Deep Reinforcement Learning for Quantitative Finance: Time Series Forecasting using Proximal Policy Optimization

by

David Chiumera

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfilment of

the requirements for the degree of

Master of Information Technology in Digital Media: specialization in Data Science

Department of Information Technology

Carleton University

Ottawa, Ontario, Canada

April 2022

Abstract

Investment and financial management has existed as a field of study for quite some time, with roots in practical application. One of the approaches used in order to assist with investment decision making is the broad area of modelling. Modelling approaches can be found within multiple distinct, but still related, unique fields. These include statistics, machine learning, deep learning, reinforcement learning, discrete mathematics, dynamic programming, and many more. Not only are there multiple modelling approaches belonging to multiple different fields, but there are also different unique problems in investment management. In this work, the focus is on price prediction and concurrent strategy building. The modelling approach chosen for this is of the deep reinforcement learning type, and actor-critic class. Specifically, in this work the proximal policy optimization (PPO) architecture is employed individually on each stock's market history in order to try and solve the price prediction problem. A custom RL environment was built to run the proposed experimental sequence and to test which parameter values should be used in regards to learning rate, discount factor, feature space, action space, and look-back length. These values were subsequently used for experiments on different datasets, exploring the portability of the model, effect of transfer learning, as well as portability of the parameter configuration. The results show our experimental sequence was successful at determining optimal values for training and that the PPO model can be effectively used for the price prediction problem, and in some instances outperform a practical B&H strategy.

Acknowledgments

First, I would like to acknowledge my supervisor Dr. Wei Shi for her dedication, patience, encouragement, and support. Without her guidance, knowledge, and expertise, this dissertation would have not been possible.

I would also like to thank my family for their continued encouragement and support in regards to my research interests and educational pursuits. In addition, I thank my close friend, David R Bellamy, for the continued advice, guidance, and discussion throughout the years.

Lastly, I would like to dedicate this work to my former mentor and confidante of many years, Dr. Robert "Jack" Cornett, who unexpectedly passed away in the Fall of 2017. You shall forever be missed.

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviation	Explanation
TA	Technical Analysis
OHLCV	Open High Low Close Volume
MA	Moving Average
MACD	Moving Average Convergence Divergence
RSI	Relative Strength Index
OBV	On Balance Volume
VR	Volatility Ratio
RoMaD	Return Over Maximum Drawdown
B&H	Buy and Hold
MDD	Maximum Draw Down
S&P500	Standard and Poor's 500
HSI	Hang Seng Index
ETF	Exchange Traded Fund

NLP	Natural Language Processing
DL	Deep Learning
DRL	Deep Reinforcement Learning
MPT	Modern Portfolio Theory
RL	Reinforcement Learning
AC	Actor-Critic
PPO	Proximal Policy Optimization
DQN	Deep Q-Network
DDQN	Double Deep Q-Network
DPG	Deterministic Policy Gradient
PG	Policy Gradient
A2C	Advantage Actor Critic
TRPO	Trust Region Policy Optimization
MLP	Multilayered Perceptron
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
GPT-3	Generative Pre-trained Transformer 3
DF	Discount Factor

LR	Learning Rate
GAE	General Advantage Estimator
SB3	Stable Baselines 3
AS	Action Space

Chapter 1

Introduction

1.1 Motivation

The beginning of model driven approaches really goes back to applied statistical methods from the early 1900's, such as mean reversion, however, these early methods were far from automated [1]. Automated approaches really took off once accompanying advancements in computation were made and exchanges pursued computerizing the markets through the 1970's onward [2, 3, 4, 5, 6]. One method of investing which arose from this digitization of markets, high frequency trading, often employs automated trade executions which classically used simplistic algorithmic rules, but is now applying more advanced methods such as using DRL [7, 6]. More impressively, relying more on accuracy than execution speed to achieve an advantage, in 2001, IBM developed a trading system that was completely automated and able to outperform human traders [8, 9]. The idea of having automated computational machines performing many tasks has been in existence for a great deal of time, but it has only been recently that personal computing power and resources have reached a level where sole individuals can start building these automated model based solutions, or at least components, to attempt to solve investment problems.

In this work we are focused on the approach of predictive machine learning based

methods, as this is the area which significantly overlaps with the areas of Information Technology, Data Science, and Computer Science. Not only this, but the modelling approach to solving investment problems is the most well defined, shows to be a trending, and is relatively novel. Many different machine and deep learning methods exist, so in order to narrow the scope of this work we will be focused specifically on DRL methods [10, 11, 7, 12, 13, 14, 15, 16].

Although there are many different modelling approaches, the modelling approach of focus for this work is that of DRL. The reason for this is multifold, but mainly due to the fact that reinforcement learning is outfitted to address Markovian-Decision Processes, of which the investment process is deemed to be [17]. As well, some of the most noteworthy performance gains in the domain of machine learning have been made via DRL algorithms, such as the significant accomplishments made in autonomous driving, or those made by Google's DeepMind with their AlphaGo project [18, 19, 20]. With a prevalence of impressive results in other problem domains of a similar nature, this work investigates the body of literature applying DRL to problems in quantitative finance, specifically regarding the time series prediction problem. Although the focus of this work pertains to the time series forecasting problem, it does significantly overlap with other problem formulations in the investment space. As a result, not only the time series forecasting problem, but also the portfolio allocation problem, are thoroughly reviewed in relation to DRL. This will be discussed further in the literature review section.

1.2 Problem Statement

The problem of interest for this work, pertains to the time series forecasting problem. Specifically, we are interested in determining whether or not implementing a cutting edge DRL algorithm, PPO, can achieve performance greater than that of a buy and

hold strategy (B&H), given the problem at hand. The B&H implementation details are outlined in the experiment set-up. That said, a brief description of B&H is, purchasing the maximum number of shares possible at the beginning of an evaluation period for the stock under question, and holding these shares until the end of this period. In addition to the above, we also look to test the portability of our trained model, as well as optimal parameter configuration, on a novel dataset, crafted using a previously validated stock selection method. We aim to answer the following research questions:

R1 Can PPO be successfully applied to the time series forecasting problem?

R2 What is an optimal configuration using the PPO architecture in terms of learning rate, discount factor, number of action spaces, input features, and look-back length, given SPY market data?

R3 Does using the PPO algorithm outperform a B&H strategy on the SPY dataset given the evaluation metrics of profit, annualized return, and maximum draw down?

R4 Does the best performing model trained on the SPY dataset generalize well to a new different dataset?

R5 Does performance improve for the best performing model trained on the SPY dataset if transfer learning is performed given a new dataset?

R6 Do the optimal parameters from the previous SPY experiments generalize to a new dataset?

1.3 Major Contribution

The major contribution of this work is applying the PPO algorithm in an attempt to determine its usefulness in solving the time series forecasting problem. Since very little DRL work has been done in regards to this problem to begin with, and none using PPO, a number of experiments exploring basic hyperparameter configurations is first proposed. For these experiments, we use a dataset that is more popular in the literature, which is the SPY ETF daily historical data, which is composed of S&P500 stocks weighted by market capitalization. These experiments are done to see the effects of different input features and lengths, as well as different action space sizes. Next, experiments are conducted to determine learning rate and discount factor values, while input features, action-space size, and look-back are kept constant. After these initial experiments, of which will be compared to a B&H strategy during testing, we will move on to using the best performing model on a new dataset, consisting of stocks with high Piotroski scores and a beta value between 1.0 & 1.5. Piotroski scoring is a value based method of stock ranking and selection, and beta is a measure of a stocks volatility compared to the general market as a whole, with values above 1 indicating greater volatility and values below 1 indicating less. Lastly, using the optimal parameter configuration from the SPY experiments, training, validation, and testing is performed for each stock in the new Piotroski dataset, in order to test the portability of the optimal parameter configurations from the SPY experiments. It is believed that doing these experiments will be a good measure of whether using such an implementation is of any value, since Piotroski ranking using a B&H strategy has been previously experimentally validated [21].

In summary, this works contributions can be summarized into two major points, each with their own sub-contributions:

C1.0 Apply PPO algorithm for time series forecasting

C1.1 Optimal Look-back, input feature, and action space size experiments

C1.2 Optimal learning rate and discount factor experiments

C1.3 Comparative evaluation with a B&H strategy

C2.0 Test portability of the best PPO model on Piotroski dataset

C2.1 Comparative evaluation of the best SPY model with a B&H strategy

C3.0 Test the effect of transfer learning on the best performing model trained on the SPY dataset, given a new different dataset.

C3.1 Comparative evaluation of the best SPY model with the best SPY model after transfer learning using the Piotroski dataset, both compared to a B&H.

C4.0 Test portability of the parameter configuration from SPY experiments on Piotroski dataset

C4.1 Train and validate model for each stock in Piotroski dataset, then test best model for each and compare to a B&H

1.4 Background

Traditionally, investment management has been a heavily debated topic, with many professionals and gurus alike all making claims that one investment approach is better than the next. The investment tools that are employed today can be placed into a few general categories, intuitive and heuristic based, Technical Analysis (TA), fundamental analysis, and model driven. Some investment funds combine multiple of these in order to try and average out the competitive advantage, or alpha, of each method, resulting in an almost living ensemble of approaches composed of people and machines.

1.4.1 Investment Tools and Styles

Intuitive

In the intuitive and heuristic based approach, the data comes from everywhere and anywhere, with potentially zero standardization. Investors and fund managers process this information, and follow general heuristics they have developed from their experience and training, which guides their decision making process. This is an approach many employ, institutionally affiliated or not, as it requires very few technical skills in order to execute. It is by far the least well defined approach, regardless of the implementation, and can be thought of as a true generalist strategy. As a result, very little research has been done on this method, since each individual or institution is different, with their methods poorly defined, kept secret, or both.

Technical Analysis

The timeline regarding the development of TA is highly debated, however early elements can be found in 17th century Holland with the accounts of Joseph de la Vega in his work Confusion of Confusions, where he introduced some of the earliest tools for this category of approach to investment [22]. Just as noteworthy were the later developments by individuals in Asia contributing things such as candlestick patterns, a major tool used in TA still to this day [23, 24]. Fundamentally, today TA is a method of summarizing market data. The proponents of this technique employ pattern matching, and use time series data such as open, high, low, close, and volume (OHLCV). This data is then used to develop abstractions or representations, using these variables, in order to be employed as indicators for signalling certain market dynamics and/or buy and sell positions. These indicators are also sometimes used as features for stock selection, even in the area of machine learning [25, 26]. Some examples of these indicators or representations would be moving average convergence

divergence (MACD), relative strength index (RSI), and on-balance volume (OBV), to name a few. Examples of visual patterns in charting price and volume data would be things such as head and shoulders or cup and handle formations [27, 28]. This approach has more well defined methods than the heuristic based approach, since the data used is standardized, and concrete rules can be put in place in regards to buy and sell positions using these indicators and patterns.

Fundamental Analysis and Value Investing

Fundamental analysis, like technical analysis, focuses on a finite set of variables. These variables are used to summarize company fundamentals such as accounting data, like net income and operating expenses, in combination with price and even economic data. The goal is to determine a fair valuation for the underlying asset based on this data. What is considered a good or bad combination for analysis given these variables and their respective values, is rooted in business, accounting, and economic theory[29]. Value investing uses the tool of fundamental analysis, in order to find undervalued securities [21]. One example of this would be Piotroski scoring, which specifically uses a rule based scoring system to rank securities based on their level of being undervalued, with a focus on three categories of evaluation: profitability, liquidity and funding sources, as well as operating efficiency. Fundamental analysis is often used for strategies associated with less trading, such as long-only value based approaches. This is at least in part due to the fact that most of the variables being used in this paradigm only change on a quarterly or yearly basis, and thus sufficient amounts of new data is only available on a quarterly basis at most. This approach can be as well defined as technical analysis based strategies [30, 29].

Programmatic and Model Based

Lastly, there are programmatic model driven approaches; these approaches use statistical and machine learning methods to help predict certain variables. These methods use all different types of financial data, with the most common being market and fundamental data, likely as a result of availability and standardization. Although appealing, it requires a great deal of theoretical and practical knowledge of financial markets, economics, statistics, and computer science. Many funds are now banking on this approach to get a competitive edge, with large funds acquiring leading experts in the field of machine learning and AI in order to assist with their efforts [31]. Two good examples of long standing successful funds using primarily model based approaches are Two Sigma, and Renaissance Technologies. Both these funds have competed with and outperformed some of the world leading funds, despite using alternate approaches to most in the investment space. From this alone, it can be concluded that although difficult, it is possible to successfully manage investments using complex model based systems, and even obtain a significant advantage over the average market performance, otherwise known as alpha. This model based approach is likely the most well defined, yet difficult to implement, as it relies strictly on quantitative and technological means, and requires fairly extensive knowledge in overlapping disciplines [32].

As previously mentioned, sometimes these methods are combined together. For example, you could have a team of economists assessing general macro and micro economic climates, accounting and finance experts reviewing some fundamental data, and have developers building dashboards and widgets, including predictive tools, all to be used by expert traders to assist in their decision making process. Each method has its own pros and cons, but with the rise of advanced modelling and machine learning methods alongside personal computing power, automated and predictive means are

becoming much more common and realistic to employ.

1.4.2 General Problems and Data Types

Types of Data

The categories of data to be used for the investment process can be summarized into a few categories. These are: market, fundamental, and sentiment data. Market data simply refers to the time series data for an asset or collection of assets, which includes OHLCV data on different intervals (monthly, weekly, daily, hourly, etc.), as well as any representation that can be constructed with these variables. Fundamental data refers to the accounting data related to the asset at hand. This type of data can be found on balance sheets, as well as cash flow and income statements. This data category is infrequently updated, with changes occurring at most every quarter or three months. Lastly, there is sentiment data, which is a category that includes market and fundamental data, since sentiment can arise from this type of information. That said, when talking about sentiment analysis in the modelling literature, it usually excludes these types during practical implementation, since the methods used for assessing text based sentiment are not usually outfitted for numeric time series data. This data category is usually composed of news articles, twitter and reddit feed, transcripts, and analyst recommendations [33]. Sentiment data is likely the hardest to work with, since there is no standardization of reporting. Not only this, but in the realm of modelling and classification, assessing sentiment is fundamentally a natural language processing (NLP) task. Training such models requires massive amounts of compute power to generate state of the art models, such as the Generative Pre-trained Transformer 3 (GTP3), which has been estimated to have cost up to 4-12 Million USD for a singular training run, not considering the 250 Million USD price tag of the compute cluster used [34]. In this work, we will be focusing solely on market

and fundamental data, due to these categories being standardized by exchanges and regulatory bodies around the globe. One could argue that the sentiment output of a model is numerical and should be included, however this would require evaluating each model and dataset it was trained on, as well as the input dataset to be used for classification (reddit, twitter, news, etc.), which would extend the scope of this work too far.

Types of Problems

Although there are a great deal of formulations of investment problems, there are a few key ones which encompass most of the space and the potential sub-problems that can be formulated. These problems are the security selection, time series prediction, and portfolio management problems. Some elements of these problems overlap, such as the data being used or the sub problems they address, however each one of these tasks is unique, especially in regards to practical implementation, and the ways they can be formulated will be discussed below.

Security Selection

A security is simply an investment instrument which gives ownership to the holder and encompasses stocks, bonds, and many more. The problem of security selection can be verbally described as the following "given a set of securities or investment instruments, which should be invested in or have the highest expected return". Alternatively, the problem can be formulated mathematically as: given a list of securities C, $[c_1, c_2, ...c_n]$, which subset of C of size x, will be most profitable t steps into the future. This problem has many nuanced formulations, since the duration of holding the selected assets and number of assets selected can vary. Prediction or classification of which assets to buy, based on expected returns, could be recalculated daily if incorporating daily market data for each asset. In this example, the modelling goal itself would

be to construct a day trading system, given a certain number of input features. A day trading system is simply any system which produces trading decisions on a daily frequency. In contrast, some works may outfit their training in a different way, in which the goal is to predict a longer-term performance, such as a year out, while potentially ignoring market data.

Time Series Forecasting

Time series forecasting has existed for quite some time and has been applied to a variety of fields, such as econometrics, financial mathematics, and signal processing to name a few [35]. Time series analysis, different from forecasting, is a collection of various methods which aim to describe characteristics of the data at hand [36]. Time series forecasting on the other hand consists of methods used to model and predict future values based on previously observed data [37]. Time series analysis can often act as an informative procedure to help determine which kind of forecasting approach should be taken. The types of forecasting methods can be summarized into multiple categories depending on who you ask, however we will be separating classical methods in the field of statistics from modern methods of prediction in Machine Learning, and in particular deep learning methods [38]. These two categories have some overlap, since ML models rely upon statistical concepts, however, the key difference is that classical statistics/stochastic data modelling assumes that the data generated is reliant upon a stochastic data model, whereas machine learning methods use algorithmic models which treat the data generating mechanism as unknown. In this work, we will be focusing solely on the algorithmic/machine learning paradigm, in specific deep reinforcement learning, since this area is less explored in the domain of time series forecasting in stock trading, and more focused on predictive accuracy rather than explainability. In the framework of stock trading, time series forecasting is used to predict when to buy and sell, or when a stock will be moving upwards or

downwards given a certain predictive window (ex. 1 month, 1 day, 4 hour, etc.). In some novel Deep Reinforcement Learning (DRL) formulations the problem definition also includes the concept of position sizing, which is the ability to scale the degree of buying and selling for each action.

Portfolio Management

The portfolio management problem can be formulated in different ways. However, modern portfolio theory (MPT), published by Markowitz in 1952, has remained the modern theoretical paradigm, despite not being used in practice [39, 40]. Markowitz' formulation extends the theory of diversification, which states that owning multiple different types of assets is less risky than owning only one type. His theory assumes that investors are risk averse, and that if one portfolio has the same expected returns as another, investors would prefer the portfolio which carries less risk. His model posits that portfolio return is the proportionally weighted combination of the assets expected return, and that portfolio volatility is a function of correlations of all asset pairs. Many critiques of MPT exist, however one critique of this formulation that is particularly relevant to our work, is that it relies on expected values, which is a generalization of the weighted average, and may not be the best way of forecasting expected returns of an asset [41].

As is probably apparent, each one of these problem formulations has an element of timing involved as well as some overlap with each other when considering the entire investment process. That said, each formulation can be thought of as practically distinct, especially in relation to predictive modelling. It would be convenient if fields converged upon agreed formulations of investment problems and approaches to solving them, however as with most interdisciplinary fields, different perspectives remain. In this work in particular we will be focusing and employing deep reinforcement learning methods with a focus on the time series forecasting problem.

1.5 Thesis Overview

In Chapter 2, we review preliminaries as well as introduce related works from the field attempting to solve the time series forecasting and portfolio management problems using DRL. In Chapter 3, we describe the dataset and model being used in this work, as well as include the assumptions made for our approach. Chapter 4 goes through all our experimental results and the analysis of such. Lastly, Chapter 5 concludes on our findings and suggests future directions for further work.

Chapter 2

Preliminaries and Related Work

In this section we will outline the preliminary information needed to understand the approach taken in this work, before moving onto reviewing the relevant literature employing DRL techniques in regards to the time series forecasting and portfolio management problems. Although the focus for this work is in fact on the time series forecasting problem, data is needed in order to train our algorithm. Selecting data in this situation is actually akin to stock selection. Not only this, but if one takes a time series forecasting approach, but with multiple securities, it automatically becomes the portfolio management problem since an extra dimension of choosing stocks from some predefined list is added. Considerable performance gains could be made by combining the problems, which would make this a worthwhile approach, forgoing the need to approach from a single stock time series forecasting set-up. As a result of this, we also investigate the portfolio management problem, with a particular focus on the stock selection methods used, as well as the ability to scale the solutions. If the stock selection method used to construct our new dataset is the same as those used in the DRL portfolio literature, and show scalability of the portfolio size, our contribution of using a novel dataset selected via a previously validated quantitative method, will be less worthwhile. This is because if the problem can be approached as a multi-stock problem successfully, a single stock approach would not necessarily

be preferable, as the portfolio problem in practice is inseparable from the investment problem as a whole, unless only looking to invest in a single stock. When approaching the investment problem set-up from a portfolio management perspective, aka multi-stock, the action and state space becomes much more complex. In a single stock scenario, there is one dimension used for the action type (buy/sell), and another for the action size. Alternatively, with the portfolio management set-up, an additional dimension is added, with its size equalling the number of stocks in the portfolio. As can be seen, the portfolio problem is n times larger, assuming everything else is the same, where n is the number of stocks considered for the portfolio; this is the same for the state space. With this in mind, if literature already exists showing that there is no limitation to approaching the problem set-up in a multi-stock fashion using PPO, it would be more reasonable to use this formulation, since it takes into account position sizing, timing, as well as portfolio management. However, as will be shown in further detail, this is not the case.

2.1 Reinforcement Learning

Reinforcement learning (RL) is a type of machine learning paradigm distinctly different from supervised and unsupervised learning. It is a learning paradigm that self-learns through interacting with a dynamic environment and attempts to learn a policy function which maps observations to actions while maximizing for its defined reward function [42]. The main components of the RL paradigm are the agent, action, reward, and environment, as well as the observations it generates. The RL process progresses over a trajectory of time steps, t. With each time step the agent is presented with an observation state, S_t, generated by the environment. The agent takes an action, A_t, which interacts with the dynamic environment to generate a reward, R_t, and the next observation state, S_{t+1}. This recursive loop continues for the entire

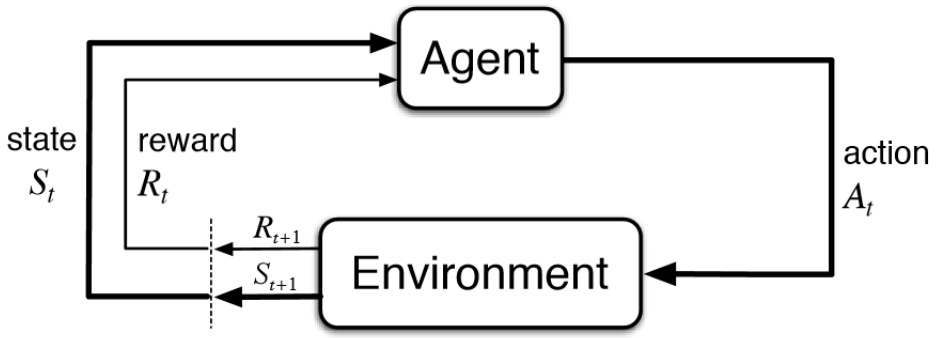

Figure 2.1: Standard Reinforcement Learning Loop [43]

learning process, as can be seen in the figure below, with the hope of converging on a well performing policy.

The goal of RL is to find a policy that maps observation states to actions in a manner to maximize for rewards. The general form of the return function can be seen below:

$$r_t = \sum_{k=0}^{\infty} \gamma_k R_{t+k} \text{ [44]}$$

r_t, the return, is the sum of all discounted future rewards that the agent receives from time t onward, $k \in [0, \infty]$. Further, $\gamma \in [0, 1]$, is the discount factor, and R_{t+k} is the reward, for the current value of k. RL uses a function called the value-function, the purpose of this function is to generate expected rewards given an observation state S. The value function can be expressed as $E(R_t|S_t = S)$, where E represents the expected value. This can be further denoted as $V^{\pi}(S)$, where V stands for the value of state S, given policy π. Given all the possible variations of $V^{\pi}(S)$, there exists an optimal value-function which can be denoted as $V^*(S) = max_{\pi} V^{\pi}(S)$, where $\pi^* = argmax_{\pi} V^{\pi}(S)$ is the optimal policy which maximizes possible value for state S. Since this representation excludes the action itself, a second function called the action value function is presented. This function is alternatively known as the Q-function, which takes state-action pairs and produces the value of rewards. We

can re-formulate the previous optimal policy function as $\pi^* = argmax_A Q^*(S, A)$, where Q^* stands for the optimal value for the Q function, or action-value function. Furthermore, using the Bellman equation, a definition for Q^* can be defined as:

$$Q^*(S, A) = R_t(S, A) + \gamma E_{S'}[V^*(S')] = R_t(S, A) + \gamma \sum_{S' \in S} P(S'|S, A)V^*(S') \textbf{[44]}$$

$R_t(S, A)$ is the immediate reward for taking action A at state S and $\gamma E_{S'}[V^*(S')]$ is the expected discounted reward after transition to the subsequent state S'.

One foundational concept in reinforcement learning is that of exploration vs exploitation. Exploitation refers to selecting the action which gives the highest reward given what is currently known about the environment, i.e. operating on some potentially non-optimal policy π. Exploration on the other hand is taking an action in order to gain more knowledge. Both exploration and exploitation can be approached in different ways. That said, one can intuitively see how if an agent is to always be entirely greedy, always taking the action which generates the highest return given some initial policy, the agent will very likely fail to find an optimal policy for the given problem, and its success will be entirely random based on where the agent begins. Doing such will result in the agent not having experimented enough with its environment, which could very well result in the lack of discovery of other policies which may be much better to exploit. With a stochastic task, of which the generative model of the environment is not known, each action within the action space must be tried many times in order to be able to get an appropriate estimate on the expected reward. Thus, on the other hand, it could also be seen why only exploring could be very problematic, as there would not be enough experimentation with a given single policy.

The problem of exploration vs exploitation has been studied by mathematicians for decades, and as a result many approaches exist to this particular problem today.

That said, there are common state-action value function updating strategies, such as Sarsa and Q-learning, as well as common action selection strategies, such as ϵ-greedy and Boltzman selection [45, 46, 42]. The rate of exploration can be represented by ϵ, and the rate of exploitation as $\epsilon - 1$, where ϵ ranges between 0 and 1.

2.1.1 Model-free vs. Model-based

There are two main approaches to RL based systems, model-free and model-based. These approaches are specifically referring to how the RL algorithm is learning. In model-based, the agent uses predictions of the environment responses, using a state transition probability function $T(S_{t+1}|S_t, A_t)$ which predicts the next state, S_{t+1}, given a state-action pair. Model-free RL on the other hand focuses directly on optimizing its policy via the state-action value, or q-function. When a model isn't available, model-based is impractical since the action and observation states are very large as it is infeasible to learn all the transition state probabilities prospectively. RL in general is often regarded as sample-inefficient, in that it takes a large number of examples to converge on a stable well-performing policy. For this reason, as well as other factors which will be discussed later in this work, we will be focusing on model-free methods only. Model-free methods belong to two distinct categories: on-policy and off-policy, of which will be discussed next [47].

2.1.2 Model Free: On-Policy vs Off-Policy

The key difference between on and off policy based model free RL methods relies upon whether or not the same policy to evaluate decision making is being used to make action decisions. In on-policy methods, the same policy to evaluate actions and rewards is used to make the action decisions themselves. In off-policy methods, different policies are used to make the action decisions and evaluate the performance.

Put another way, on-policy methods learn action values not for the optimal policy, but for a near-optimal policy that it still explores. Alternatively, the off-policy approach is formulated to use two policies, one that is learned and becomes the optimal policy, and one that is more exploratory. The learned policy is usually referred to as the target policy, and the policy used to generate the actual actions is termed the behavior policy. [42] Generally speaking, off-policy algorithms are more sample efficient yet less stable than on-policy [47].

2.1.3 Deep Reinforcement Learning (DRL)

DRL encompasses the use of neural networks to approximate various functions common in reinforcement learning algorithms, such as the action-value function. The reason this is so common and has been so successful likely has to do with the fact that neural networks are universal approximators which can approximate any measurable function to any desired degree of accuracy, as proven in [48]. When action and observation state spaces get large enough, it becomes infeasible to use classical RL methods and converge fast enough on any meaningful and well performing policy, thus using something such as a neural network to approximate the action-value function for faster convergence becomes useful. Many of the most impressive systems achieving state of the art performance today in RL employ neural networks of some sort, and thus belong to the sub-category of RL known as DRL. Since deep learning (DL) is its own entire field of study within machine learning, as is RL, there are a great number of architectural formulations that have been built for DRL, each with their own advantages and disadvantages [11, 49].

2.1.4 Actor Critic Learning

Aside from the distinctions of being model-based or model-free, as well as whether the RL algorithm being on-policy or off-policy in the model-free category, there is a third distinction for RL methods. This final distinction has to do with whether the RL algorithm is approximating the state value and state-action value functions, the policy function, or both. In the value-based category, the state value or state-action value function is approximated. In contrast, in policy based methods the policy function is estimated. In actor-critic (AC) methods both are used. Specifically AC methods learn a parameterized value function (the critic) to estimate the state-action value, as well as a policy function which selects appropriate actions (the actor). This final method is thought to be beneficial to use as knowing or having an estimate of the value function can assist in the policy update and approximation [50].

2.1.5 Proximal Policy Optimization (PPO)

The PPO DRL architecture was originally implemented by OpenAI, in order to improve upon the already existing trust region policy optimization (TRPO) artchitecture [51]. PPO is a model free, on-policy, actor-critic method. As a result, it is thought to be less sample efficient, however more stable. There are a number of different implementations of PPO, employing different variations of neural nets and activation functions. Regardless of the specific implementation, this architecture has shown to perform very well on a number of tasks, one such example being the OpenAI Five project which has beaten some of the best Dota 2 players in the world [52]. Since DRL methods can often be very fragile to hyperparameter configurations, the extra added stability of PPO could potentially be very useful in an environment as chaotic as the stock market.

2.2 Time Series Forecasting

As previously mentioned, time series forecasting is a foundational problem in many different fields, with stock trading and investment management being one of them. Since DRL is a fairly new paradigm of RL, and this area of applied work is an interdisciplinary one, very little consistency was seen between each paper, making it very difficult to determine where improvements could be made. As a result, we go through each paper in our review independently, highlighting key details for each. In particular, we will be focusing on the factors listed below:

1. Data
    (a) Source
    (b) Stock Selection
    (c) Dataset Size (Train, Test, Validation)
    (d) Frequency (e.g. hourly, daily, etc.)
    (e) Input/Observation State Features
        i. Types
        ii. Duration or Look-Back (eg. 100 days, 200 hours, etc.)
2. Environment Set-Up Details
    (a) Are Training Samples Fed Sequentially or at Random?
3. Reward Function
4. Architecture
    (a) Model Type
    (b) Specifications

5. Hyperparameter Specifications
    (a) Learning Rate
    (b) Discount Factor
6. Technological Specification
    (a) Is the Code Available?
    (b) Which Languages and Libraries are Used?
7. Findings

First, we look at a paper titled *Action-specialized expert ensemble trading system with extended discrete action space using deep reinforcement learning* from 2020. The dataset used for this work was acquired from Yahoo-Finance and consisted of the daily price history for three well known indexes, specifically the Standard and Poor's 500 (S&P500), Hang Seng Index (HSI), and Eurostoxx50. This dataset spans from January 1987 to December 2006 for the training period, and from January 2007 to 2017 for the testing period. No validation set nor specific input features, other than price, are mentioned. For the input feature length or look-back, it is set to 200 time steps or days. Training is performed on each index separately, thus each environment consists of the price data of one of the three indices, other than this nothing is mentioned in regards to the observation state space, and it is not clear whether training samples were selected at random or sequentially. Three separate reward functions are used in this paper, profit, Sortino, and Sharpe. All three of these functions consider not only the return from one day to the next but also the return over the last 100 time steps. The authors state this is preferable since it considers both long and short term profits. The DRL architecture used in this work was that of a Deep Q-Network (DQN), ensembled for specific actions (buy, sell, hold), using an epsilon

greedy action selection strategy, while also employing experience replay memory. The hyperparameters of relevance in regards to our work are the learning rate (0.0001), discount factor (0.85), and mini-batch size (64). Unfortunately, no code was made publicly available for this work, and thus very specific implementation details couldn't be reviewed. After testing the effects of different sizes of discrete action spaces, variable lengths of training data, different reward functions, and different ensembled architectures they conclude the following: that a 21-action space size, longer training data-sets, a profit or Sortino based reward function, and using their action specialized expert ensemble, resulted in the best performance. It should be noted that although this paper did not compare to a buy and hold strategy, only different more simplified architectures, it was further investigated and does in fact outperform a buy and hold strategy for their defined testing period [12].

In another work, *Adaptive stock trading strategies with deep reinforcement learning methods*, the authors use a dataset consisting of daily price history for a number of stocks. Unfortunately, the work doesn't mention where they obtained their data, however they do state which securities it consists of and the method used for selection. Securities were assessed based on volatility and whether or not it was trending, and then randomly selected from each geographical market category (UK, US, and Chinese). Their data spans from 2008 to 2018, with a training period from 2008-2015, and testing between 2016-2018. No validation set was mentioned. The input observation space features were moving average (MA), exponential moving average (EMA), moving average convergence/divergence (MACD), bias (BIAS), volatility ratio (VR), and on-balance volume (OBV). These were further abstracted by a Gated Recurrent Unit (GRU) representation learner before being fed to the agent. The raw input feature length was set to 300 days or time steps. It was not specified whether training was performed on each stock individually, nor whether the training examples were sampled at random or sequentially. The reward function used in this work was

set as the Sortino ratio considering the rewards of the last 200 time steps. The DRL method used in this work was two-fold, with a custom implementation of DQN as well as deterministic policy gradient (DPG). All the neural nets used in this work were 2 layers deep, each consisting of slightly different sizes. No hyperparameter values are specified in this work, and since no code is available, couldn't be investigated further. This work compares their novel implementation to the Turtle Strategy, as well as a model implemented by another work, showing that in almost all cases their implementation outperforms the others. However with that in mind, it should be noted that this work also did not compare to a buy and hold strategy [53, 13].

Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading also uses data taken from Yahoo Finance, of which spans 2000 days for training and 3000 days for testing. The stocks used are from the S&P500, separated into two sets. The first consisting of S&P500 stocks in different trending categories, and a second to test generalizability, which consisted of 5 stocks from each of the 11 sectors in the S&P500. Again, as with the last two works, this paper doesn't specify the use of a validation set. This paper, as with the last reviewed, also specifies the use of a representation learner/feature abstractor which processes the raw input before passing data to the agent. The raw data consists of price, volume, cointegration values, and many technical indicators. The raw input feature length is not specified, however the DL representation learner does have a sliding temporal attention window of size 30. This work, as with the last, fails to mention whether training is done with each stock separately, and whether the training examples were sampled at random or sequentially. The reward function used for this work is defined as immediate and long term profit, while also including transaction costs of the trading actions taken. The type of DRL architecture used was a custom form of policy gradient (PG), employing a learning rate of 0.0001 and a mini-batch size of 32. The action space is set to long, short, and hold represented by [1,-1,0]. The paper compares

the authors custom implementation to B&H, as well as model implementations from other older papers. It is demonstrated that their system outperforms those they compare to on total profit, annualized return, annualized Sharpe ratio, as well as transaction times. As with the previous works above, no code was provided [14].

Another paper, *Deep Reinforcement Learning for Trading*, from 2020 published by members of the prestigious Oxford-Mann Institute, attempt to use DRL for this problem formulation as well. This work however is unique in the dataset they use, which comprises 50 daily ratio adjusted futures contracts from 2005-2019 obtained from Pinnacle Data Corporation's Continuously Linked Commodity Contacts database. This dataset is unique in that it is composed of futures contracts which are inherently different than trading shares of a company. The training and testing data includes the period from 2005 to 2011 and 2011-2019 respectively, with retraining happening every 5 years during testing. This paper is also unique in the fact that it uses a validation set, which is set at 10% of the training data. The dataset itself, nor researchers, employ any specific contract selection. That said, the contracts fall into one of the following distinct categories: commodities, equity indexes, fixed incomes, or foreign exchange. The observation state comprises the following features over the last 60 days: normalized close prices, returns over the past one to three months, return over the last year, a number of technical indicators, and an exponentially weighted moving standard deviation. It isn't clear whether training samples for each contract were fed sequentially, however it is known that training was set up in a manner where only one contract is considered per step. The reward function for this work is a modified version of profit which considers the weighted profit made over some time, t, while also incorporating a volatility factor. The architectures used in this work pertain to DQN, PG, as well as advantage actor critic (A2C), with the first two operating on discrete action spaces [1, 0, -1], and then later on a continuous one [-1, 1]. The mini-batch size was 64 for DQN and 128 for A2C, each using different sizes of long short-term

memory (LSTM) neural nets, using a Leaky ReLu activation function. The learning rate implemented for all networks but the critic network in A2C were set to 0.0001 with the critic network having a rate of 0.001. All discount factor values were set to 0.3, and the optimizer used was Adam. No code was made available for this work, and thus technological specifications like libraries used, could not be determined. It should be noted that this paper used a portfolio approach in their evaluation, despite not training their algorithm on the portfolio management problem formulation. This was executed by testing each contract independently, and comprising a portfolio with equal weight for each contract tested. The model performance was evaluated on annualized expected returns and its standard deviation, downside deviation, Sharpe, Sortino, MDD, Calmar, Calmar Ratio, % of positive returns, as well as the average profits/losses ratio compared to B&H, sign of return strategy, and a MACD signal strategy. The results of this paper demonstrate that on this dataset their DRL models are able to outperform their benchmark strategies on most metrics, with their DQN model performing the best [15].

The next paper, *Application of deep reinforcement learning in stock trading strategies and stock forecasting*, is unique in the sense that the dataset used was actually specified to not be consistent in the time dimension. What is meant by this is that some stocks had longer histories than others. The dataset is obtained from Kaggle and was split into a train to test set with a ratio of 4:6 for each stock, with 10 stocks randomly selected. No validation set nor publicly available code is mentioned for this work. The observation state available to the agent consists of price and volume features, with no mention on the look-back for each. In addition, there is no mention on how training examples were fed to the agent. The reward function used for this work was defined as the immediate profit for a single time-step and action, for all three model types tested. The model architectures used were DQN, Double DQN (DDQN),

and Dueling DDQN, each employing experience replay. Unfortunately, hyperparameter values such as learning rate and discount factor were not specified. The paper evaluates their models using a portfolio of the 10 stocks randomly selected, each with equal weighting, evaluated solely on profit. This is compared to the same portfolio but using a B&H as well as KD technical indicator strategy for each stock. They conclude using these evaluation metrics that their DQN model performs the best [54].

The final paper reviewed was *A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning*. This work uses a dataset comprised of the S&P500 index, as well as JP Morgan and Microsoft stock. The training set spans between 2012 and 2018, with the testing and a small validation set taken from between 2018 and 2019. The dataset frequency was hourly, with their testing period allowing for intraday trading. Even more interestingly, this work comprised two separable components with the first being a convolutional neural network (CNN) feature abstraction layer, comprised of a 1000 CNN's. These take in images before passing the output to the ensemble of DRL agents in the second layer, of which is comprised of agents that receive the raw price data without abstraction. These images are generated using an algorithm which converts time series data (open, high, low, close) into Gramian angular field images. The reward function used for this approach was defined as profit over a one day period, allowing for both short, long, and hold positions for the action space. The model type used for the DRL component was a DQN, using an epsilon greedy strategy, and a fully connected neural network. Experiments testing for different rates of exploration and exploitation, as well as optimizer algorithm and activation function selection were run. These experiments showed that the best performance, when assessed by return over maximum drawdown (RoMaD), was when using an exploration rate of 0.5, ADADELTA optimizer, and TanH activator. Unfortunately the learning rate and discount factor were not reported

in the publication. That said, the authors did make the code publicly available, and it was discovered they used a learning rate of 0.001 and a discount factor of 0.99 for their optimal configuration. The project was done using Python and associated libraries such as keras-rl. Lastly, the authors compare their whole system to their first layer only using a majority voting system summarizing the 1000 CNN trading signals, first layer only with a 50% threshold of agreement, first layer only with a 70% threshold of agreement, other models from 2 other papers using multiple time intervals, and finally a B&H strategy. The evaluation was made using trades out of total trading days, precision, MDD, Return, and RoMaD, with the exception of when comparing to other papers. Their results conclude that their system is the best performing given all evaluation metrics [55, 56, 16].

In regards to the papers reviewed, the goal for this work is to keep what has been demonstrated to work well, while also exploring that which has yet to be fully investigated. One aspect in regards to the experimental set-up which was demonstrated to work well was that of the reward function being defined as profit. Another would be the action-space sizing from the same work demonstrating larger discrete action space sizes leading to better testing performance [12]. Additionally, due to very few papers mentioning it, and the level of credibility the Oxford-Mann Institute has in regards to this area of research, we chose to use Adam as our optimization algorithm, especially since this is not an element we will be exploring experimentally. In addition, the stable implementation of PPO uses the same choice, as discussed later [15]. More so than what was demonstrated to work, elements which were missing from some or all of the papers reviewed were incorporated into this work. This is true in regards to the learning rate, discount factor, action space, look-back, and effect of technical indicators. For the learning rate and discount factor values, although some of the works reviewed specify the value used, no experiments were included to explore these values. Since these factors can take on a wide range of values and can be critical

to the achieved performance, it is decided this would be worthwhile to investigate. In regards to the action space, although one paper did explore the effect of sizing, it did not explore whether it was best formulated on a percentage of available cash or using shares. Look-back length was also an element not explored by any of the papers reviewed, even for those which specified it. As a result, this is also added as an element of investigation for our experiments. Lastly, although some papers do use technical indicators as features, none of the papers explored whether including them was preferable. Since technical indicators are adamantly used in practice by certain investor types, and there isn't a clear consensus from the papers reviewed, this is also included as an element of investigation.

2.3 Portfolio Management

This section reviews the DRL literature in regards to the portfolio management problem. This section will contain less depth compared to the previous section, as the primary focus for this work is the time series forecasting problem. That said, we still investigated these works thoroughly, with a particular focus on the stock selection method used, model types, action and observation space complexity to assess scalability, as well as if a feature abstractor or representation learner was used to process the raw observation signal before passing it to the RL agent. All the results of our review can be found below in Table 2.1, 2.2, 2.3, 2.4, and 2.5.

Ref	Year	Stock Selection Method	AS Complexity	Types
[57]	2017	Crypto: 11-most volumed non-cash assets	11 securities Continuous for sizing	DPG + CNN, LSTM, RNN
[58]	2018	5 Randomly chosen chinese stocks	5 random stocks Continuous for sizing	DDPG, PPO, PG
[59]	2017	Crypto: Coins selected based on trading volume	12 Coins Continous for sizing	DPG + CNN
[60]	2019	The selection is qualitative with a balanced mix of volatile and less volatile stocks	6 Securities Continuous for sizing	DDPG

Table 2.1: Portfolio Optimization Papers Utilizing Deep Reinforcement Learning 1

Ref	Year	Stock Selection Method	AS Complexity	Types
[61]	2020	Crypto: Risk assets with top month trading volumes in Poloniex.	4 DataSets (#assets = 12, 16, 21, 44) Also appy to S&P500 (#assets = 506)	DPG
[62]	2019	Dow Jones 30	DJ30 Stocks	Modified DDPG
[63]	2020	Minimum variance portfolio of 6 stocks from the overall stocks list from another paper	6 Stocks Continous for sizing	DDPG Generalized DPG PPO
[64]	2019	Crypto: Coins with the highest trading volume are selected	8 Coins Continous for sizing	Custom DQN CNN for q-network

Table 2.2: Portfolio Optimization Papers Utilizing Deep Reinforcement Learning 2

Ref	Year	Stock Selection Method	AS Complexity	RL-Type/ Architecture
[65]	2019	4 unspecified assets	4 Assets unspecified Continuous	PG + CNN
[66]	2021	S&P500 - subset of 15 randomly selected stocks	15 Securities Continuous	Autoencoder DDPG
[67]	2019	10 big name American stocks	10 US Companies Continuous	VPG, DDPG
[68]	2021	30 stocks on DJIA index	DJIA 30 companies Continous	DDPG
[69]	2021	Crypto: Only assets that can be directly exchanged for USDT	3-85 Assets (dynamic) Continuous	PPO

Table 2.3: Portfolio Optimization Papers Utilizing Deep Reinforcement Learning 3

Ref	Year	Stock Selection Method	AS Complexity	RL-Type/ Architecture
[70]	2020	Ten stocks with greatest weight in the first trading session for BOVA11 ETF	10 Assets Continuous	DDPG
[71]	2018	Subset of S&P500 index stocks selected based on data quality	50 Stocks Continuous	Preprocessing+ A3C
[72]	2021	DJIA - 5 stocks (second level)	5 Stocks Continuous	Abstractor+ PG, AC, PPO
[73]	2019	S&P500 - top 100 companies with the highest combined ESG	100 Stocks Continuous	Abstractor+ RL

Table 2.4: Portfolio Optimization Papers Utilizing Deep Reinforcement Learning 4

Ref	Year	Stock Selection Method	AS Complexity	RL-Type/ Architecture
[74]	2020	Bitcoin + 10 crytpo coins + "HighTech" stocks (9) - assets with the most news	20 Assets Continuous	Abstractor+ PG
[75]	2020	Synthetic Data	Variable Continuous	DDPG
[76]	2020	Low relevance US risk assets	6 Stocks Continuous	DQN + CNN for q-network
[77]	2020	Dow Jones 30	30 Stocks Continuous	PPO, A2C, DDPG

Table 2.5: Portfolio Optimization Papers Utilizing Deep Reinforcement Learning 5

As is highlighted by the AS complexity column in Table 2.1, 2.2, 2.3, 2.4, and 2.5, it can be seen that it is much larger for the portfolio management problem than the time series forecasting one. This can be easily demonstrated by a simple comparison of the two problem formulations as done earlier in this chapter. From the papers reviewed, it was discovered that the action space for the portfolio problem doesn't exceed more than a handful of different securities when there isn't a second representation learner component to the system. It is well known that DRL is deemed to be fairly sample inefficient in general, and thus likely explains why so few securities are used. This issue in DRL is due to the small gradient updates necessary for the neural networks used to achieve accurate performance, as well as a result of a weak inductive bias. These two issues become more severe as the complexity of the problem increases (i.e. larger action and observation spaces) [78]. The ability to scale using DRL for the portfolio management problem, with raw market data as input from each stock as part of the observation space, is not sufficient for practical implementation, since the problem complexity grows significantly with each added stock in the portfolio. This was directly outlined in a work by Filos, A., where they state using the raw features for each stock does not scale appropriately [79]. There were a couple papers which used portfolios of substantial size, such as the one just mentioned, however these works required a representation learner of some sort to compress the observation space being passed to the agent, used very few input features for each stock, or had little historical data passed to the agent at each step, in order to reduce the problem complexity. From this review of over 20 unique works, it can be concluded that approaching the portfolio management problem using raw input features for each stock (open, high, low, close, volume, etc.) is not feasible at this time.

From the tables above, it is noted that there are a number of papers which use PPO to address the portfolio allocation problem, that said, these fall short in a number of regards, mostly relating to the scaling ability to larger action and observation spaces.

In *Novel Deep Reinforcement Algorithm With Adaptive Sampling Strategy for Continuous Portfolio Optimization*, only 5 stocks are considered, with features consisting of OHLC for the current time step, as well as MA's (5, 20, 30, 60, 120, 900, 1800, 3600, 7200, 10800). Although the number of features is quite large, the number of stocks is considerably limited, and the look-back, or number of previous time steps, provided to the agent is also sparse. Lastly, the frequency of data is at the second level. [72].

Additionally, *Deep reinforcement learning for portfolio management of markets with a dynamic number of assets* also uses a limited action and observation space, with the number of stocks during training being under 25 at most (it is dynamic in this setup), with a total of 6 features for each for only a single time step. Their model is also designed in a fashion where the number of assets doesn't matter, which ends up preventing the model from learning any relationship between the stocks considered. This is the most considerable limitation as knowing the degree of correlation between the assets in a portfolio helps make decisions to reduce unnecessary risk [69].

In *Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation*, 30 stocks are considered, but with only two features as input for the current time step. This is considerably limited even though considering 30 stocks, as the number of features and look-back length are so small [68].

Deep Reinforcement Learning for Stock Portfolio Optimization also uses PPO for the portfolio problem but again is limited, with only 6 stocks considered, three input features, and an unspecified look-back length. This is considerably limited compared to others as the number of stocks and input features are both very small [63].

Lastly, in *Adversarial Deep Reinforcement Learning in Portfolio Management* the authors select only 5 stocks, at random, include 2 input features for each stock, and don't specify the look-back length [58].

As outlined, the existing papers which use PPO for the portfolio management

problem are limited in the sizing of their action and observation spaces. Although not explicitly discussed in every paper, it is believed not to be coincidence that at least one aspect of the spaces (number of stocks, input features, or look-back) for each paper are limited, as it is known that DRL is sample inefficient. With so many possible combinations of action and observation space values, it becomes hard for the agent to converge on a meaningful policy.

Both sections of the review illustrate a number of things, however we try to focus on discussing major gaps that exist in the literature. To start, we bring attention to the model types selected for each of the works above - it can be clearly seen that although used for the portfolio problem in a handful of papers, this review finds that the PPO architecture has not been applied the time series forecasting problem alone, in a single stock fashion [77, 58, 63, 69, 72]. Applying such would definitely fill a gap in the literature, and thus as a result, it becomes the main architecture of consideration for this work. Not only this, but it can be clearly seen that the stock selection methods used for both problem formulations are simplistic at best. Many papers select their securities at random, which is not a well defined and empirically validated approach by any means. It is concluded from this review, that for both the time series forecasting and portfolio management problems, that the most sophisticated and previously validated approach for stock selection is that of using an index, such as the S&P500. Indexes do in fact have criteria for inclusion, and thus technically do count as a type of stock selection method, that said, many others exist and could be explored to test the robustness of this approach. As a result, we also propose testing our DRL system on data obtained using a validated stock selection method, to see if adding the DRL agent to this can lead to an overall increase in performance. It is believed that a B&H strategy employed on stocks selected using a previously validated approach, is a much more practical baseline than a B&H strategy employed on randomly selected stocks.

One unique finding from this review, is that very little reporting was done on the model hyperparameter configurations. Considering there are so many potential combinations, and model tuning is probably the least well defined process in any experiment of this nature, this is considered pertinent information. This is especially true in the case where experiment code isn't provided; which was the situation for almost all the works reviewed. As a result, we conclude that since the PPO architecture has not been tested on the time series problem yet in a single stock manner, that it would also be worthwhile to run experiments investigating a number of different experiment parameter and model hyperparameter combinations. As mentioned in the time series forecasting review, these specifically regard: the learning rate and discount factor for the model itself, look-back length for the input or observation state data, and different sizes of action space, representing position sizing. Additionally, we find that some works employ technical indicators as part of their state or observation space, and so we also decide to investigate the effect of adding common technical indicators as well.

Many of the papers reviewed had a number of different innovations or formulations regarding the reward function definition, order of training, environment specifications, action space type (continuous vs. discrete), as well as others. For these aspects, there is very little consistency between papers, and because each paper is using a different dataset, specific problem formulation, and/or model architecture, it can not be determined which is truly the best. This is a recurring difficulty in disciplines that involve modelling that don't have a standard benchmark dataset and/or problem formulation. Since the model architecture used for this work has not been tested yet, we select the implementation for these elements based on simplicity and what intuitively makes sense regarding the investment problem. These selections will be discussed further in Chapter 4.

Chapter 3

Datasets, Model Details, and Assumptions

3.1 Dataset

This work uses two different sets of data, the first is for experiments concerning the discount factor, learning rate, action space, look-back length, and input features. The second set is used for experiments concerning the portability of the model, protocol, and parameters under question, utilizing the optimal set-up determined from the experiments done with the first dataset.

This first dataset is fairly simplistic and specifically consists of the daily OHLCV SPY market data between 1993/1/29-2021/07/30. The period between 1993/1/29-2013/01/09 is used for training, 2013/01/10-2015/11/13 for validation, and 2015/11/16-2021/07/30 for testing.

The second dataset consists of multiple tables, one of which is created from a couple of different data sources. The first is a custom data table which consists of over 83k securities from a number of exchanges (NYSE Arca, Nasdaq Global Select, New York Stock Exchange, NASDAQ Global Market, NASDAQ Capital Market, NYSE American, Nasdaq, NYSE, NasdaqGM, NasdaqCM, NasdaqGS, Paris, Toronto, NYSEArca), for each of their earnings periods, including a number of different variables, such as their calculated Piotroski score. This set, created by combining multiple

different types of data frames, is used to determine the securities chosen for the Piotroski portfolio experiments. The other data tables are the daily OHLCV data for each of the securities chosen. Since many securities had the highest Piotroski score possible, we needed to limit the scope of securities selected even further. In order to do this stocks with a beta value between 1-1.5 were selected to create a smaller list of candidates, consisting of 6 stocks: AGRO, AIN, ASTE, INT, LH, PPBI. These tables are used for training, validation, and testing each stock's model, and is split in the same proportions as the first dataset consisting of the SPY daily market data. The datasets used were obtained from Tiingo and Financial Modelling Prep, the former for market data and the latter for fundamental data needed to calculate the Piotroski scores. After considering and investigating a number of different data sources, these were decided upon due to their quality, breadth, and cost. Both services provide API access to a very large amount of data, which has a low level of missingness, each for under $20 USD per month. Other data sources considered, but not used for a number of different reasons are: Alpha Vantage, Wharton Research Data Services, Yahoo Finance, Bloomberg, and Trading View.

3.2 Algorithm Description

Although different implementations of PPO exist, the core purpose of this algorithm was to attempt to answer the problem of what is the biggest possible **gradient** step that can be taken, given the available data. Although other unique implementations aim to solve this problem, such as TRPO, PPO is significantly less complex to implement and performs just as well [51].

As mentioned, PPO is an on-policy actor-critic method, however there are different ways of implementing PPO. One such variation that is seen is PPO-penalty vs PPO-clip. PPO-penalty uses a KL-constrained update but penalizes the KL-divergence

term in the objective function, whereas PPO-clip doesn't have a KL-divergence term in the objective function and instead has a clipping term which is to ensure the new policy doesn't get too far from the old one. In this work, we use the clipping version of PPO as is implemented in the Stable Baselines 3 (SB3) library. The pseudocode for PPO can be seen in the algorithm below, where in this work the advantage estimation is normalized [80]. This specific SB3 implementation is used since there are a great deal of variations when it comes to PPO implementation. SB3 looks to try and remedy this issue by providing a standard and stable implementation for people to work off of, allowing for more easy comparison between different works [81].

Algorithm 1: PPO-Clip Pseudocode [80]

Input: initial policy parameters θ_0, initial value function parameters ϕ_0 ;

for $k = 0, 1, 2, ...$ **do**

Collect set of trajectories $D_k = \tau_i$, by running policy $\pi_k = \pi(\theta_k)$ in the environment;

Compute rewards-to-go $\hat{R}_t$;

Compute normalized advantage estimates, $\hat{A}_t$ based on the current value function V_{ϕ_k};

Update the policy by maximizing the PPO-Clip objective function with Adam;

Fit value function by regression on mean-squared error;

end for

The SB3 implementation has two policy types to select from which are convolutional neural network (CNN) and multilayer perceptron (MLP), with the former being good for 3D tensor image data as observations. No matter the policy type, the optimizer used for each by default is Adam, which is very standard for DL work. The pseudo-code for the Adam optimizer can be found in Algorithm 2.

Algorithm 2: Adam Algorithm [82]

Require: α: Stepsize;

Require: $\beta_1, \beta_2 \in (0, 1)$: Exponential decay rates for the moment estimates;

Require: $f(\theta)$: Stochastic objective function with parameters θ;

Require: θ_0: Initial parameter vector;

$m_0 \leftarrow 0$ (Initialize 1^{st} moment vector);

$v_0 \leftarrow 0$ (Initialize 2^{nd} moment vector);

$t \leftarrow 0$ (Initialize timestep);

while θ_t *not converged* **do**

$t \leftarrow t + 1$;

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t);

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate) ;

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate);

$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate) ;

$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate) ;

$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters) ;

end while ;

return θ_t (Resulting parameters)

3.3 Assumptions

The largest practical assumption made for this work is that our agent's actions do not interfere with the data seen in the next observation. This is a considerable assumption since in a real world scenario any order affects the price of a stock. In order to overcome this, it would require a live trading simulation, which is not entirely feasible. Thus, as a result, a relatively small amount of starting capital is used ($100,000 USD), since it would likely not move the price of any stock used in this work significantly.

The other assumptions made for this work pertain to specific model implementation details with PPO. We assume that using an Adam optimizer for the neural networks along with the MLP DL architecture implemented in SB3 is sufficient and doesn't need to be explored further.

Chapter 4

Evaluation and Result Analysis

The following chapter proceeds to discuss the exact experimental set-up used for this work and why, along with the evaluation metrics used. This chapter also reviews all the experimental results, and then compares them to a B&H strategy on all evaluation metrics employed.

4.1 Experiment Setup

Part of the contribution of this work is the custom environment created, which is a subclass of OpenAI's gym environment class. The custom environment was built to be compatible with SB3, allowing for the use of multiple DRL types with only minor modifications to the code. Although work has been performed in the past to create custom environments, many of those reviewed did not easily allow for the experiments employed in this work, nor were well maintained. The custom environment in this work employs multiple unique features, such as being able to specify look-back length, additional feature columns other than OHLCV, log folders, and the start and end of the data frame to be used for the environment, as well as employs data normalization, evaluation metrics, training metrics, action space sizing, transaction costs, and custom tensorboard functionality for easy model performance monitoring.

The training metrics used for the tensorboard logs, other than the standard already included with the SB3 library were episode maximum networth, minimum networth, and profit, as well as the networth at every log step. These metrics allow for more insightful monitoring during training rather than only using entropy, value, and policy gradient loss as well as explained variance and approximate kl. The thinking behind implementing these additional metrics was to ensure that the policy being converged upon was in fact a meaningful one in relation to the task at hand, maximizing profit during trading.

Data normalization is well known to reduce training times and increase performance in the supervised deep learning literature, and since a component of our model employs a deep neural network it was thought best to normalize the data used in the environment [83]. The normalization undergone was standard mean zero normalization for all columns in the data frame, using the entire data frame length specified. In addition to this, a non-formalized normalization of the account data (balance, shares held, and cost basis) was also performed, using max account balance, maximum number of shares, and maximum share price variables. The max account balance can be seen as an environment hyperparameter, and is set to 5 Million USD for our experiments. The maximum number of shares is calculated by dividing the maximum account balance by the minimum share price in the data frame. Finally, the cost basis is normalized using the data frames maximum share price. Although the max account balance chosen is somewhat arbitrary, there is no mention or specification of this variable in the other works. We chose this value as it would require approximately a 20% annual return throughout the entire training period, which is considered extremely good, especially considering the first dataset is a non-volatile ETF.

An important component on any reinforcement learning set up, or specifically environment creation, is deciding upon the definition of the reward function. The reward function is the main guiding principle to any reinforcement learning loop, as

it is technically its objective function, or in other words what is being optimized for. Three common reward functions are found in the literature reviewed in this work, these are the Sharpe and Sortino ratios as well as profit, with slight modifications. Since the reviewed literature all use different datasets, features, action space design, and more, no clear optimal reward function could be determined. That said, one paper did review all three reward functions as part of their work, and depending on the dataset being tested on, using profit and Sortino ratio performed the best, with profit performing slightly better [12]. As a result of this, we proceed with the profit function, defined as the change in the total net worth from one timestep to the next, as can be seen below:

$$R_t(S, A) = (NW_t - NW_{t-1})/NW_{t-1}$$

R_t is the reward, A,S is the state-action pair, t is the current time step, and NW is the net worth, which is a sum of both the current cash balance, as well as present value of shares held. This is done since only optimizing for cash balance or present value of shares held will result in the agent always converging on a policy that either only buys or sells.

For any deep reinforcement learning problem a tremendous number of hyperparameters exist which can often take on an infinite number of values, in example, some of these for the PPO algorithm are discount factor (DF), learning rate (LR), entropy and value coefficient, number of steps for each environment, batch size, number of epochs, policy type, general advantage estimator (GAE) value, whether or not to use generalized state dependent exploration or normalize the advantage, and many more. Not only this, but many potential experiment parameters exist for the exact problem being focused on in this work; such as how many past days of data to provide the agent at each step, which features to use, weather the reward function should consider the change in only one time step or multiple, what the max account balance

should be (as described earlier), which account data to provide to the agent, and many more. As a result of this, the focus of this work is on what is believed to be the most impactful, as well as what is missing from the literature. For model hyperparameters, this work focuses on the learning rate and discount factor, whereas for the experiment parameters the interest is the impact of how many past days of data is provided to the agent, different action space sizes and configurations, as well as the impact of adding common technical indicator features derived from the OHLCV data. Since this small set of parameters, even when only considering a handful of potential values for each, could generate 1000's of experiments of significant duration, this work focuses on what is hypothesized to be the most impactful parameters first, and continues downward in perceived level of importance. The hyperparameter values used were either within the range deemed to be optimal in PPO RL Baselines Zoo experiments, or were chosen based on either what has been previously done with other DRL types or were missing from the literature [84, 12, 13]. The values tested for each parameter can be seen in Table 4.1 below:

As for other hyperparameters and their values, anything not mentioned in the table above, except for the number of time steps and batch-size (34), were set to their default values as specified by StableBaselines3. The number of time steps chosen was determined by what has been used in other works as well as guided by training metrics such as entropy, value, and policy loss in addition to explained variance [77].

To start the sequence of experiments, the look-back back period is changed while keeping all other parameters constant. From here, the best look-back length is used for the next set of experiments and kept constant while exploring the next parameter, which is the effect of adding technical indicator features. This cycle continues in the following order: action space, discount factor, and learning rate, as well as can be seen represented in Figure 4.1 below:

RL is known to be notoriously unstable, and although using PPO should help

Parameter	**Values**
Look-Back	10, 20, 40, 80, 120, 160, 240
Technical Indicators	MA, BBs, Adj. OHLCV, MACD, RSI, OBV
Action Space	Shares: 1, 5, 9, 4*5, 4*10, 4*15 % Available Balance: (0, 25, 75, 100)
Discount Factor	0.99, 0.97, 0.95, 0.9, 0.85, 0.8, 0.75
Learning Rate	0.0001, 0.0002, 0.0005, 0.00001, 0.00002, 0.00005

Table 4.1: Experiment Parameter and Model Hyperparameter Values

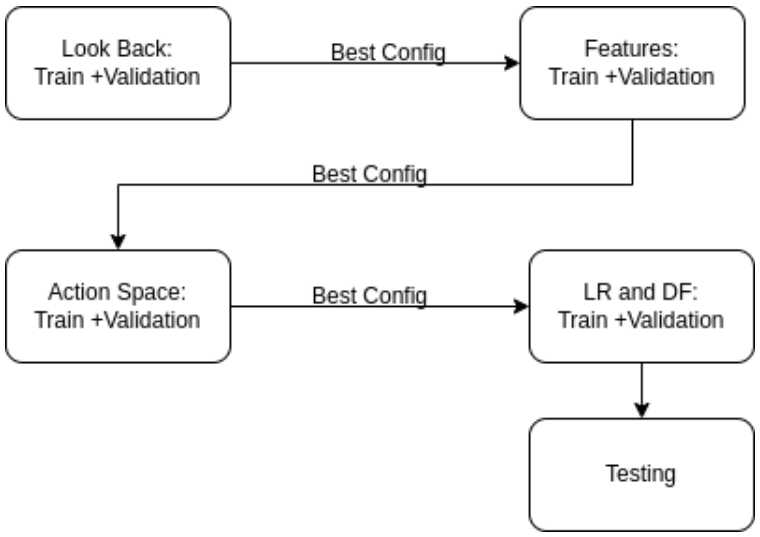

Figure 4.1: Experiment Sequence

combat that effort, it is still a prominent issue when comparing to other types of machine learning algorithms [85]. As a result of this, during training, we don't want to just save the final model but rather the ones which perform the best. This is accomplished by collecting the mean reward for the current episode at every log step, if it exceeds the best mean reward collected so far, the model is saved. This allows us to save multiple potentially useful models as well as calculate an average of the top performers for each experiment configuration.

All experiments were performed using Python 3.6.13, and utilized the following noteworthy libraries: StableBaselines3, Tensorflow, PyTorch, Numpy, Pandas, Matplotlib, Gym, and btalib. All the environment details and requirements other than those mentioned are included in a requirements.txt file which can be used directly within Anaconda in order to install the necessary libraries. Aside from software specifications, all experiments are performed using an Intel 10th Gen I5-10600K, Nvidia GeoForce 3060 with 12GB of GDDR6 RAM, and 16GB of CPU RAM.

4.2 Evaluation Metrics

Evaluation is one of the most important aspects of any modelling experiment. Generally within the ML community there is a great deal of focus on the predictive accuracy and precision of the model, that said, the RL set-up does not necessarily lend well to this unless the reward is specified in way that specifies this and the agent is told to focus only on the immediate reward, or in other words completely ignores taking advantage of the sequential and forward planning of reinforcement learning. As a result, RL usually formulates more practical evaluation metrics related to the task it is being trained to perform. In the realm of finance and investment management the performance metrics are fairly clear, the goal is to maximize profit as fast as possible while minimizing draw down of the total net worth. For the goal of maximizing profit,

we look to the evaluation metrics of total profit and annualized return. In regards to minimizing losses, we use maximum draw down, which is defined as the difference in net worth from the highest peak to the lowest trough, given the peak is before the trough. Another set of metrics tracked, relating to the amount and type of trading, was the number of buy, sell, and hold action, as well as total trades, which is the sum of all buy and sell actions.

Every set of experiments uses all evaluation metrics and is compared to a buy and hold strategy. A buy and hold strategy is considered a good comparison since it is easily employed, practical, and also a historically successful strategy [86]. The exact implementation of this is as follows. For the single stock under consideration, purchase the maximum number of shares the starting balance will allow for, and hold these shares to the end of the evaluation period. At each time step, use the price of the shares at that time to calculate the unrealized return, as well as the other evaluation metrics mentioned earlier, to allow for a direct comparison between this and the model performance at any point, including the last step of the the evaluation period.

4.3 Experiment Results and Analysis

This section begins with the results from the look back experimentation. In Figures 4.2, 4.3, and 4.4, we have results from the training itself. We monitor the classical training metrics such as entropy, value, and policy gradient loss, as well as explained variance. Not only this, but we also implement custom evaluation metrics which we believe to be more practical, which are the episode profit, maximum draw down, highest net worth, and lowest net worth. One episode is defined as either the agent losing all of its capital, or reaching the end of the data frame, at which point the data frame is reset, along with all other environment variables. Since many experiments

were run, and there are 8 figures per experiment, there are 100's of figures, not only this but the metrics are only used to guide training and are thus far less valuable than validation and testing figures. As a result, the experiment files include the tensorboard logs for brevity, but only one set of figures for one training experiment is presented within this work. Figures 4.2, 4.3, and 4.4 below are representative of what is also witnessed in the other training experiments.

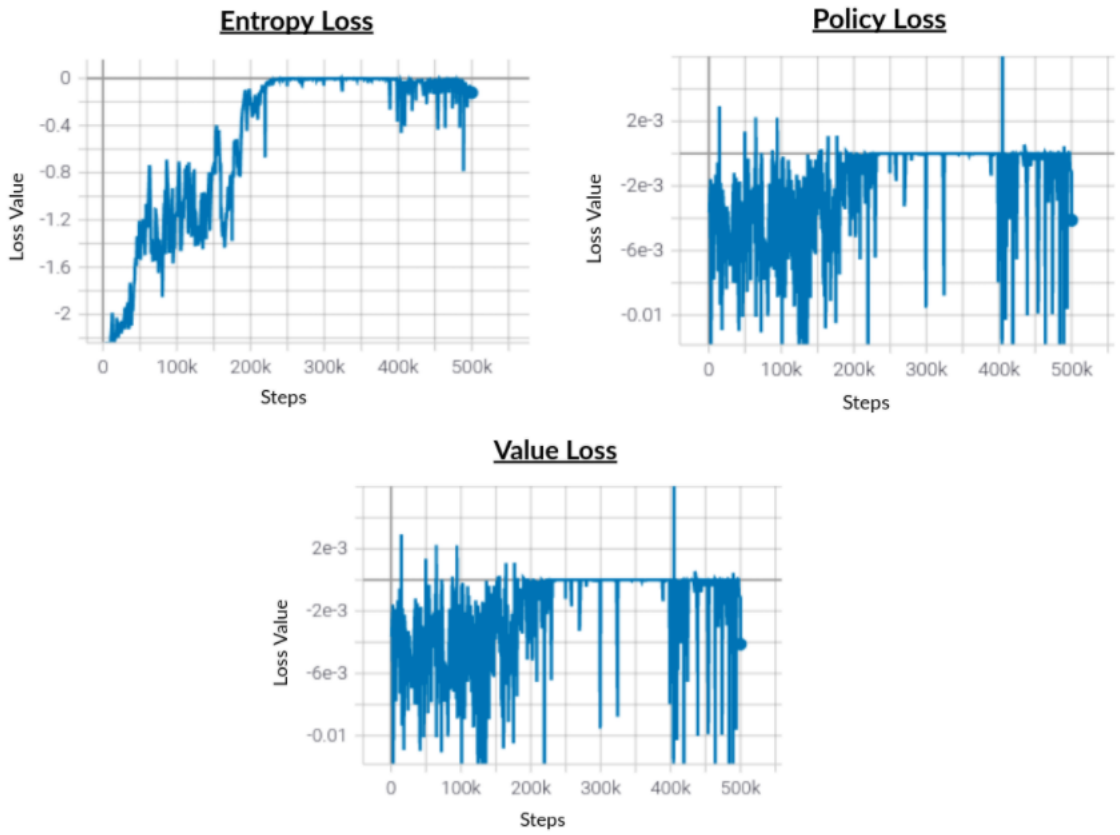

Figure 4.2: SPY Training Loss Curves

As can be seen in Figure 4.2, the values for all three graphs are quite chaotic, with maybe the exception of the entropy loss. Not only are they chaotic, they are unstable even when reaching optimal value ranges. This highlights the need to regularly monitor performance and save model configurations. Regardless of these two points, the graphs do show that at seemingly the same point in time they reach optimal value ranges. This indicates that at least one of the saved models for this run should be from this time point and have optimal parameters given the policy at that moment.

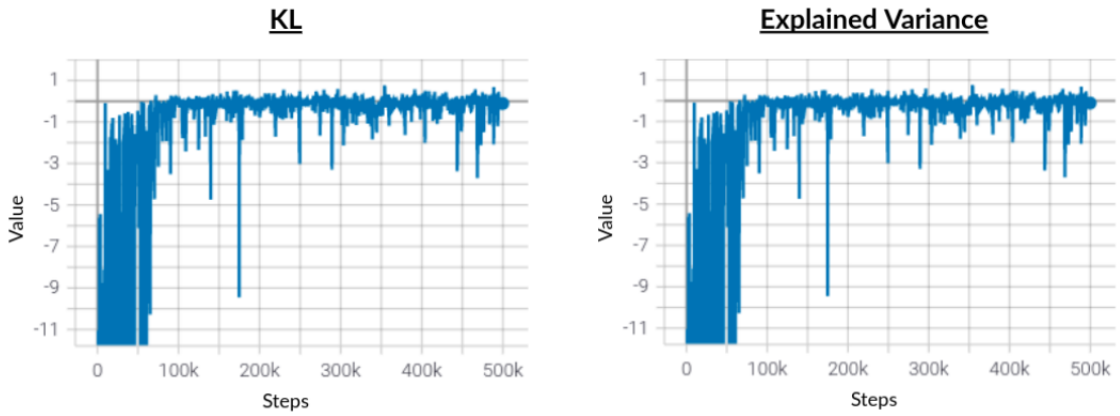

Figure 4.3: SPY Training KL and Explained Variance

Figure 4.3 above shows that for the KL values as well as the explained variance, both approach the optimal value range of [-1, 1] but seem to plateau very quickly thereafter.

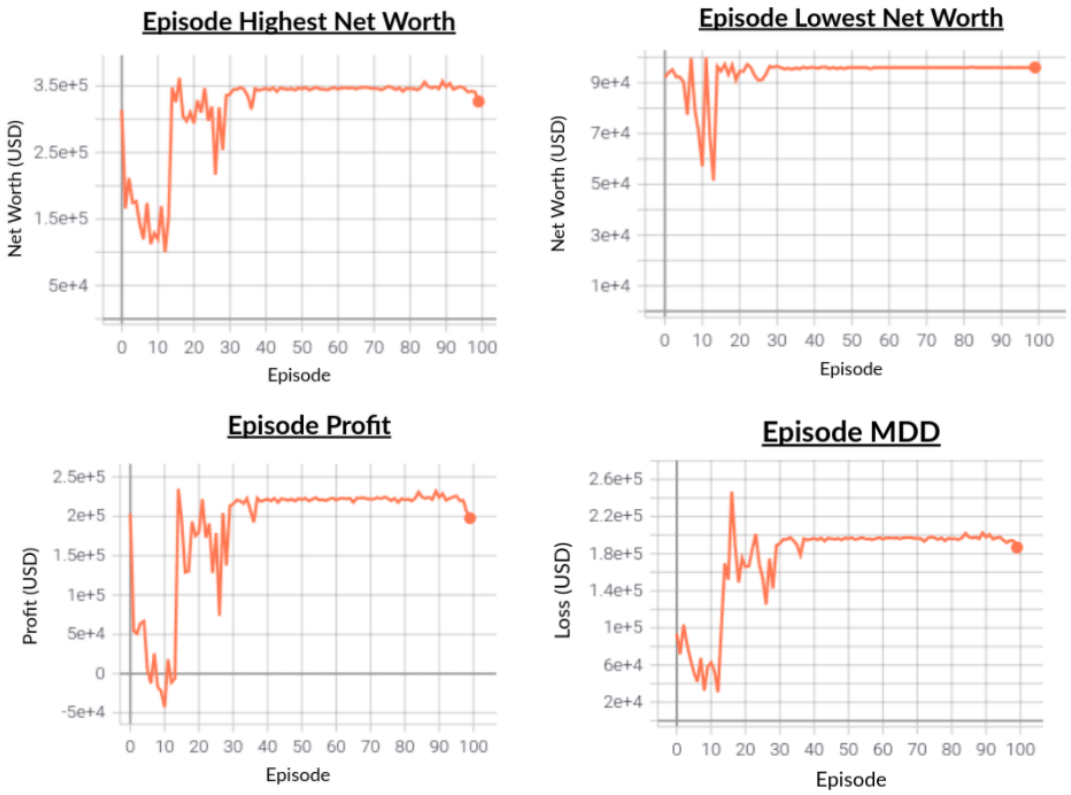

Figure 4.4: SPY Training Practical Evaluation Metrics

To finish for training, we draw our attention to the graphs in Figure 4.4, where we see all 4 graphs to be highly correlated with one another. This intuitively makes sense, as one would think the highest and lowest net worth, as well as profit, would be correlated in any trading simulation. This should be especially true in a situation where the optimization is for a specific reward function, such as profit. That said, the lowest net worth and MDD graphs also show the same increase as the other two, indicating the agent is having trouble minimizing its losses. In addition to this, these graphs also align well with the graphs in Figure 4.3. This also makes sense since the greater the degree of explained variance, and the smaller the difference between the two data distributions for the KL term, the greater ability to predict. All in all, the training metrics look promising, and provide credence to progressing to the validation experiments.

Next, we present the validation results for all 5 sets of experiments, of which all parameters are kept constant except for those mentioned in the tables themselves. For all tables below, the values presented are the mean result of the top three performing models during validation.

LB	**AR**	**Profit**	**MDD**	**Hold**	**Sell**	**Buy**	**Total Trades**
BH	*0.122*	*38937.99*	*17507.48*	*0*	*0*	*1*	*1*
10	**0.115**	**36235.49**	**17166.67**	**52**	**0**	**656**	**656**
20	0.111	34815	17034	321	1	376	377
40	0.100	31065	16514	0	0	678	678
80	0.085	26022	15836	0	1	637	638
120	0.086	26500	15939	22	0	576	576
160	0.061	19004	7453	114	212	232	444
240	0.040	11719	14077	0	0	478	478

Table 4.2: SPY Validation Experiments Testing the Effect of Look-Back Length

As per our experimental sequence discussed earlier, we begin experimentation by testing the effect of look-back length. As can be clearly seen by the bolded row in Table4.2, the best performing model out of those tested utilizes a look-back period of 10-days. This is interesting as one would intuitively expect that the greater the history the better, since many technical analysis traders often employ indicators which summarizes data over a much longer period than 10-days, in example, a 200-day moving average. Rather than this being true however, it is very clear that adding more days of history, given our experimental set-up, did not help increase performance. Not

only this, but it can also be seen that the number of selling actions is zero and the number of holding actions is 52, with the rest being buy actions for our best model. This could indicate that the agent has converged upon a policy which only buys, but at more optimal times, potentially taking advantage of a cost-averaging type of strategy. Lastly, before moving to the technical indicator experiments, it is acknowledged that the MDD for our best model exceeds that of the others, which falls in line which what was observed in the training figures and our hypothesis that the agent is trading off MDD for larger gains in net worth.

LB	**AR**	**Profit**	**MDD**	**Hold**	**Sell**	**Buy**	**Total Trades**
BH	*0.122*	*38937.99*	*17507.48*	*0*	*0*	*1*	*1*
10	**0.114**	**35964.91**	**17132.33**	**15**	**0**	**693**	**693**
20	0.108	33922	16875	0	0	698	698
40	0.099	30995	16506	24	0	654	654
80	0.035	10724	7847	7	396	235	631
120	0.087	26701	15965	0	0	598	598

Table 4.3: SPY Validation Experiments Testing the Effect of Look-Back Length and Adding Technical Indicators

For the technical indicator experiments, as seen above from Table 4.3, not only was the best performing 10-day look-back period used, but also 20, 40, 80, and 120. The reasoning for this was that it was believed that larger look-backs may have not produced desirable results in the last set of experiments due to a lack of meaningful features. As a result, in this experiment we add all the technical indicators presented earlier, and explore different look-backs with these present. As was in the

first set of experiments, we see that adding a larger look-back does not help with performance, even when adding technical indicators. In fact, the performance drops slightly compared to that of the look-back experiments without technical indicators, although by an insignificant amount. In addition to this we can clearly see the best performing configuration may be employing a similar type of cost-averaging strategy, as was suggested for the look-back experiments without technical indicators. Since the difference in performance, for the best configuration when comparing this experiment and the last, is so insignificant, we proceed to the next set of experiments utilizing technical indicator features in the case that having them present while changing other parameters may lead to better performance than without, as is in line with proponents of technical analysis highly valuing these features.

AS	LB	AR	Profit	MDD	Hold	Sell	Buy	Total Trades
BH	*0*	*0.122*	*38938*	*17507*	*0*	*0*	*1*	1
1	10	0.015	4447	8961	0	236	472	708
4	**10**	**0.114**	**35965**	**17132**	**15**	**0**	**693**	**693**
9	10	0.000	0	0	0	708	0	708
4*5	10	0.058	17510	14806	0	0	708	708
4*10	10	0.075	22817	15476	1	0	707	707
4*15	10	0.000	0	0	1	708	0	708
1	40	0.006	1680	4412	1	452	225	677
4	40	0.063	18936	14987	0	0	678	678
9	40	0.017	5017	4856	121	458	99	557
4*5	40	0.088	27109	16017	0	0	678	678
4*10	40	0.098	30445	16437	0	0	678	678
4*15	40	0.100	31194	16532	0	0	678	678

Table 4.4: SPY Validation Experiments Testing the Effect of Look-Back, Technical Indicator, and Action Space

Lastly for experiment parameters, the action space is modified to see the effect of different configurations. In this table, where a "*" is present in the action space, share scaling is being utilized rather than percentage of available balance. For example, 4*5 means that for sizing of each of the actions (Buy and Sell), there are 5 options(0-4), 0, 5, 10, 15 and 20 shares, where a buy or sell action with 0 shares purchased or sold is counted as a hold action in the tables. Alternatively, where a single number is present we use percentage of the total possible shares purchasable, given the current available balance. In the example where AS=4, the agent has the options of buying 0%, 25%, 50%, 75%, or 100%. As with the previous experiments we also look at more than one look-back length for sanity, of which again shows that increasing look-back length past 10-days is not useful. Interestingly enough, the AS configuration used for the previous set of experiments, AS=4, ended up being the best for this set as well. Attention is now drawn to the model hyperparameter tuning, where the effect of discount factor and learning rate values are investigated.

DF	LB	AR	Profit	MDD	Hold	Sell	Buy	Total Trades
BH	*0*	*0.122*	*38938*	*17507*	*0*	*0*	*1*	1
0.75	10	0.114	35899	17124	6	0	702	702
0.8	10	0.115	36301	17175	0	0	708	708
0.85	10	0.117	36912	172434	22	4	682	686
0.9	10	0.055	17416	11709	614	9	85	94
0.95	**10**	**0.120**	**38196**	**16713**	**0**	**17**	**691**	**708**
0.97	10	0.102	31977	16615	3	19	686	705
0.99	10	0.114	35965	17132	9	0	699	699
0.8	40	0.101	31566	16286	3	2	673	675
0.85	40	0.099	30995	16506	41	0	637	637
0.9	40	0.067	20938	11038	0	226	452	678
0.95	40	0.083	25655	12502	76	184	418	602
0.99	40	0.099	30995	16506	24	0	654	654

Table 4.5: SPY Validation Experiments Testing the Effect of Discount Factor

The discount factor is the first focus since this is believed to be more impactful than the learning rate. This is because the learning rate really only should effect the step size taken during each update, of which is unlikely to effect the policy converged upon, but rather the time to convergence. Looking at Table 4.5, it can be seen that the best metric used for the discount factor was in fact 0.95, with 0.99, the metric used

for all previous experiments, coming in close second. An interesting finding with this table is that the number of hold actions has dropped to zero and the selling actions have increased, potentially indicating that there are some, although seemingly few, optimal times to sell shares according to the policy. Before moving on to the testing results for this dataset, we present the learning rate experiment results.

LR	AR	Profit	MDD	Hold	Sell	Buy	Total Trades
B&H	*0.122*	*38938*	*17507*	*0*	*0*	*1*	*1*
0.0001	**0.120**	**38196**	**16713**	**0**	**17**	**691**	**708**
0.0002	0.111	35085	17203	0	2	705	708
0.0005	0.108	33868	17328	0	7	701	708
1E-05	0.113	35762	17107	0	0	708	708
2E-05	0.113	35830	17115	0	0	708	708
5E-05	0.082	25602	14971	255	0	453	453

Table 4.6: SPY Validation Experiments Testing the Effect of Learning Rate

For the learning rate set of experiments above in Table 4.6 there are very few surprises. These experiments confirm that our intuition; that the learning rate, especially when tested last, will simply increase the amount of time or steps necessary before achieving zero loss, but will still converge upon the same level of performance during validation. Nonetheless, a learning rate of 0.0001, as can be seen from the table, results in the top performance compared to other values used. As a result of all the above experiments, it is decided to move forward with the best model for testing, which utilizes a look-back length of 10-days, added technical indicators, action space

size of 4, discount factor equal to 0.95, and a learning rate of 0.0001.

Lastly, before moving onto the second dataset, the final figure and table for the SPY dataset is presented. It should be noted, that only one true test is performed for each dataset, each utilizing the best model of all those tested for that dataset. We compare to a B&H strategy, to see if our method is worthwhile to employ. As mentioned earlier, a B&H strategy is very practical, common, historically well performing, and requires potentially the lowest level of skill necessary to implement successfully, it is therefore considered a good lower lever practical baseline.

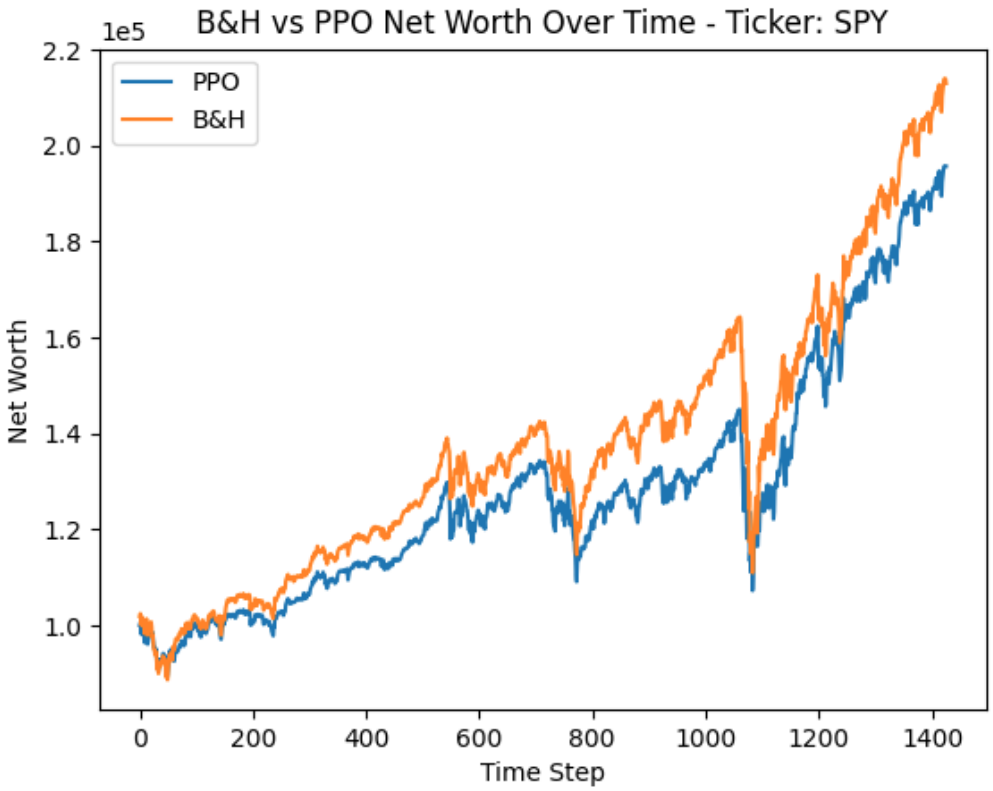

Figure 4.5: SPY Testing Experiment Comparing the Best PPO Model from Validation to a B&H Strategy

	AR	Profit	MDD	Hold	Sell	Buy	Total Trades
BH	*0.142*	*112971*	*53302*	*1425*	*0*	*1*	*1426*
Model	0.125	95703	37692	0	119	1307	1426

Table 4.7: SPY Testing Experiment Comparing the Best PPO Model from Validation to a B&H Strategy Using the SPY Test Set

Figure 4.5 and Table 4.7 above show model testing performance on the SPY test set. As can be seen from Figure 4.5, the net worth of the model under performs the B&H strategy at almost all points in time. Looking at Table 4.7 regarding this test confirms the same, with the AR and profit being lower for the model. Although the MDD is also lower, when evaluated in conjunction with Figure 4.5, it can be seen that this is inconsequential, and only a result of a lower total net worth, and not the strategy itself. Although the number of sell actions are greater than 0 for our model, which could be indicative of a strategy identifying optimal selling times, we can fairly clearly see from the testing figure that this is not the case.

For the second dataset, the first focus is evaluating the portability of the best SPY model from the earlier validation and testing experiments. Below is Figure 4.6 comparing our best PPO model to a BH strategy on all Piotroski testing sets. In addition to this, a summary table containing all evaluation metrics for both the model as well as the B&H strategy for each dataset is presented in Table 4.8.

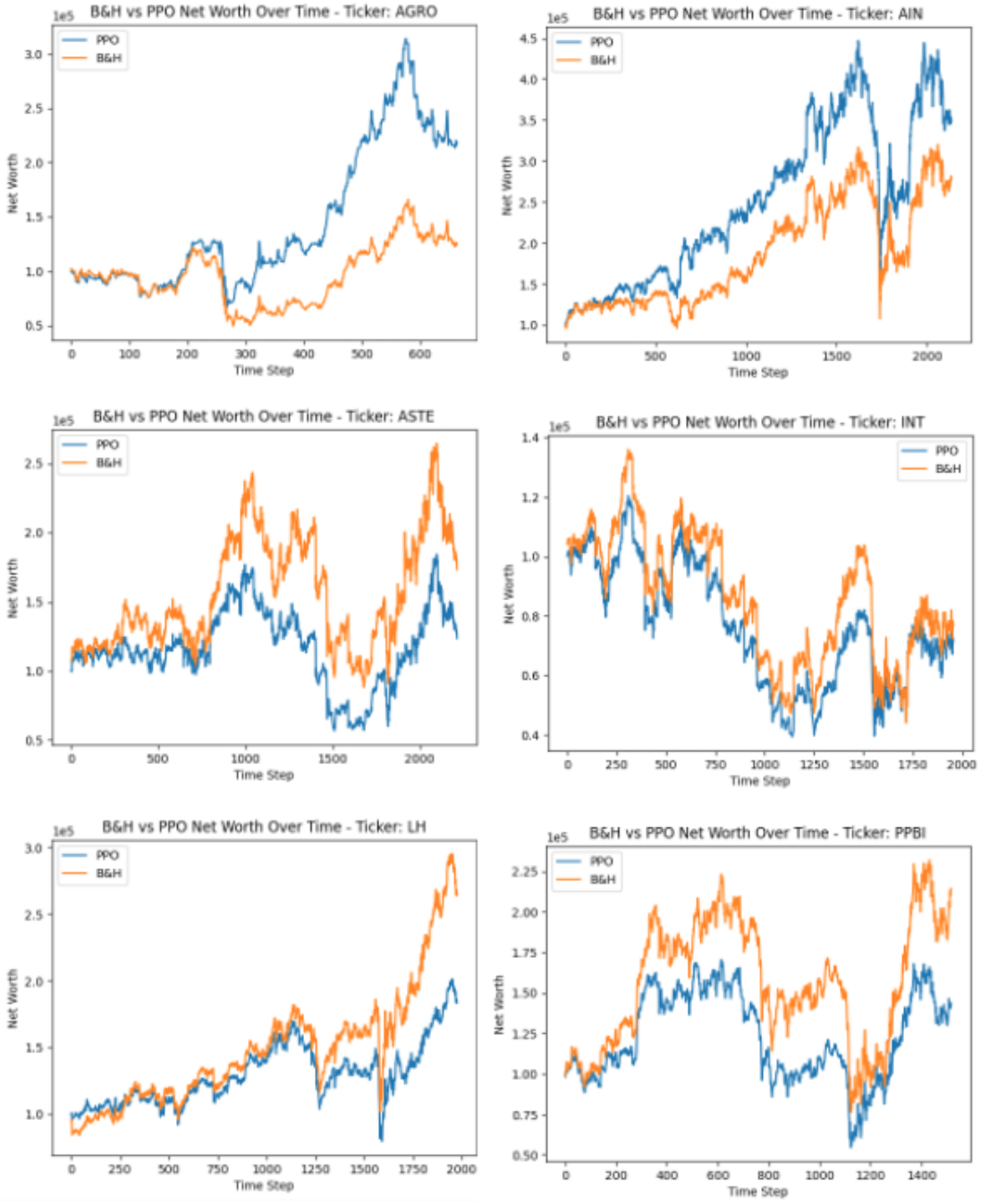

Figure 4.6: SPY Testing Experiment Comparing the Best PPO Model from Validation to a B&H Strategy Using the Piotroski Test Set

	Model							B&H		
Ticker	AR	Profit	MDD	Hold	Sell	Buy	TT	AR	Profit	MDD
AGRO	0.340	118578	100379	0	97	567	664	0.088	25388	72355
AIN	0.158	248131	306013	0	236	1899	2135	0.127	177616	209217
ASTE	0.024	23573	119983	0	275	1941	2216	0.064	73220	154931
INT	-0.043	-28846	81063	0	210	1746	1956	-0.034	-23408	92009
LH	0.080	83846	90081	0	205	1772	1977	0.131	164609	84052
PPBI	0.061	43594	115736	0	179	1339	1518	0.134	114391	146772
	0.103							**0.085**		

Table 4.8: SPY Testing Experiment Comparing the Best PPO Model from Validation to a B&H Strategy Using the Piotroski Test Set

Since the model under performs a B&H strategy on the SPY testing set, it was believed to be unlikely that the portability of the best model trained on the SPY dataset would perform any good on the selected Piotroski stocks. That said, it can be seen from Figure 4.6, that this is not entirely true. In fact, for two of the 6 graphs in the figure (tickers: AGRO & AIN), the model outperforms a B&H strategy. For the other graphs the model behaves similarly to that seen with the SPY testing data, where the model net worth follows the trend of a B&H strategy', yet still underperforms. With this in mind, When averaging all the annualized returns for each stock, as seen in Figure 4.8, the return outperforms a B&H strategy by almost a couple percent.

With the portability performance of the best SPY model exceeding expectations on the Piotroski stocks, training is performed on each stock's dataset using the best performing model from the original SPY experiments. This is done in order to assess the impact of transfer learning using the Piotroski stocks, of which the results are presented below.

	Model							B&H		
Ticker	AR	Profit	MDD	Hold	Sell	Buy	TT	AR	Profit	MDD
AGRO	-0.037	-9546.87	24906.59	0	511	152	663	-0.161	-37370.24	53633.22
AIN	0.039	38873.30	132707.54	13	103	2018	2121	-0.004	-3723.58	127239.18
ASTE	0.091	115719.53	297328.77	13	1	2201	2202	0.104	140431.87	331399.56
INT	0.034	29327.73	116345.61	121	10	1825	1835	0.037	32688.71	119338.74
LH	0.084	89533.92	46522.09	298	13	1666	1679	0.087	93559.76	51930.22
PPBI	0.279	344941.42	83235.00	127	37	1355	1392	0.283	353685.37	81848.18
	0.082							**0.058**		

Table 4.9: SPY Transfer Learning Validation Experiment Comparing the Best PPO Model from Validation to a B&H Strategy Using the Piotroski Test Set

	Model							B&H		
Ticker	AR	Profit	MDD	Hold	Sell	Buy	TT	AR	Profit	MDD
AGRO	0.113	33218	21895	0	472	192	664	0.088	25388	72355
AIN	0.105	134512	199931	9	73	2053	2126	0.127	177616	209216.91
ASTE	0.055	60343	143422	13	0	2203	2203	0.064	73220	154931
INT	-0.040	-27362	88520	96	7	1853	1860	-0.034	-23408	92009
LH	0.137	174448	87540	287	10	1680	1690	0.131	164609	84052
PPBI	0.118	97062	143291	40	15	1464	1479	0.130	109808	146772
	0.081							**0.085**		

Table 4.10: SPY Transfer Learning Testing Experiment Comparing the Best PPO Model from Validation to a B&H Strategy Using the Piotroski Test Set

Presented above in Table.4.9, Table.4.10, and Fig.4.7 show the validation and testing results of the transfer learning experiments run using the best SPY model. The best SPY model was retrained on the respective Piotroski stocks, with the same configurations from the SPY experiments. As can be seen from the testing results, the performance for the PPO model with transfer learning is worse on average than a B&H strategy. Using an equal weighted average for all stocks, the B&H strategy and PPO models have an AR of about 8.5% and 8.1% respectively. Not only this, but only two stocks in the validation and testing experiments outperformed a B&H, with only one of these stocks, AGRO, consistently outperforming a B&H between both validation and testing. This potentially indicates that this stock has enough signal in its market data to explain variance in prices well enough to allow for the RL agent to develop a generalizable policy which can perform well. In addition to this, we can also see that for both the AGRO and AIN stocks the performance doesn't greatly exceed that of a B&H strategy. This is interesting since this was not the case with the portability experiments without transfer learning, where the difference in performance for those that did outperform a B&H strategy was quite significant. One would think that transfer learning would potentially increase the performance of the model, however clearly in this example it is not the case. This could be for a number of reasons, one of which being that the dynamics of the data for SPY and the Piotroski stocks differ significantly enough that a negative transfer is occurring. Since the transfer learning experiments didn't show promise, attention is directed to the next set of experiments, which entail testing the ideal experiment parameters and model hyperparameters from the previous experiments; AS=4, 10-day look-back, added technical indicators, learning rate of 0.0001, and a discount factor of 0.95. For this, models for each Piotroski stock are trained and subsequently validated, before taking the best model from each dataset's validation experiments, and proceeding onto testing.

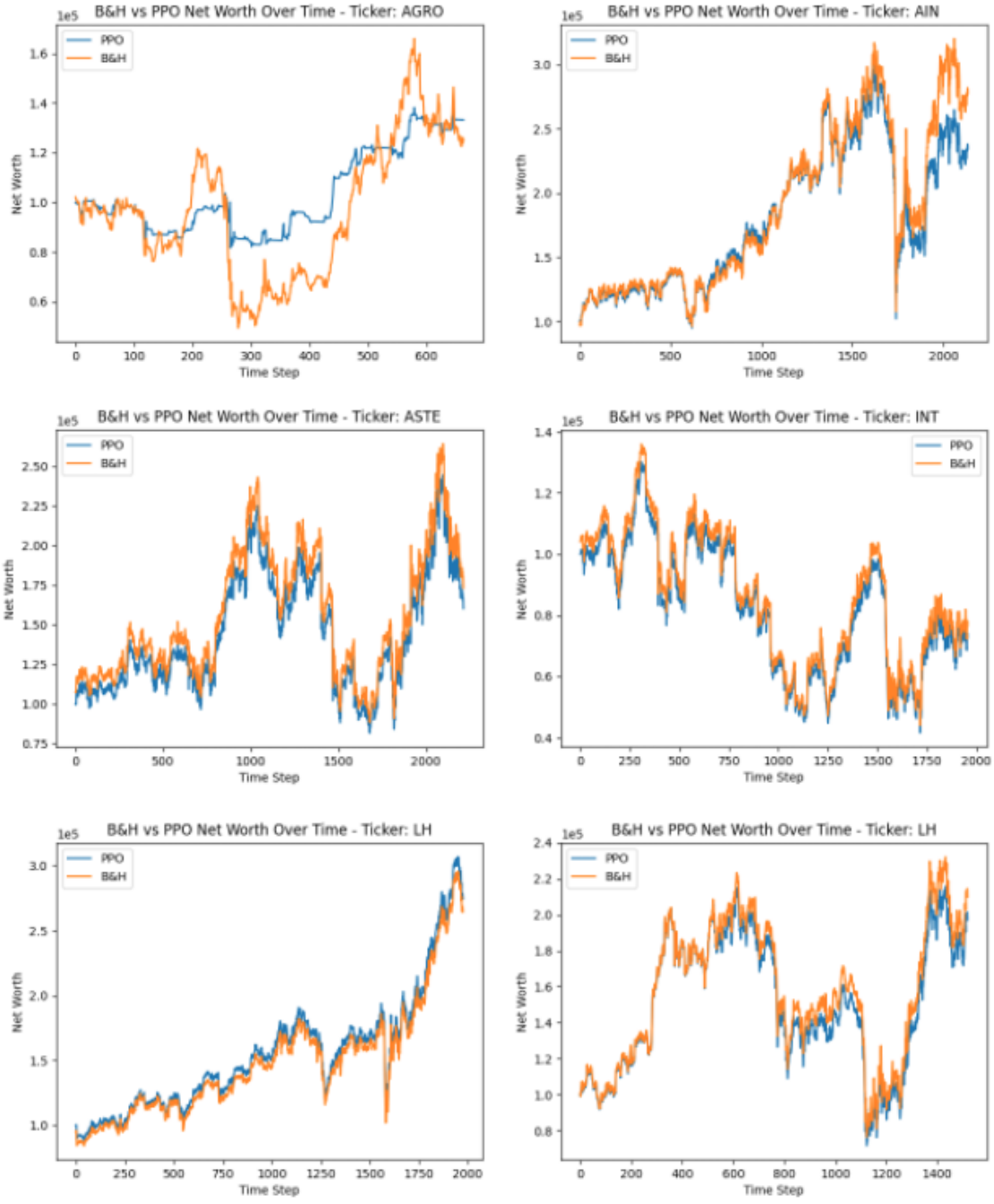

Figure 4.7: SPY Transfer Learning Testing Experiment Comparing the Best PPO Model from Validation to a B&H Strategy Using the Piotroski Test Set

	Model							B&H		
Ticker	AR	Profit	MDD	Hold	Sell	Buy	TT	AR	Profit	MDD
AGRO	0.017	4719	3991	0	656	8	664	-0.161	-37457	53633
AIN	0.075	85305	83881	0	1163	971	2134	-0.004	-3724	127239
ASTE	0.093	118437	301072	0	1	2214	2215	0.104	140432	331400
INT	0.080	82161	156202	13	352	1591	1943	0.037	32689	119339
LH	0.082	86061	50084	7	2	1968	1970	0.087	93560	51930
PPBI	0.302	396430	89559	0	25	1494	1519	0.283	353685	81848
	0.108							**0.058**		

Table 4.11: Piotroski Validation Experiment Comparing the Best PPO Model for Each Stock to a B&H Strategy Using the Piotroski Validation Set

The best models from validation for each ticker are presented in Table 4.11, as can be seen, 4 out of 6 of the best models outperform a B&H strategy during validation. This is promising, especially since for these models the MDD isn't larger than a B&H. Note is kept for which models outperformed a B&H strategy during validation, AGRO, AIN, INT, PPBI, to see if this continues during testing.

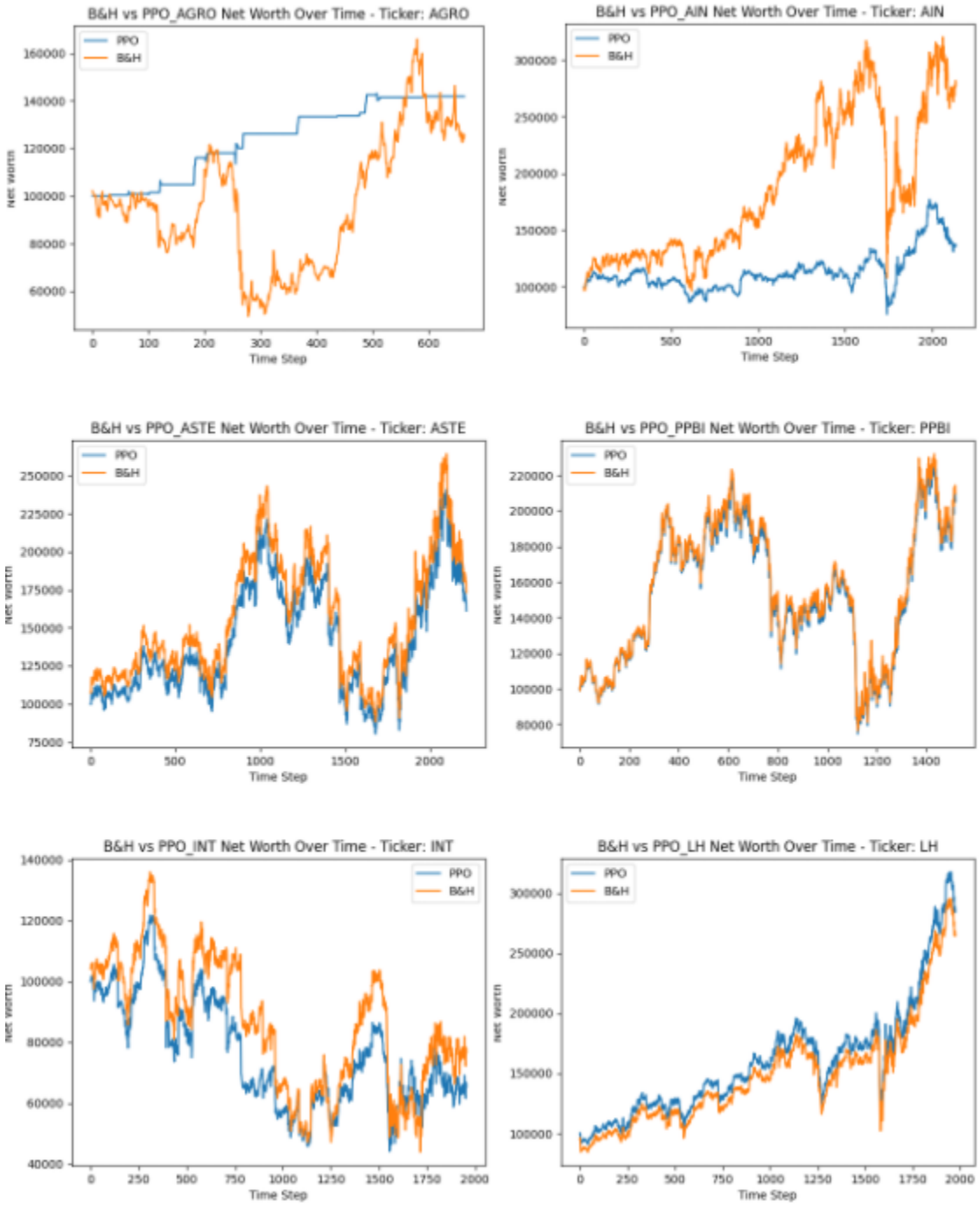

Figure 4.8: Piotroski Testing Experiment Comparing the Networth of the Best PPO Model from Validation to a B&H Strategy Using the Piotroski Test Set

	Model							B&H		
Ticker	AR	Profit	MDD	Hold	Sell	Buy	TT	AR	Profit	MDD
AGRO	0.140	41930	4647	0	648	16	664	0.088	25387	72355
AIN	0.037	36061	58118	0	1136	999	2135	0.127	177616	209216
ASTE	0.056	61187	141058	0	1	2215	2216	0.064	73219	154931
INT	-0.053	-34658	77561	11	350	1595	1945	-0.034	-23407	92009
LH	0.142	184418	90341	2	0	1975	1975	0.131	164608	84051
PPBI	0.126	105154	142776	0	4	1515	1519	0.130	109807	146771
	0.074							**0.085**		

Table 4.12: Piotroski Testing Experiment Comparing the Best PPO Model from Validation to a B&H Strategy Using Piotroski Test Set

As can be seen from Figure 4.8, the graphs unfortunately show similar dynamics to that of the testing done on the SPY dataset, where the net worth mimics the trend of a B&H strategy, but either under or over performs slightly. Only two exceptions can be seen in the top two graphs, AGRO and AIN. The performance on AGRO's testing data is exceptional, with almost zero draw down occurring throughout the whole testing period. Not only this, but net worth increases in an almost step wise fashion and ends up exceeding that of a B&H strategy for the vast majority of the time steps. As for AIN, the testing clearly shows the same dynamics as a B&H strategy, but tremendously under performs such. Turning attention to Table 4.12, we see confirmation of what was concluded from the figures. However, it is also noted that only 1 out of the 4 models which outperformed a B&H strategy during validation, also outperformed the B&H strategy during testing. Of the others, two were very comparable to a B&H with the other performing significantly less well, however still positive in return.

Interestingly enough, regardless of the approach taken for the Piotroski dataset, certain stocks inherently performed better than others. Specifically, AGRO performed quite well in regards to AR and MDD in all the tests run, certainly some where better than others, but the distinction is stark. This could very well indicate that AGRO's price movements are more explainable given its market data, and allows for price prediction with high enough accuracy that it allows a well performing RL policy to be developed throughout training.

4.4 Discussion

To begin the discussion, the original research questions are brought back into focus. With respect to whether PPO can be successfully applied to the time series forecasting problem, we define successfully as not incurring a net loss during testing.

This was shown to be true given the set-up and dataset used in this work. For R2, *What is an optimal configuration using the PPO architecture in terms of learning rate, discount factor, number of action spaces, input features, and look-back length, given SPY market data?*, we define optimal as the best performing in regard to total profit during validation. For this, it was determined to be a learning rate of 0.0001, discount factor of 0.95, action space size of 4, look-back of 10 days, and using technical indicators. With that said, there are limitations to the sequencing of experiments as well as including the technical indicator features, which will be described further after revisiting the rest of the original research questions. As for R3, PPO did not outperform a B&H strategy on the SPY dataset with regarding profit or annualized return, but did slightly on MDD. For R4 and the portability of the trained model, the best performing PPO model trained on the SPY dataset ported over well to the Piotroski dataset, outperforming a B&H strategy. However, with that in mind and in reference to R6, performing transfer learning did not outperform the same model without transfer learning. This could be due to the model and experiment configuration needing further tuning for this new dataset to achieve better performance, but further experimentation is needed to validate such. This leads well into R6 and the portability of the optimal parameters from the SPY experiments. Specifically, under question is whether a positive return could still be maintained using the parameters deemed appropriate on the SPY dataset - which it could. Knowing this though, and due to the results of the transfer learning experiments, it is recommended for future work to use the values converged upon in the SPY experiments only as a starting point to reduce the scope of possible values.

One considerable limitation with the approach used in this work has to do with the sequence of experiments and number of parameters experimented with. The sequence chosen for this work was one of many possible, and thus it cannot be said for certain that the values determined to be ideal are better than those using a different sequence.

Not only this, but because only a small number of parameters were considered, there could exist different values for the parameters considered had other parameters also been considered and experimented with.

Another limitation with this work would be the choice to include technical indicators for subsequent experiments, despite it actually decreasing performance during validation. The reason this was done was because of how little the presence appeared to affect the performance, and the thought that potentially modifying parameters in subsequent experiments with the presence of technical indicators could lead to better performance than without. Had the presence of the technical indicators dramatically decreased the performance, they would have been excluded. This limitation however, brings about a more theoretical point about whether or not technical indicators should be used as features in general. Since neural networks are able to abstract the data any way training guides it to, one may argue that if certain indicators are the right abstractions of price and volume data that the network should learn this. That said, without experimentation counter arguments can be easily made. It is recommended that future work focus on ablation experiments with specific technical indicators to garner high quality evidence for or against their use.

There is also limitation in the experimental design regarding PPO as the selected model of interest and only using the best performing model during validation. Since PPO chooses its actions stochastically, even after training, the best performing model from one validation episode could, to some degree, be because of chance. However, because we see a decrease in the entropy loss values during training, the number of stochastic actions taken during validation should be relatively small. Not only this, but a model is saved every time the previous mean reward from the last log step is exceeded during the many training episodes, leading to a large number of models being tested on the validation set. Nevertheless, this is still a limitation and future work should consider repeating the validation experiments multiple times for each

model for quantifiable certainty from validation.

It is clearly apparent from the experiments that a B&H strategy is only outperformed some of the time by PPO. With such little effort being needed to implement a BH strategy, one may question the value of using PPO at all. The value of this thesis is exploring PPO as an algorithm of choice for the task at hand, as well as the model and experiment parameter configurations tested, since it had yet to be done for the time series forecasting problem. Since there is no guarantee the results of this work, or any in this space for that matter, are fully generalizable, the findings of this work act as a good basis for future work to explore further, including different experimental parameters. As mentioned earlier, an unintended finding of this work was highlighting the importance of stock or data selection for this task. Certain stocks are able to perform well with PPO while others are not, given the same set-up while only varying the data being used to train and test. Certain datasets or stocks can have different explanatory variables, or those which can vary in importance between stocks. This provides some evidence as to what may be more worthwhile to focus on as part of the entire investment process if using DRL agents to trade (i.e. stock selection, price forecasting, or portfolio allocation). If one could devise a way to select stocks or datasets which have a high degree of explained variance in their data, then it could very well allow for high performing policies to develop, and lead to increased performance.

The last point of discussion is the generalizability of this work. Although the experimental protocol and parameter configuration were portable to a new dataset for the experiments in this work, there is no guarantee that it would generalize to other datasets. Different datasets coming from other exchanges could have different dynamics leading to different final policy formations. As such, and to remain consistent with previous commentary, it is recommended that future work consider using this experimental protocol as a starting point and still consider exploring a number

of parameter combinations for each dataset.

Chapter 5

Conclusion and Future Work

This section brings this work to a close with final conclusions and suggestions for future work.

5.1 Conclusions

From this work, it can be seen that the process for obtaining a positively performing model was successful given the experimental protocol, however, outperforming a lower bound practical B&H strategy for one of the datasets, SPY, was not. This is not entirely surprising considering the dataset, as it is an ETF consisting of the top 500 US companies, of which are likely going to be quite stable over time, leaving little room for exploitation past a B&H strategy. The hope was that given the protocol used, a better performing model capable of identifying large changes in overall trends would be discovered, and times of significant down trending could be avoided, unfortunately this is not to be true for the SPY dataset. That said, the portability of the best model from the SPY validation experiments was deemed to be fairly successful. With the average annualized return for our model, when considering all testing sets, being a couple percent higher than that of a B&H strategy. Not only this, but the portability of the experimental protocol used within this work was deemed to be fairly good

considering that 4/6 of the best models during the Piotroski validation experiments outperformed a B&H strategy, with 1 out of these 4 continuing to outperform a B&H strategy during testing by a significant extent. In addition to this, when averaging across all datasets AR, we see that during validation the average AR is almost double that of a B&H as well as very comparable to such during testing for the Piotroski model experiments. This indicatesa high potential upside, with very little potential down side, especially considering online or continued training was not employed in this work.

5.2 Future Work

Since the experimental protocol was deemed to be somewhat successful, the first recommendation is to further explore this, by using the same stock selection method found within this work (Piotroski & beta 1.0-1.5) but utilizing more stocks. Alternatively, one could use different stock selection techniques, potentially pertaining to only market data or fundamental data characteristics. It may also be worthwhile to incorporate macro-economic indicators into the state space in order to see if this could boost performance by potentially indicating large liquidity contractions or expansions in the market. Additionally, given the results in this work, it is believed that focusing on a security selection mechanism that employs any form of technique which selects for stocks which are highly explainable given its market data. Another recommendation would be to employ online training or to retrain each model after validation tests, to see if this has any significant impact.

Additionally, different model hyperparameters such as the entropy and value coefficients could be explored, however it is not believed that this is likely going to make much of a difference using the SPY dataset. One hyperparameter which may be worth

exploring, is the policy hyperparameter. This specifies which type of DL architecture to use, which very well could make a significant difference in performance. Not only has adding neural networks to reinforcement learning been historically successful, but experiments concerning DL alone show different architectures can cause dramatic performance gains. Unfortunately, SB3 only has two options, and only one of them makes sense given the nature of time series data. That said, there is no limitation to sub classing the part of the library concerning this, as it is well abstracted and lends well to this kind of customization. Another hyperparameter that could be explored would be that of the optimization algorithm used, we used Adam, however this could be replaced with something else such as AdaGrad, AdaMax, and many more. Some experts, such as the distinguished Michael I. Jordan, firmly believe the next steps in AI involve advancements in optimization[87], thus providing some credence to the importance of optimization algorithms and potentially exploring different ones for this task. We also suggest that it may be worthwhile to change the starting position of the agent at the beginning of every episode, as it could increase the novelty of the episode for the agent. Along a similar line of thought, one could invert the data at random in order to train for a more robust policy capable of dealing with inverse trading dynamics.

List of References

[1] J. M. Poterba and L. H. Summers, "Mean reversion in stock prices: Evidence and implications," *Journal of financial economics*, vol. 22, no. 1, pp. 27–59, 1988.

[2] B. Wyss and O. Wyss, *Fundamentals of the Stock Market*. Fundamentals of Investing, McGraw-Hill, 2001.

[3] N. Helms, R. Hölscher, and M. Nelde, "Automated investment management: Comparing the design and performance of international robo-managers," *European Financial Management*, 2021.

[4] N. G. Iannarone, "Rethinking automated investment adviser disclosure," in *The Routledge Handbook of FinTech*, pp. 347–357, Routledge, 2021.

[5] B. P. Edwards, "The rise of automated investment advice: can robo-advisers rescue the retail market," *Chi.-Kent L. Rev.*, vol. 93, p. 97, 2018.

[6] M. J. McGowan, "The rise of computerized high frequency trading: use and controversy," *Duke L. & Tech. Rev.*, vol. 9, p. 1, 2009.

[7] A. Briola, J. Turiel, R. Marcaccioli, and T. Aste, "Deep reinforcement learning for active high frequency trading," *arXiv preprint arXiv:2101.07107*, 2021.

[8] S. Gjerstad and J. Dickhaut, "Price formation in double auctions," *Games and economic behavior*, vol. 22, no. 1, pp. 1–29, 1998.

[9] R. Das, J. E. Hanson, J. O. Kephart, and G. Tesauro, "Agent-human interactions in the continuous double auction," in *International joint conference on artificial intelligence*, vol. 17, pp. 1169–1178, Lawrence Erlbaum Associates Ltd, 2001.

[10] T.-V. Pricope, "Deep reinforcement learning in quantitative algorithmic trading: A review," *arXiv preprint arXiv:2106.00123*, 2021.

[11] A. Mosavi, Y. Faghan, P. Ghamisi, P. Duan, S. F. Ardabili, E. Salwana, and S. S. Band, "Comprehensive review of deep reinforcement learning methods and applications in economics," *Mathematics*, vol. 8, no. 10, p. 1640, 2020.

[12] J. Leem and H. Y. Kim, "Action-specialized expert ensemble trading system with extended discrete action space using deep reinforcement learning," *Plos one*, vol. 15, no. 7, p. e0236178, 2020.

[13] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita, "Adaptive stock trading strategies with deep reinforcement learning methods," *Information Sciences*, vol. 538, pp. 142–158, 2020.

[14] K. Lei, B. Zhang, Y. Li, M. Yang, and Y. Shen, "Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading," *Expert Systems with Applications*, vol. 140, p. 112872, 2020.

[15] Z. Zhang, S. Zohren, and S. Roberts, "Deep reinforcement learning for trading," *The Journal of Financial Data Science*, vol. 2, no. 2, pp. 25–40, 2020.

[16] S. Carta, A. Corriga, A. Ferreira, A. S. Podda, and D. R. Recupero, "A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning," *Applied Intelligence*, vol. 51, no. 2, pp. 889–905, 2021.

[17] N. Bauerle and U. Rieder, *Markov decision processes with applications to finance*. Springer Science & Business Media, 2011.

[18] S. D. Holcomb, W. K. Porter, S. V. Ault, G. Mao, and J. Wang, "Overview on deepmind and its alphago zero ai," in *Proceedings of the 2018 international conference on big data and education*, pp. 67–71, 2018.

[19] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[20] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Perez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[21] J. D. Piotroski, "Value investing: The use of historical financial statement information to separate winners from losers," *Journal of Accounting Research*, pp. 1–41, 2000.

[22] J. De la Vega, *Confusion de Confusiones [1688]: Portions Descriptive of the Amsterdam Stock Exchange*. No. 13, Colchis Books, 1957.

[23] S. Nison, *Japanese candlestick charting techniques: a contemporary guide to the ancient investment techniques of the Far East*. Penguin, 2001.

[24] S. Nison, *Beyond candlesticks: New Japanese charting techniques revealed*, vol. 56. John Wiley & Sons, 1994.

[25] S. Emir, H. Dincer, and M. Timor, "A stock selection model based on fundamental and technical analysis variables by using artificial neural networks and support vector machines," *Review of Economics & Finance*, vol. 2, no. 3, pp. 106–122, 2012.

[26] C. D. Kirkpatrick II and J. A. Dahlquist, *Technical analysis: the complete resource for financial market technicians*. FT press, 2010.

[27] C. L. Osler and P. Chang, "Head and shoulders: Not just a flaky pattern," *FRB of New York staff report*, no. 4, 1995.

[28] S. C. Suh, D. Li, and J. Gao, "A novel chart pattern recognition approach: A case study on cup with handle," in *Proc of Artificial Neural Network in Engineering Conf*, Citeseer, 2004.

[29] J. S. Abarbanell and B. J. Bushee, "Fundamental analysis, future earnings, and stock prices," *Journal of accounting research*, vol. 35, no. 1, pp. 1–24, 1997.

[30] B. Graham and D. Dodd, *Security Analysis*, vol. 6. McGraww Hill, 2008.

[31] R. Wigglesworth, "Fintech: Search for a super-algo," *Financial Times*, vol. 20, 2016.

[32] M. L. De Prado, "Invited editorial comment: Mathematics and economics: A reality check," 2016.

[33] A. Mittal and A. Goel, "Stock prediction using twitter sentiment analysis," *Standford University, CS229*, vol. 15, 2012.

[34] M. Ryabinin and A. Gusev, "Towards crowdsourced training of large neural networks using decentralized mixture-of-experts," *arXiv preprint arXiv:2002.04013*, 2020.

[35] A. Tealab, "Time series forecasting using artificial neural networks methodologies: A systematic review," *Future Computing and Informatics Journal*, vol. 3, no. 2, pp. 334–340, 2018.

[36] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020.

[37] C. Chatfield, *Time-series forecasting*. CRC press, 2000.

[38] N. Jahan, J. J. Cheh, and I.-w. Kim, "A comparison of graham and piotroski investment models using accounting information and efficacy measurement," *Journal of Economic & Financial Studies*, vol. 4, no. 01, pp. 43–54, 2016.

[39] H. Markowitz, "Modern portfolio theory," *Journal of Finance*, vol. 7, no. 11, pp. 77–91, 1952.

[40] H. Marling and S. Emanuelsson, "The markowitz portfolio theory," *November*, vol. 25, p. 2012, 2012.

[41] O. Iyiola, Y. Munirat, and C. Nwufo, "The modern portfolio theory as an investment decision tool," *Journal of Accounting and Taxation*, vol. 4, no. 2, pp. 19–28, 2012.

[42] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[43] S. Bhatt, "Reinforcement learning 101," Apr 2019.

[44] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[45] M. Coggan, "Exploration and exploitation in reinforcement learning," *Research supervised by Prof. Doina Precup, CRA-W DMP Project at McGill University*, 2004.

[46] M. Aloqaily, S. Otoum, I. Al Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles in smart cities," *Ad Hoc Networks*, vol. 90, p. 101842, 2019.

[47] A. massoud Farahmand, A. Shademan, M. Jagersand, and C. Szepesvári, "Model-based and model-free reinforcement learning for visual servoing," in *2009 IEEE International Conference on Robotics and Automation*, pp. 2917–2924, IEEE, 2009.

[48] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[49] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: an overview," in *Proceedings of SAI Intelligent Systems Conference*, pp. 426–440, Springer, 2016.

[50] J. Wen, S. Kumar, R. Gummadi, and D. Schuurmans, "Characterizing the gap between actor-critic and policy gradient," *arXiv preprint arXiv:2106.06932*, 2021.

[51] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[52] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[53] 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*, vol. 28, no. 3, pp. 653–664, 2016.

[54] Y. Li, P. Ni, and V. Chang, "Application of deep reinforcement learning in stock trading strategies and stock forecasting," *Computing*, pp. 1–18, 2019.

[55] G. G. Calvi, V. Lucic, and D. P. Mandic, "Support tensor machine for financial forecasting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8152–8156, IEEE, 2019.

[56] O. B. Sezer and A. M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, vol. 70, pp. 525–538, 2018.

[57] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," *arXiv preprint arXiv:1706.10059*, 2017.

[58] Z. Liang, H. Chen, J. Zhu, K. Jiang, and Y. Li, "Adversarial deep reinforcement learning in portfolio management," *arXiv preprint arXiv:1808.09940*, 2018.

[59] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," in *2017 Intelligent Systems Conference (IntelliSys)*, pp. 905–913, IEEE, 2017.

[60] P. Yu, J. S. Lee, I. Kulyatin, Z. Shi, and S. Dasgupta, "Model-based deep reinforcement learning for dynamic portfolio optimization," *arXiv preprint arXiv:1901.08740*, 2019.

[61] Y. Zhang, P. Zhao, B. Li, Q. Wu, J. Huang, and M. Tan, "Cost-sensitive portfolio selection via deep reinforcement learning," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[62] X. Li, Y. Li, Y. Zhan, and X.-Y. Liu, "Optimistic bull or pessimistic bear: Adaptive deep reinforcement learning for stock portfolio allocation," *arXiv preprint arXiv:1907.01503*, 2019.

[63] L. T. Hieu, "Deep reinforcement learning for stock portfolio optimization," *arXiv preprint arXiv:2012.06325*, 2020.

[64] W. Shin, S.-J. Bu, and S.-B. Cho, "Automatic financial trading agent for low-risk portfolio management using deep reinforcement learning," *arXiv preprint arXiv:1909.03278*, 2019.

[65] J. Wang, Y. Li, and Y. Cao, "Dynamic portfolio management with reinforcement learning," *arXiv preprint arXiv:1911.11880*, 2019.

[66] K. Yashaswi, "Deep reinforcement learning for portfolio optimization using latent feature state space (lfss) module," *arXiv preprint arXiv:2102.06233*, 2021.

[67] N. Kanwar, *Deep Reinforcement Learning-Based Portfolio Management*. PhD thesis, The University of Texas at Arlington, 2019.

[68] P. Koratamaddi, K. Wadhwani, M. Gupta, and S. G. Sanjeevi, "Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation," *Engineering Science and Technology, an International Journal*, vol. 24, no. 4, pp. 848–859, 2021.

[69] C. Betancourt and W.-H. Chen, "Deep reinforcement learning for portfolio management of markets with a dynamic number of assets," *Expert Systems with Applications*, vol. 164, p. 114002, 2021.

[70] 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)*, pp. 1–8, IEEE, 2020.

[71] Q. Kang, H. Zhou, and Y. Kang, "An asynchronous advantage actor-critic reinforcement learning method for stock selection and portfolio management," in *Proceedings of the 2nd International Conference on Big Data Research*, pp. 141–145, 2018.

[72] S.-H. Huang, Y.-H. Miao, and Y.-T. Hsiao, "Novel deep reinforcement algorithm with adaptive sampling strategy for continuous portfolio optimization," *IEEE Access*, vol. 9, pp. 77371–77385, 2021.

[73] N. N. Vo, X. He, S. Liu, and G. Xu, "Deep learning for decision making and the optimization of socially responsible investments and portfolio," *Decision Support Systems*, vol. 124, p. 113097, 2019.

[74] Y. Ye, H. Pei, B. Wang, P.-Y. Chen, Y. Zhu, J. Xiao, and B. Li, "Reinforcement-learning based portfolio management with augmented asset movement prediction states," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 1112–1119, 2020.

[75] A. Chaouki, S. Hardiman, C. Schmidt, E. Sérié, and J. De Lataillade, "Deep deterministic portfolio optimization," *The Journal of Finance and Data Science*, vol. 6, pp. 16–30, 2020.

[76] 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)*, pp. 268–275, IEEE, 2020.

[77] 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*, pp. 1–8, 2020.

[78] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, "Reinforcement learning, fast and slow," *Trends in cognitive sciences*, vol. 23, no. 5, pp. 408–422, 2019.

[79] A. Filos, "Reinforcement learning for portfolio management," *arXiv preprint arXiv:1909.09571*, 2019.

[80] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[81] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," *GitHub repository*, 2019.

[82] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[83] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[84] A. Raffin, "Rl baselines zoo." `https://github.com/araffin/rl-baselines-zoo`, 2018.

[85] 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*, vol. 518, no. 7540, pp. 529–533, 2015.

[86] A. Shiryaev, Z. Xu, and X. Y. Zhou, "Thou shalt buy and hold," *Quantitative finance*, vol. 8, no. 8, pp. 765–776, 2008.

[87] A. Wibisono, A. C. Wilson, and M. I. Jordan, "A variational perspective on accelerated methods in optimization," *proceedings of the National Academy of Sciences*, vol. 113, no. 47, pp. E7351–E7358, 2016.