACER's profitability may be used as an algorithm that is mostly used or integrated into a live trading framework.

In terms of overall trading characteristics, the DQN outperforms the other algorithms. DQN exhibits the highest Sharpe ratio, implying that it has the greatest risk-adjusted reward. The Sharpe ratio in sports markets is typically lower than in financial markets, which is the reason for the value of 0.4 [18]. The DQN also produces the highest win-to-loss ratio evident in Table 4.1 with a value of 2.21. These results are confirmed by Figure 4.3 and Figure 4.4 as the DQN outperforms throughout time. It is possible that this is due to the nature of the agent, unlike policy gradient algorithms, DQNs do not directly affect the policy space. Consequently, other policy algorithms converge quicker than the DQN agent, but tend to reach a local optimal rather than a global optimal [6]. The policy gradients' biggest drawback is the high variance in estimating the gradient $\mathbb{E}[R_t]$, by estimating the gradient generated by a series of data points accumulated through a single market [6]. In our particular case, our agents interact with episodes that are particularly brief, which exacerbates this variance. Hence, a noisy estimation leads to poor stability in learning, leading to a more probable convergence around a local maximum [6]. Thus, realising the DQN as the preferred trader of the set of algorithms.

Although the policies show convergence, it is evident from Figure 4.3 and Figure 4.4 that there is no clear trend of profit convergence. It is evident that there is no trend within the statistics of Sharpe, VaR, CVaR, and win-loss ratio that converges to a single point. Despite learning policies to determine when to trade, the agents are subject to randomness and noise in the state space due to the stochastic nature of trading. The future suggestions to improve this would be to investigate over longer time spans or to produce more informative state spaces that would reduce the noise presented to the trader, resulting in more profitable policies.

For the purpose of being objective, we compare our findings with that of a related previous paper on market-style trading [24]. In our analysis, the agents outperform a neural network, which is currently the benchmark paper. This is evident in Figure 4.5(b) as the Sharpe of the DQN is higher than a machine learning model. This is a result of the deeper structure of the agents constantly interacting with the trading environment and constantly learning throughout each episode. As each episode will inform the next, the learner can be considered more practical for sequential decision problems. We can therefore conclude that reinforcement learning has improved in solving market-style gambling problems.

5.2 Limitations and future work

Despite the fact that deep reinforcement learning agents are capable of acting as traders within market-style gambling systems, some aspects of the framework

and features need to be improved in order to fully integrate an agent into a trading environment. It is evident that while this thesis represents a significant step forward in the market-style gambling space there is still a need to improve certain aspects in order to ensure that the best traders are found on the market. It is important to note that although this study answers key questions regarding the topic of DRL in trading, improvements can be made by improving the framework. The agent can be improved to trade with back, hold and lay, allowing an agent to take both sides of a proposition. It would also be beneficial to address the assumptions imposed by this author. Including, trading using volume amounts on the live market, would remove the first three assumptions as an agent is now exposed to complications with trading on market-style gambling. The evaluation metrics could be developed further using the entire framework provided by [73] using IQR of episode reward and dispersion through a time of the policies. Finally, development in the field of transfer learning [90] could answer if the policies generated on racing markets could apply to other codes of racing and sports.

Chapter 6

Conclusion

In this study, a novel deep reinforcement learning framework that utilises gambling market-specific data on thoroughbreds and greyhounds were developed to identify if agents could converge on a profitable policy and whether these policies outperform current models. We conclude that out of the different agents tested, most agents converge on a profitable policy on thoroughbreds and greyhound data. In addition to this, we also conclude that DQN outperformed the other agents with respect to being the most viable trader. This is most likely due to its tendency to converge to a global optimum. These agents have been selected on the basis of their widespread use within deep reinforcement learning literature. However, we suggest that these agents are not yet suitable to trade in live environments due to other considerations like trading time period, computational speed and adversarial considerations. Our deep reinforcement learning framework can be easily adapted for future work in the market-style gambling space.

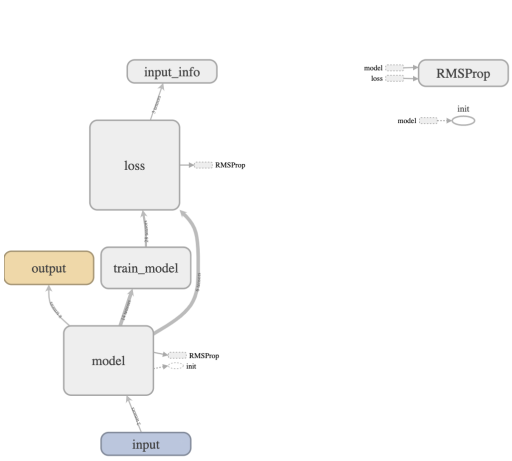
Chapter 7

Appendix

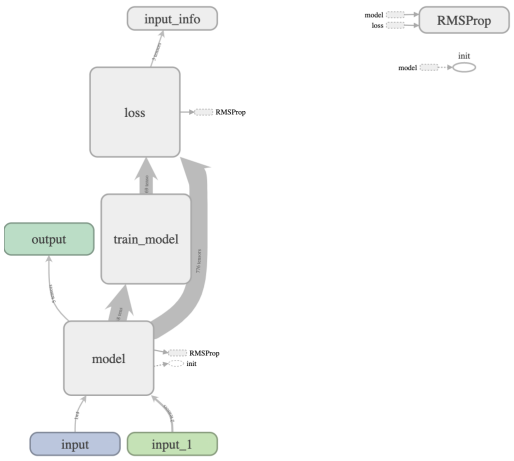
7.1 Code availability

The code in this study is partially available on Github at: `https://github.com/georgemaksour/final_honours_datapull.git`

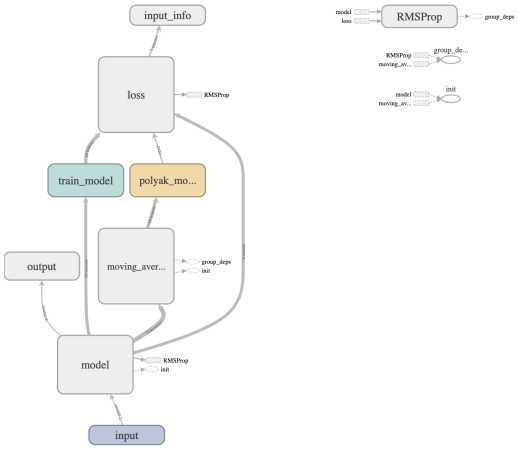
7.2 Further figures



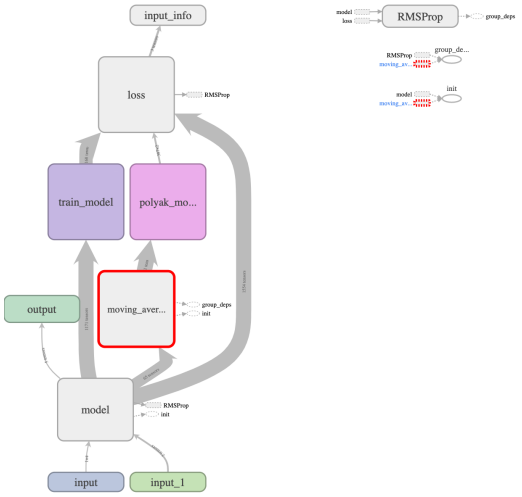
(a) A2C



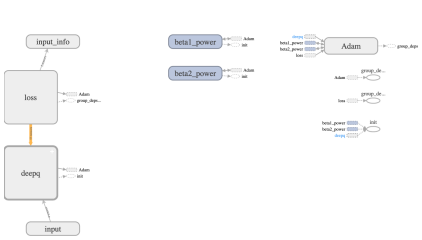
(b) A2C (LSTM)



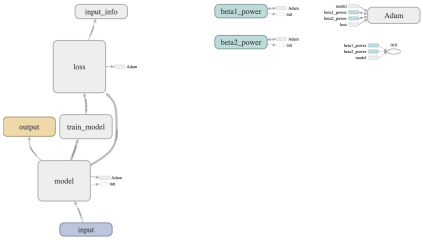
(c) ACER



(d) ACER (LSTM)



(e) DQN



(f) PPO

Figure 7.1: Diagrams of trained models

References

[1] I. Abinzano, L. Muga, and R. Santamaria, *Game, set and match: the favourite-long shot bias in tennis betting exchanges*, Applied Economics Letters, 23 (2016), pp. 605–608.

[2] A. F. Agarap, *Deep learning using rectified linear units (relu)*, arXiv preprint arXiv:1803.08375, (2018).

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, *Optuna: A next-generation hyperparameter optimization framework*, in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2623–2631.

[4] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, et al., *What matters for on-policy deep actor-critic methods? a large-scale study*, in International conference on learning representations, 2021.

[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, *A brief survey of deep reinforcement learning*, arXiv preprint arXiv:1708.05866, (2017).

[6] ——, *Deep reinforcement learning: A brief survey*, IEEE Signal Processing Magazine, 34 (2017), pp. 26–38.

[7] AurelianTactics, *Ppo hyperparameters and ranges*, Jul 2018.

[8] K. Azizzadenesheli, E. Brunskill, and A. Anandkumar, *Efficient exploration through bayesian deep q-networks*, in 2018 Information Theory and Applications Workshop (ITA), IEEE, 2018, pp. 1–9.

[9] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, *An actor-critic algorithm for sequence prediction*, arXiv preprint arXiv:1607.07086, (2016).

[10] A. W. Beggs, *On the convergence of reinforcement learning*, Journal of economic theory, 122 (2005), pp. 1–36.

[11] W. Benter, *Computer based horse race handicapping and wagering systems: a report*, in Efficiency of racetrack betting markets, World Scientific, 2008, pp. 183–198.

[12] R. P. Bunker and F. Thabtah, *A machine learning framework for sport result prediction*, Applied computing and informatics, 15 (2019), pp. 27–33.

[13] M. Cain, D. Law, and D. Peel, *The favourite-longshot bias and market efficiency in uk football betting*, Scottish Journal of Political Economy, 47 (2000), pp. 25–36.

[14] S. C. Chan, S. Fishman, J. Canny, A. Korattikara, and S. Guadarrama, *Measuring the reliability of reinforcement learning algorithms*, arXiv preprint arXiv:1912.05663, (2019).

[15] C. L. Claussen and L. K. Miller, *The gambling industry and sports gambling: A stake in the game?*, Journal of Sport Management, 15 (2001), pp. 350–363.
[16] L. Conegundes and A. C. M. Pereira, *Beating the stock market with a deep reinforcement learning day trading system*, in 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.
[17] K. Croxson and J. James Reade, *Information and efficiency: Goal arrival in soccer betting*, The Economic Journal, 124 (2014), pp. 62–91.
[18] D. S. da Silva, *Proper application of sharpe ratio in sports investing*, 2016.
[19] I. Daunhawer, *Evolutionary algorithms for the discovery of trading rules in high-frequency betting markets*, PhD thesis, University of Konstanz, 2018.
[20] M. Davies, L. Pitt, D. Shapiro, and R. Watson, *Betfair. com:: Five technology forces revolutionize worldwide wagering*, European Management Journal, 23 (2005), pp. 533–541.
[21] M. L. de Prado, *Advances in Financial Machine Learning*, John Wiley & Sons, New York, NY, 2018.
[22] L. Deng and J. Platt, *Ensemble deep learning for speech recognition*, in Proc. of INTERSPEECH, 2014.
[23] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, *Deep direct reinforcement learning for financial signal representation and trading*, IEEE transactions on neural networks and learning systems, 28 (2016), pp. 653–664.
[24] I. Dzalbs and T. Kalganova, *Forecasting price movements in betting exchanges using cartesian genetic programming and ann*, Big data research, 14 (2018), pp. 112–120.
[25] T. Eimer, C. Benjamins, and M. Lindauer, *Hyperparameters in contextual rl are highly situational*, arXiv preprint arXiv:2212.10876, (2022).
[26] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al., *Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures*, in International conference on machine learning, PMLR, 2018, pp. 1407–1416.
[27] A. Fodor, M. DiFilippo, K. Krieger, and J. Davis, *Inefficient pricing from holdover bias in nfl point spread markets*, Applied Financial Economics, 23 (2013), pp. 1407–1418.
[28] E. Franck, E. Verbeek, and S. Nüesch, *Prediction accuracy of different market structures—bookmakers versus a betting exchange*, International Journal of Forecasting, 26 (2010), pp. 448–459.
[29] S. Fujimoto, H. Hoof, and D. Meger, *Addressing function approximation error in actor-critic methods*, in International conference on machine learning, PMLR, 2018, pp. 1587–1596.
[30] Y. Fujita and S.-i. Maeda, *Clipped action policy gradient*, in International Conference on Machine Learning, PMLR, 2018, pp. 1597–1606.
[31] Y. Gao, Z. Gao, Y. Hu, S. Song, Z. Jiang, and J. Su, *A framework of hierarchical deep q-network for portfolio management*, in ICAART, 2021, pp. 132–140.
[32] Z. Gao, Y. Gao, Y. Hu, Z. Jiang, and J. Su, *Application of deep q-network in portfolio management*, in 2020 5th IEEE International Conference on Big Data Analytics (ICBDA), IEEE, 2020, pp. 268–275.

[33] S. Goto and T. Yamada, *What drives biased odds in sports betting markets: Bettors' irrationality and the role of bookmakers*, International Review of Economics & Finance, (2023).
[34] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, *A survey of actor-critic reinforcement learning: Standard and natural policy gradients*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42 (2012), pp. 1291–1307.
[35] F. G. Haahr and S. A. Andersen, *Market efficiency: An analysis of the internet betting exchange market*, Copenhagen Business School, (2011).
[36] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, in International conference on machine learning, PMLR, 2018, pp. 1861–1870.
[37] A. Hayes, *Slippage: What it means in finance, with examples*, Sep 2022.
[38] ——, *Understanding liquidity and how to measure it*, Mar 2023.
[39] E. W. Hill, J. Gu, S. S. Eivers, R. G. Fonseca, B. A. McGivney, P. Govindarajan, N. Orr, L. M. Katz, and D. MacHugh, *A sequence polymorphism in mstn predicts sprinting ability and racing stamina in thoroughbred horses*, PloS one, 5 (2010), p. e8645.
[40] T. Horvat and J. Job, *The use of machine learning in sport outcome prediction: A review*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10 (2020), p. e1380.
[41] C. Y. Huang, *Financial trading as a game: A deep reinforcement learning approach*, arXiv preprint arXiv:1807.02787, (2018).
[42] J. S. Hunter, *The exponentially weighted moving average*, Journal of quality technology, 18 (1986), pp. 203–210.
[43] T. Huotari, J. Savolainen, and M. Collan, *Deep reinforcement learning agent for s&p 500 stock selection*, Axioms, 9 (2020), p. 130.
[44] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, *Neural architecture search with bayesian optimisation and optimal transport*, Advances in neural information processing systems, 31 (2018).
[45] T. Kharrat, *A journey across football modelling with application to algorithmic trading*, The University of Manchester (United Kingdom), 2016.
[46] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).
[47] N. Kohl and P. Stone, *Policy gradient reinforcement learning for fast quadrupedal locomotion*, in IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, vol. 3, IEEE, 2004, pp. 2619–2624.
[48] V. Konda and J. Tsitsiklis, *Actor-critic algorithms*, Advances in neural information processing systems, 12 (1999).
[49] K. Krieger, J. L. Davis, and J. Strode, *Patience is a virtue: exploiting behavior bias in gambling markets*, Journal of Economics and Finance, 45 (2021), pp. 735–750.
[50] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, nature, 521 (2015), pp. 436–444.

[51] J. Li, R. Rao, and J. Shi, *Learning to trade with deep actor critic methods*, in 2018 11th International Symposium on Computational Intelligence and Design (ISCID), vol. 2, IEEE, 2018, pp. 66–71.

[52] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, *Continuous control with deep reinforcement learning*, arXiv preprint arXiv:1509.02971, (2015).

[53] C. Liu, J. Gao, Y. Bi, X. Shi, and D. Tian, *A multitasking-oriented robot arm motion planning scheme based on deep reinforcement learning and twin synchro-control*, Sensors, 20 (2020), p. 3515.

[54] G. Lucarelli and M. Borrotti, *A deep reinforcement learning approach for automated cryptocurrency trading*, in IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2019, pp. 247–258.

[55] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, *Applications of deep reinforcement learning in communications and networking: A survey*, IEEE Communications Surveys & Tutorials, 21 (2019), pp. 3133–3174.

[56] S. Mahadevan and J. Connell, *Automatic programming of behavior-based robots using reinforcement learning*, Artificial intelligence, 55 (1992), pp. 311–365.

[57] M. A. Metzger, *Biases in betting: An application of laboratory findings*, Psychological Reports, 56 (1985), pp. 883–888.

[58] D. Miljković, L. Gajić, A. Kovačević, and Z. Konjović, *The use of data mining for basketball matches outcomes prediction*, in IEEE 8th international symposium on intelligent systems and informatics, IEEE, 2010, pp. 309–312.

[59] A. Millea, *Deep reinforcement learning for trading—a critical survey*, Data, 6 (2021), p. 119.

[60] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, *Asynchronous methods for deep reinforcement learning*, in International conference on machine learning, PMLR, 2016, pp. 1928–1937.

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

[62] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., *Human-level control through deep reinforcement learning*, nature, 518 (2015), pp. 529–533.

[63] C. J. Neely, D. E. Rapach, J. Tu, and G. Zhou, *Forecasting the equity risk premium: the role of technical indicators*, Management science, 60 (2014), pp. 1772–1791.

[64] A. Neervannan, *Evaluating the effectiveness of deep reinforcement learning algorithms in a walking environment*, Baltic Journal of Modern Computing, 6 (2018), pp. 335–348.

[65] L. Parada, E. Candela, L. Marques, and P. Angeloudis, *Safe and efficient manoeuvring for emergency vehicles in autonomous traffic using multi-agent proximal policy optimisation*, arXiv preprint arXiv:2210.17381, (2022).
[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.
[67] A. Plaat, *Deep reinforcement learning*, Springer, Singapore, 2022.
[68] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, *Multilayer perceptron and neural networks*, WSEAS Transactions on Circuits and Systems, 8 (2009), pp. 579–588.
[69] M. L. Puterman, *Markov decision processes*, Handbooks in operations research and management science, 2 (1990), pp. 331–434.
[70] S. Qiu, Z. Yang, J. Ye, and Z. Wang, *On finite-time convergence of actor-critic algorithm*, IEEE Journal on Selected Areas in Information Theory, 2 (2021), pp. 652–664.
[71] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, *Stable baselines3*, 2019.
[72] D. S. Ratcliffe, K. Hofmann, and S. Devlin, *Win or learn fast proximal policy optimisation*, in 2019 IEEE Conference on Games (CoG), IEEE, 2019, pp. 1–4.
[73] R. T. Rockafellar and S. Uryasev, *Conditional value-at-risk for general loss distributions*, Journal of banking & finance, 26 (2002), pp. 1443–1471.
[74] A. Satariano and N. Kumar, *The massive hedge fund betting on ai*, Bloomberg Markets, September, 27 (2017), p. 2017.
[75] S. Schmitt, M. Hessel, and K. Simonyan, *Off-policy actor-critic with shared experience replay*, in International Conference on Machine Learning, PMLR, 2020, pp. 8545–8554.
[76] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, *High-dimensional continuous control using generalized advantage estimation*, arXiv preprint arXiv:1506.02438, (2015).
[77] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, arXiv preprint arXiv:1707.06347, (2017).
[78] R. P. Schumaker and J. W. Johnson, *An investigation of svm regression to predict longshot greyhound races*, Communications of the IIMA, 8 (2008), p. 7.
[79] J. P. Simmons, L. D. Nelson, J. Galak, and S. Frederick, *Intuitive biases in choice versus estimation: Implications for the wisdom of crowds*, Journal of Consumer Research, 38 (2011), pp. 1–15.
[80] T. Simonini, *Proximal policy optimization (ppo)*.
[81] M. Sipko and W. Knottenbelt, *Machine learning for the prediction of professional tennis matches*, MEng computing-final year project, Imperial College London, (2015).

[82] E. Snowberg and J. Wolfers, *Explaining the favorite–long shot bias: Is it risk-love or misperceptions?*, Journal of Political Economy, 118 (2010), pp. 723–746.
[83] K. Suri and S. Saurav, *Attentive hierarchical reinforcement learning for stock order executions*.
[84] K. Suri, X. Q. Shi, K. Plataniotis, and Y. Lawryshyn, *Trader: Practical deep hierarchical reinforcement learning for trade execution*, arXiv preprint arXiv:2104.00620, (2021).
[85] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.
[86] N. Tasfi and M. Capretz, *Noisy importance sampling actor-critic: an off-policy actor-critic with experience replay*, in 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.
[87] G. Tesauro et al., *Temporal difference learning and td-gammon*, Communications of the ACM, 38 (1995), pp. 58–68.
[88] T. Théate and D. Ernst, *An application of deep reinforcement learning to algorithmic trading*, Expert Systems with Applications, 173 (2021), p. 114632.
[89] A. S. Timmaraju, A. Palnitkar, and V. Khanna, *Game on! predicting english premier league match outcomes*, 2013.
[90] L. Torrey and J. Shavlik, *Transfer learning*, in Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, IGI global, 2010, pp. 242–264.
[91] A. Tsantekidis, N. Passalis, and A. Tefas, *Diversity-driven knowledge distillation for financial trading using deep reinforcement learning*, Neural Networks, 140 (2021), pp. 193–202.
[92] L. van Hezewijk, N. Dellaert, T. Van Woensel, and N. Gademann, *Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem*, International Journal of Production Research, (2022), pp. 1–24.
[93] R. Wang, H. Wei, B. An, Z. Feng, and J. Yao, *Commission fee is not enough: A hierarchical reinforced framework for portfolio management*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 626–633.
[94] X. Wang et al., *Deep learning in object recognition, detection, and segmentation*, Foundations and Trends® in Signal Processing, 8 (2016), pp. 217–382.
[95] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, *Sample efficient actor-critic with experience replay*, arXiv preprint arXiv:1611.01224, (2016).
[96] H. Wei, Y. Wang, L. Mangu, and K. Decker, *Model-based reinforcement learning for predictions and control for limit order books*, arXiv preprint arXiv:1910.03743, (2019).
[97] L. Weng, X. Sun, M. Xia, J. Liu, and Y. Xu, *Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism*, Neurocomputing, 402 (2020), pp. 171–182.
[98] J. Williams and Y. Li, *A case study using neural networks algorithms: Horse racing predictions in jamaica*, in Proceedings of the International Conference on Artificial Intelligence (ICAI 2008), CSREA Press, 2008, pp. 16–22.

[99] L. V. Williams, *Information efficiency in betting markets: A survey*, Bulletin of Economic Research, 51 (1999), pp. 1–39.
[100] R. J. Williams, *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, Reinforcement learning, (1992), pp. 5–32.
[101] S. Winter and M. Kukuk, *Risk love and the favorite-longshot bias: Evidence from german harness horse racing*, Schmalenbach Business Review, 58 (2006), pp. 349–364.
[102] M.-E. Wu, J.-H. Syu, J. C.-W. Lin, and J.-M. Ho, *Portfolio management system in equity market neutral using reinforcement learning*, Applied Intelligence, 51 (2021), pp. 8119–8131.
[103] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita, *Adaptive stock trading strategies with deep reinforcement learning methods*, Information Sciences, 538 (2020), pp. 142–158.
[104] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, *Deep reinforcement learning for automated stock trading: An ensemble strategy*, in Proceedings of the First ACM International Conference on AI in Finance, 2020, pp. 1–8.
[105] F. Ye, X. Cheng, P. Wang, C.-Y. Chan, and J. Zhang, *Automated lane change strategy using proximal policy optimization-based deep reinforcement learning*, in 2020 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2020, pp. 1746–1752.
[106] A. Yezus, *Predicting outcome of soccer matches using machine learning*, Saint-Petersburg University, (2014).
[107] S. Zhang and R. S. Sutton, *A deeper look at experience replay*, arXiv preprint arXiv:1712.01275, (2017).
[108] Z. Zhang, S. Zohren, and S. Roberts, *Deep reinforcement learning for trading*, The Journal of Financial Data Science, 2 (2020), pp. 25–40.
[109] C. Zhou, B. Huang, H. Hassan, and P. Fränti, *Attention-based advantage actor-critic algorithm with prioritized experience replay for complex 2-d robotic motion planning*, Journal of Intelligent Manufacturing, 34 (2023), pp. 151–180.
[110] J. Zou, J. Lou, B. Wang, and S. Liu, *A novel deep reinforcement learning based automated stock trading system using cascaded lstm networks*, arXiv preprint arXiv:2212.02721, (2022).