Stock Trading with Reinforcement Learning: Average Reward Agent and Alpha Preservation

Wenyao Zhou
University of Virginia,
Mathematics and Computer Science, B.A.
`wz8ry@virginia.edu`

Distinguished Major Thesis in Computer Science

Abstract

This thesis investigates the application of reinforcement learning (RL) in stock trading using historical market data for 30 Dow Jones component stocks. We make two contributions. The first is to identify and mitigate a defect of the commonly used discounted total reward-based solutions. Namely, the performance of those solutions heavily depend on the selection of the discount factor but there is no discount factor that can consistently perform well across different settings. We instead propose average-reward-based solutions, which consistently perform well and do not have the discount factor to tune. The second is to identify the positive effect of bootstrapping in alpha preservation. In our average-reward-based solutions, we demonstrate that it is possible to enhance the agent's resilience to abnormal market incidents (i.e., alpha preservation), such as the 2020 stock market crash, by increasing the degree of bootstrapping. Overall, this thesis underscores the potential of average-reward RL in stock trading for both efficiency and strategic alpha preservation during market anomalies.

Keywords: Reinforcement learning; Average reward; Stock trading; Alpha preservation; Market anomalies; Bootstrapping

1. Introduction

In the quest to mimic human decision-making processes and learning behaviors, Reinforcement Learning (RL) has emerged as a pivotal area of study within the broader field of artificial intelligence. RL is predicated on the principle of interaction between an agent and its environment, where the agent learns to make decisions by performing actions and receiving feedback in the form of rewards. This learning paradigm is distinct from supervised and unsupervised learning, as it focuses on sequential decision-making and learning from the consequences of actions, rather than from direct instruction or finding hidden patterns (Szepesvari, 2010).

As RL has evolved, the integration of neural networks has led to the development of Deep Reinforcement Learning (DRL), a subfield that combines the decision-making power of RL with the pattern-recognition capabilities of deep learning. DRL has remarkably advanced the frontiers of research in control problems across various domains (Mnih *et al.*, 2013; Mahmood *et al.*, 2018; Silver *et al.*, 2016), setting new benchmarks for performance and adaptability. In this thesis, we explore the application of DRL within the financial sector, specifically in stock trading.

1.1 DRL in stock trading

RL is distinguished from other fields of machine learning by its focus on learning optimal policies through trial-and-error interactions with a dynamic environment, a concept that is inherently aligned with the decision-making processes in financial trading. As a result, investment science is a promising area where DRL's capabilities can be harnessed. By leveraging complex historical data and stimulating countless trading scenarios, the DRL approaches demonstrate major advantages in portfolio scalability with reasonable computational effort, and independence from the investment model (Buehler *et al.*, 2019). Recently, extensive studies have used various DRL algorithms to train trading agents, conducted trading experiments on daily close prices of stocks, and reported high-than-market-average rates of returns (Guan and Liu, 2022; Liu *et al.*, 2022; Yang *et al.*, 2020; Li *et al.*, 2019).

The majority of these studies define the step reward as the daily change in portfolio value and used discounted-reward-based algorithms, such as Deep Q Learning (DQN, Mnih *et al.* (2013)) and Proximal Policy Optimization (PPO, Schulman *et al.* (2017)). Among the hyperparameters of the models, the discounting factor γ is of particular importance, as it influences how future rewards are valued relative to immediate rewards. However, the use of "naive but classical values" can lead to sub-optimal policies (Perotto and Vercouter, 2018). Works by Pitis (2019) and Kim *et al.* (2022) have also highlighted the performance limit of using a constant discounting factor. It is thus a major challenge to choose suitable values for the discounting factor.

In this thesis, we propose a novel approach to this challenge in stock trading by considering a portfolio's rate of return as an aggregate measure of daily changes instead of a discounting measure. Since most studies have used rate of return as the key performance benchmark (Guan and Liu, 2022; Liu *et al.*, 2022; Li *et al.*, 2019), we hypothesize that average-return reinforcement learning (Tadepalli and Ok, 1998; Wei *et al.*, 2021; Ma *et al.*, 2021) is an appropriate approach to train stock trading agents. The family of average-return reinforcement learning algorithms eliminates the use of discounting factors, thus reducing the complexity of hyperparameter tuning. In this thesis, we provide a comparison between the empirical performance of average-reward and discounted-reward reinforcement learning agents using historical market data.

1.2 Alpha preservation

Furthermore, machine learning models often show susceptibility to abnormal training data (Evtimov *et al.*, 2017; Jain *et al.*, 2020), and reinforcement learning is no exception. Stock trading is a time-driven task that requires constantly updating the model's training data as time proceeds. The agent can receive abnormal training data in the event of market anomalies, such as the 2020 Stock Market Crash that began in late February and spanned the entire March. We observe that the agents can demonstrate a variant of catastrophic forgetting problem (Kemker *et al.*, 2018) in such events, as they forgo the learned trading strategy, adjust to the abnormal market pattern, and perform poorly when the market resumes to normality. Typical solutions to catastrophic forgetting require extensive re-design of the algorithm (Kirkpatrick *et al.*, 2017), but in the task of stock trading, Yang *et al.* (2020) have proposed and tested an ensemble strategy where when the market volatility index passes a given threshold, the agent performs a straightforward loss-cutting strategy. The prominent performance of this ensemble strategy with even the simplest all-sell strategy highlights the effectiveness of human intervention in market crashes. Therefore, in this thesis, we aim to provide an uncomplicated way to delay catastrophic forgetting to allow buffer time for human traders to make adjustments.

In ensuring this delayed shift in model behavior, we effectively preserve the agent's ability to make a profit in a normal market after a considerable amount of abnormal market data is added to the training set. We describe the agent's profitability with the alpha in the Capital Asset Pricing Market (CAPM) model (Sharpe, 1964), the most popular models in investment science (Kumar *et al.*, 2023). The alpha coefficient describes the excess rate of return the portfolio makes, relative to the expected return from its risk level. Existing studies in protecting the trading strategy's alpha against volatile market conditions have focused primarily on conventional mean-variance methods (Sorensen *et al.*, 2004), whereas we aim to present a novel DRL approach to alpha preservation. We will empirically compute the alphas under normal market conditions using agents' strategies trained using data before and after the market crash to evaluate the agents' ability to delay catastrophic forgetting.

2. Backgrounds

2.1 MDP setup and value functions

We will model stock trading as a Markov decision process (MDP). We consider an infinite-horizon MDP (Puterman, 2014) with a finite state space $\mathcal{S}$, a finite action space $\mathcal{A}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a transition function $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, and a discount factor $\gamma \in [0, 1]$. We begin with the more popular setup, the discounted-reward MDPs with $\gamma < 1$, and we will discuss the average-reward MDPs with $\gamma = 1$ later.

Let $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ denote a stochastic policy of an agent. At each time step t, the agent at state $s_t \in \mathcal{S}$ chooses an action $a_t \in \mathcal{A}$ according to π, so $a_t \sim \pi(\cdot \mid s_t)$. A reward $R_{t+1} \coloneqq r(s_t, a_t)$ is then emitted, and the successor state s_{t+1} is sampled from $p(\cdot \mid s_t, a_t)$. Let τ denote a trajectory $(s_0, a_0, s_1, a_1, \ldots)$, and $\tau \sim \pi$ denote that the distribution of trajectories depends on $\pi : s_0 \sim d_0$, $a_t \sim \pi(\cdot \mid s_t)$, $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.

Under the discounted reward criterion with $\gamma \in [0, 1)$, the agent aims to optimize the policy π to maximize a normalized expected discounted long-term return

$$\eta_{\pi,\gamma} \coloneqq (1 - \gamma)\mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid s_0\right]. \tag{2.1}$$

We define the state value function $V : \mathcal{S} \to \mathbb{R}$ and the action-state value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, also known as the Q-value, as

$$\begin{aligned} V_\pi(s) &\coloneqq \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s\right], \\ Q_\pi(s, a) &\coloneqq \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, a_0 = a\right]. \end{aligned} \tag{2.2}$$

The estimations for V and Q are usually initiated as 0 and stored in an array, then updated through the training process. There are many methods for the agent to adjust its estimated state value $V_\pi(s)$ during training. One of them is Temporal Difference learning with parameter *lambda* (TD(λ).

2.2 TD and GAE with lambda

To begin with, we present the formulation of TD(λ) (Sutton, 1988). We define the eligibility trace for state $s \in \mathcal{S}$ at time $t \neq 0$ as

$$E_t(s) \coloneqq \gamma\lambda E_{t-1}(s) + \mathbb{1}_{s_t=s}, \tag{2.3}$$

where the indicator function $\mathbb{1}_{s_t=s} = 1$ if $s_t = s$, and 0 otherwise. Additionally, for all $s \in \mathcal{S}$, $E_0(s) = 0$. In learning, $E_t(s)$ is updated at each time step for all states. decaying with a factor of $\gamma\lambda$. Then, we define the TD error at time t as

$$\delta_{\pi,t} \coloneqq R_{t+1} + \gamma V_\pi(s_{t+1}) - V_\pi(s_t), \tag{2.4}$$

which calculates the difference between the observed reward plus the discounted value of the next state and the value of the current state. At each training time t, the agent updates its estimate for the state value function by

$$V_\pi(s_t) = V_\pi(s_t) + \alpha\delta_{\pi,t}E_t(s_t), \tag{2.5}$$

where $\alpha \in (0, 1)$ is the learning rate.

In TD(λ), if $\lambda = 0$, then the algorithm reduces to a one-step temporal difference method, and the value function is updated based solely on the immediate reward and the value of the next state

$$V_\pi(s) = V_\pi(s) + \alpha[R_{t+1} + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)].$$

The updated value demonstrates low variance, but high bias under $TD(0)$, since the temporal difference does not reflect the consideration for all future rewards.

If $\lambda = 1$, then the algorithm would execute until the end of the episode, and then update the value of the current state, effectively reducing the algorithm to a Monte Carlo simulation. The rule for updating the state value function in TD(1) is

$$V_\pi(s) = V_\pi(s) + \alpha[G_{\pi,t} - V_\pi(s)],$$

where $G_{\pi,t}$ is the return at time step t defined as

$$G_{\pi,t} \coloneqq \sum_{\ell=0}^{\infty} \gamma^\ell R_{t+\ell+1}. \tag{2.6}$$

In TD(1), the update accounts for all future rewards, leading to lower bias from the true value. However, the variance in the update is high as it is the sum of many estimates.

An intermediate λ value thus provides a balanced blending of both methods, with higher λ leading to greater proximity to Monte Carlo and lower λ leading to greater proximity to temporal difference. The choice of λ thus provides a trade-off between bias and variance.

The generalized advantage estimator (GAE(λ)) is an extension of the TD(λ) approach specifically tailored to the needs of policy gradient methods (Schulman *et al.*, 2015). The Proximal Policy Optimization (PPO) algorithm, which we will discuss later in this thesis, uses GAE(λ) to make updates. To illustrate the GAE(λ), we first define the advantage function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ in a discounted reward problem:

$$A_\pi(s, a) \coloneqq Q_\pi(s, a) - V_\pi(s). \tag{2.7}$$

In this way, the advantage function $A_\pi(s, a)$ calculates how much better taking the particular action a in state s is compared to the average reward across all actions in s, under policy π. If the agent takes action a_t in state s_t at time step t, the GAE(λ) advantage estimation is then

$$A_\pi^{GAE(\lambda)}(s_t, a_t) = \sum_{\ell=0}^{\infty} (\gamma\lambda)^\ell \delta_{t+\ell}, \tag{2.8}$$

where $\delta_{\pi,t+\ell}$ is the TD error at time step $t+\ell$, as defined in Eq.2.4. In practice, the estimation is often computed recursively, using the formula

$$A_{\pi}^{GAE(\lambda)}(s_t,a_t) = (\gamma\lambda)A_{\pi}^{GAE(\lambda)}(s_{t+1},a_{t+1}).$$

Similar to TD(λ), there are two notable observations of Eq.2.8 when setting $\lambda=0$ and $\lambda=1$.

$$\begin{aligned} A^{GAE(0)}(s_t,a_t) &= \qquad \delta_{\pi,t} &&= R_{t+1}+\gamma V_{\pi}(s_{t+1})-V_{\pi}(s_t),\\ A^{GAE(1)}(s_t,a_t) &= \textstyle\sum_{\ell=0}^{\infty}\gamma^{\ell}\delta_{t+\ell} &&= \sum_{\ell=0}^{\infty}\gamma^{\ell}R_{t+\ell+1}-V_{\pi}(s_t)=G_{\pi,t}-V_{\pi}(s_t). \end{aligned}$$

Like TD(λ), GAE(1) reduces the estimation to a Monte Carlo simulation. It has low bias regardless of the accuracy of current V_π but high variance due to the sum of terms. On the other hand, GAE(0) reduces the estimation to the temporal difference. It has a much lower variance but induces bias, as this estimation is contingent on the accuracy of the current V_π.

2.3 Actor-critic framework

The Actor-critic methods consist of two main components: the actor and the critic. The actor is responsible for selecting actions given the current state, while the critic evaluates the actions taken by the actor by computing its expected return (Konda and Tsitsiklis, 2003). We make slight modifications to the definitions we have provided so far and define

- **Actor**: The policy function, $\pi_\theta(a \mid s)$, maps state to a probability distribution over actions. The actor aims to learn a policy that maximizes the expected return.
- **Critic**: The value function $V_\phi(s)$ estimates the expected return from s, which we have defined as the discounted total future rewards. The critic's role is to evaluate the quality of the states encountered by the actor, providing feedback on the actor's performance.

The new parameters θ and ϕ represent the parameters of the function approximator that models the value function. The actor approximator is often a neural network and the parameters θ are adjusted in training to maximize the expected return. Meanwhile, the parameters ϕ are adjusted to minimize the error in value estimates.

Naive policy gradient methods work by computing an estimator of the policy gradient and then use a stochastic gradient ascent algorithm to find the optimal θ. The most commonly used gradient estimator is obtained by differentiating the objective

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \, \hat{A}_t \right],$$

where $\hat{A}_t$ is the advantage estimate for s_t, a_t. However, Schulman *et al.* (2017) argues that empirically, performing steps of optimization on this loss L^{PG} often leads to destructively large policy updates. They thus propose the PPO algorithm.

2.4 PPO for discounted reward

Proximal Policy Optimization (PPO) is a popular policy gradient method in reinforcement learning that seeks to improve the efficiency and reliability of policy updates. It is designed to take multiple small steps in updating the policy using the same data, which helps in preventing large destructive updates. The core idea of PPO is to maintain a balance between exploration and exploitation by limiting the changes to the policy at each update. This is achieved through a specially designed objective function using a clipping function that discourages large deviations from the current policy (Schulman *et al.*, 2017).

The clipped objective function for PPO is

$$L(\theta) \coloneqq \hat{\mathbb{E}} \left[\min \left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} \hat{A}_t, \operatorname{clip} \left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}, 1 - \epsilon, 1 + \epsilon \right) \right) \right], \tag{2.9}$$

where ϵ is a hyperparameter to tune, and $\hat{A}_t = A^{GAE(\lambda)(s_t, a_t)}$ is the advantage estimation that we implement for our experiments. Given that the formulation of PPO through its clipped objective function effectively balances exploration and exploitation in discounted reward problems, we now turn our focus to the average reward problems.

2.5 Average reward policy optimization

Using the same infinite-horizon MDP definitions in section 2.1, an average-reward agent aims to maximize the long-run average performance under π

$$\eta_\pi \coloneqq \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} R_{t+1}\right]. \tag{2.10}$$

We use the setup by Ma *et al.* (2021) to define value functions as

$$\begin{aligned} V_\pi(s) &\coloneqq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}(r_{a_t}(s_{t+1}, s_t) - \eta_\pi) \mid S_0 = s\right], \\ Q_\pi(s,a) &\coloneqq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}(r_{a_t}(s_{t+1}, s_t) - \eta_\pi) \mid S_0 = s, a_0 = 0\right], \\ A_\pi(s,a) &\coloneqq Q_\pi(s,a) - V_\pi(s). \end{aligned} \tag{2.11}$$

Unlike the discounted reward problem where η_π needs not to be explicitly estimated using Eq.2.1, here for each episode in the training process, we update the estimate of the long-run average reward by

$$\hat{\eta} = (1-\alpha)\hat{\eta} + \alpha\frac{1}{N}\sum_{n=1}^{N} R_{n+1}, \tag{2.12}$$

where N is the number of time steps collected in the episode and α is the learning rate. The original discounted GAE estimator is thereby modified to an average-reward variant as

$$\hat{A}_t = \hat{A}(s_t, a_t) = \sum_{\ell=0}^{\infty} \lambda^\ell \delta_{t+\ell}, \tag{2.13}$$

where the new TD error is defined as

$$\delta_t = R_{t+1} - \hat{\eta} + V(s_{t+1}) - V(s_t). \tag{2.14}$$

Then, the average-reward agent should aim to optimize its policy parameters θ to minimize the same surrogate loss as the discounted-reward PPO in Eq.2.9, with the updated advantage estimator in Eq.2.13.

Similar to the discounted GAE(λ) estimator, we make two observations of Eq.2.13 with $\lambda = 0$ and $\lambda = 1$:

$$\begin{aligned}\hat{A}_t^{\lambda=0} &= \delta_t = R_{t+1} - \hat{\eta} + V(s_{t+1}) - V(s_t),\\ \hat{A}_t^{\lambda=1} &= \sum_{\ell=0}^{\infty} \delta_{t+\ell} = \sum_{\ell=0}^{\infty} R_{t+\ell+1} - \hat{\eta} + V(s_{t+\ell+1}) - V(s_{t+\ell}).\end{aligned} \tag{2.15}$$

Therefore, similar to the discounted GAE(λ), the average reward advantage estimator also updates $\hat{A}_t$ as the temporal difference when $\lambda = 0$, which has low variance but introduces bias due to inaccurate estimations in the current $V(S_{t+1})$'s. When $\lambda = 1$, the average reward advantage estimator updates $\hat{A}_t$ using a Monte Carlo simulation, leading to low to no bias but high variance, due to the sum of the estimated TD error terms $\delta_{t+\ell}$. This hyperparameter λ thus provides the trade-off between bias and variance.

2.6 CAPM model and alpha preservation

The Capital Asset Pricing Model (CAPM) is a cornerstone of modern financial theory that offers a method to assess the expected return on an investment and quantify the systematic risk associated with it. At its core, CAPM assumes that the expected return on a portfolio equals the rate on a risk-free security plus a risk premium. The risk premium is derived from the *market* risk, also known as the *non-diversifiable* or *systematic* risk, which cannot be mitigated through diversification (Sharpe, 1964). This risk is measured by the beta (β) coefficient, which represents the tendency of the security's returns to respond to swings in the market. The CAPM is encapsulated by the equation:

$$\mathbb{E}[R_{p,t}] = R_f + \beta(\mathbb{E}[R_{m,t}] - R_f), \tag{2.16}$$

where

- $\mathbb{E}[R_{p,t}]$ is the expected return rate on the portfolio p over time;
- R_f is the risk-free rate of return, which is usually the rate of Treasury bond;

- $\mathbb{E}[R_{m,t}]$ is the expected return rate of the market over time.

In the context of portfolio management, the alpha (α) coefficient, also known as "Jensen's alpha", denotes the excess return on an investment relative to the return predicted by the CAPM. It is computed as the intercept term in a time series regression derived from Eq.2.16:

$$R_{p,t} - R_f = \alpha_p + \beta_p(R_{m,t} - R_f) + \varepsilon_{p,t}, \tag{2.17}$$

where $R_{p,t}$ is the time series of the portfolio's realized rate of return (Fama and French, 2004).

Preservation of alpha, therefore, means safeguarding this excess return which represents the value added by the trading strategy. After adding the data of the market crash to the training data, if the DRL trading agent still maintains a high level of α, this indicates that the DRL agent demonstrates a certain level of resilience against abnormal market data in the training set.

3. Methodology

3.1 Gymnasium Environment: trading simulator

Gymnasium is a popular open-source Python library that provides a standard API to communicate between reinforcement learning algorithms and environments. It is a fork of the OpenAI Gym (Brockman *et al.*, 2016) by its maintainers at the Farama Foundation. Gymnasium includes a set of task environments for classic reinforcement-learning problems, as well as the ability to customize new environments through its Env API. We will exploit this ability to train and test our stock trading agents.

Stock trading tasks are stochastic and discrete-time in nature, so we model it as a Markov Decision Process (MDP) control problem. The customized environment should stimulate the real-world stock markets and provide realistic feedback, namely rewards, corresponding to interactions with the agent. We thus construct the simulator with the historical daily market data for the 30 constituent stocks of the Dow Jones Industrial Average index.

We formulate our environment as a modified variant of the implementation by Liu *et al.* (2022).

The State Space $\mathcal{S}$ is a set of the environment and agent states. In other words, at each time step t, the state $s_t \in S$ should describe the current state of the market and the current portfolio. Given our selection of stocks, s_t consists of

- Close price $p_t \in \mathbb{R}^{30}$: daily close prices that the agent trade on.
- Selected m indicators $M_t \in \mathbb{R}^{30 \times m}$: optional technical indicators for additional information.
- Current holding $h_t \in \mathbb{Z}^{30}$: current portfolio holding of each stock in shares.
- Cash balance $b_t \in \mathbb{R}$: cash component in the portfolio.

For simplicity, we include only the publicly available indicators of daily opening, high, and low prices, as well as the daily trading volume. We also restrict the cash balance to be non-negative and the shareholdings to be only non-negative whole numbers. After column-major order flattening of M_t, we have each state s_t as a vector of 181 elements.

The initial state s_0 of the environment is a vector with entry $b_0 = 1,000,000$ as cash balance, and entries of 0 elsewhere. This represents a starting wealth of one million dollars and nothing else. During training epochs, we can add random variations to b_0 to encourage exploration.

The action space $\mathcal{A}$ is the set of allowed actions that the agent can choose to perform. Due to nonnegative constraints on b_t and h_t, for each stock i of the 30 stocks, there is a maximum amount of shares that the agent can buy or sell, denoted as N_b, N_s. The stock's action space a_i is a continuous box from -1 to 1, where a negative value represents the proportion of N_s shares to sell, and a positive value represents the proportion of N_b shares to buy. The number of shares to trade is always rounded down, as fractional trading is not a common practice in stock trading.

The reward function $R_{t+1} = r(s_t, a_t)$ provides immediate feedback to the agent's actions. In the training process, the agent should adjust its policy to maximize the total rewards, discounted or undiscounted. Since our benchmark for agent performance is the rate of portfolio return, we define the step reward as changes in portfolio value, so

$$R_{t+1} \coloneqq h_{t+1}^T p_{t+1} + b_{t+1} - h_t^T p_t - b_t. \tag{3.18}$$

The agent thereby aims to maximize the total positive change in the value of the portfolio.

3.2 Experiment procedures

We explore two research questions in this thesis. The first question is formulated in a natural and intuitive manner, whereas the second question involves a more logically complex formulation. As presented in Eq. 2.15, the hyperparameter lambda (λ) adjusts the degree of bootstrapping between Temporal Difference (TD) and a Monte Carlo (MC) simulation of the actual total future return. A lower λ increases the weight of the TD estimation, thus leading to a higher bias from the latest training data. Since temporal differences are computed directly using the current model, we hypothesize that a lower λ may result in more dependence on the current policy. We thus want to explore the effect of lower λ values in tackling catastrophic forgetting in abnormal training data. In the context of stock trading, we formulate two research questions in this thesis:

1. Does average reward reinforcement learning agents perform at a level comparable to the discounted reward PPO agents, without the need to tune discounting factors?

2. Can a trader use a lower λ to make the average-reward agents preserve their lambda against longer market anomalies?

The trading environment is set up to answer these questions with empirical evidence. To ensure that our results are representative of the mainstream traders in the U.S. stock market, we focus on the latest data of the major companies, whose stocks are included as components of the Dow Jones Industrial Average (DJIA) index. We download the open market data of the 30 stocks from Yahoo Finance from Jan 1, 2015, to Jan 1, 2024 at a frequency of per trading day, and set up the environment for trading daily close prices, as detailed in Section 3.1.

We base the training of the agents on the Stable Baseline 3 (SB3, Raffin *et al.* (2021)) package. SB3 provides a set of reliable implementations of numerous reinforcement learning algorithms, including the PPO algorithm for discounted-reward MDPs. For the average-reward agents, we use the algorithm proposed by Ma *et al.* (2021), which only requires modifications to the advantage estimations of the basic PPO, as described in Section 2.5. Thus, our average-reward agents are trained by integrating SB3's PPO implementation with the updated advantage estimator in Eq.2.8.

To answer the first question, each agent is trained using data from the start of 2015 to the end of 2022. Then, we test the trained agent using a testing environment with the stock data of 2022 and 2023 to evaluate their performance in out-of-sample scenarios. We will train a batch of models with randomized seeds using different gamma values with discounted reward PPO, and then compare their performance to that of the batch trained with average-reward learning. We will answer the first question based on their rates of return over the two testing years.

To answer the second question, we will use the 2020 stock market crash as an example of market anomalies. Since the market crash spanned from February 20 to April 7, we make three roughly 15-day intervals with four dates: Feb 20, Mar 6, Mar 21, and Apr 5. We will use each date as the training ending date to train a random sample of average reward agents using different lambda values. Then, we test these trained agents using 2022-2024 data, when the market has resumed normality, to empirically calculate and compare their alpha performance in normal data. This comparison will answer the second research question.

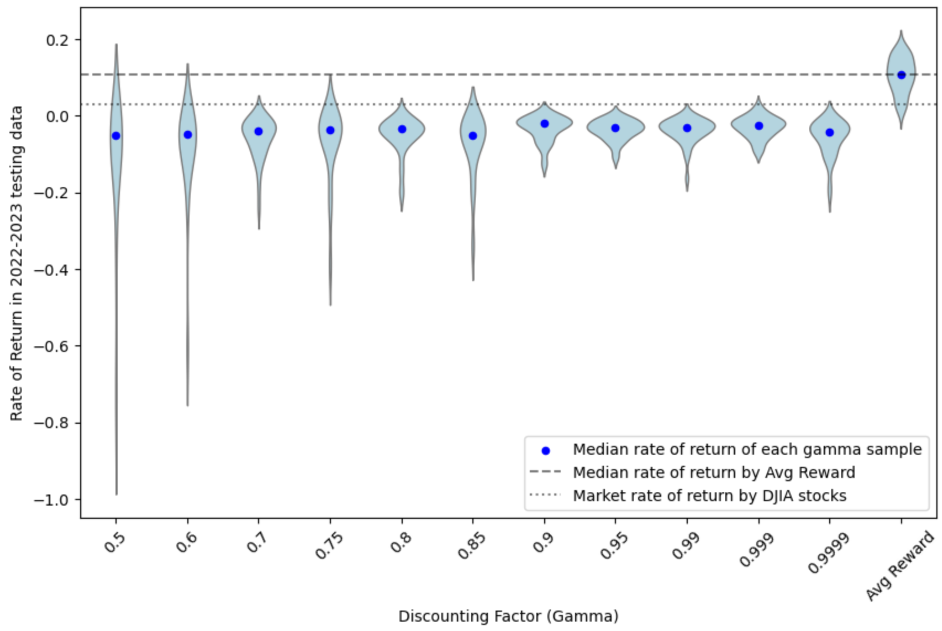

Fig. 1. **Violin diagram of the rates of return in each gamma's sample of trained agents.** The plot shows the variations in the rates of returns by agents trained with each gamma value. The width of the violin indicates the distribution at each rate of return. Compared to the discounted-reward agents, the sample of average-reward agents shows a very compact violin, indicating a consistent performance across different random seeds.

4. Results

4.1 Performance of average and discounted reward agents

We chose an array of common values for the discounting factor γ and trained a random sample of 50 discounted-reward PPO agents for each γ value while holding other hyperparameters constant at SB3's PPO default. The training data consisted of the 30 stocks' daily open-market data from Jan 01, 2015, to Dec 31, 2021, and we tested their trading actions using data from Jan 01, 2022, to Jan 01, 2024. We then applied the same testing procedures to a random sample of 50 average-reward agents.

Figure 1 presents a violin diagram comparing the distribution of mean relative return for agents trained across various gamma values. The widths of the violins at different returns indicate the

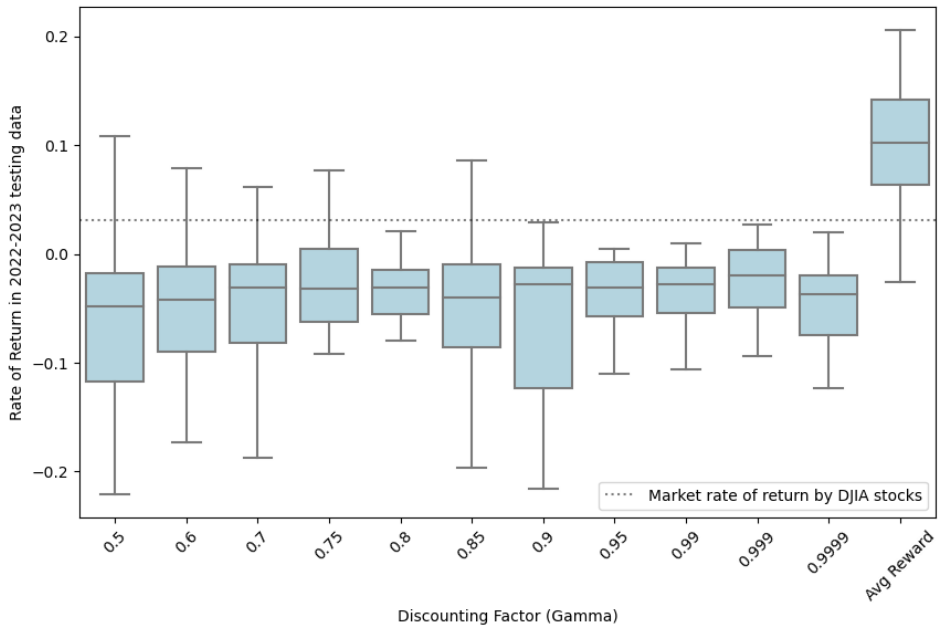

Fig. 2. **Box-plot of the rates of return in each gamma's sample with outliers removed.** This plot shows the stock-trading performances of the sample of agents trained with different discounting factors. We observe no clear monotonic trend in the median rates of returns as gamma changes, whereas the average-reward agents constantly outperform almost all of the discounted-reward agents.

probability density of the agents achieving that return. This visualization illustrates the variability and distribution of performance at each gamma level.

The data suggests an associativity between lower gamma values and broader distributions of returns, implying higher unpredictability in the performance of the agents. This might be due to lower gamma values placing less emphasis on future rewards, which can lead to short-sighted decision-making in the context of investment strategies. Conversely, as gamma values increase to approach 1, the distribution of returns narrows, indicating more consistent performance across different test cases.

The average-reward agent sample shows a very narrow distribution, hinting at a high degree of consistency across different agents. The relatively compact violin suggests that the average-reward approach provides a stable performance regardless of the random seeds, which might be indicative of a more reliable investment strategy in the face of diverse market conditions.

To better examine whether a clear optimization trend in gamma exists, we will examine the distribution of the rates of return relative to gammas with outliers removed. We define the interquartile range as the distance between a sample's first and third quartile and consider a data point an *outlier* if it is more than 1.5 times IQR lower than the first quartile or 1.5 times IQR higher than the third quartile. With these outliers removed, Figure 2 depicts a box plot of the rate of returns for the same sample of agents.

We observe no monotonic relationship between the median rate of return and the gamma values of the sampled models. This suggests a lack of a consistent optimization trend in tuning the discounting factor for discounted-reward agents. On the contrary, the box for the average-reward agents is positioned higher on the rate-of-return scale while demonstrating a smaller IQR than most of the boxes corresponding to the discounted-reward agents. This suggests that not only do the average-reward agents tend to produce higher returns in stock trading, but it also do so with less variance, reinforcing the interpretation from Figure 1 of it being a more consistent approach.

4.2 Market crash in training data for agents with different lambdas

We select a list of four common values for lambda, the hyperparameter that controls the degree of bootstrapping between Monte Carlo and temporal difference for advantage estimation in the average-reward algorithm. With a $\lambda \in [0.5, 0.6, 0.75, 0.85, 0.95, 0.999]$, we train a random sample of 10 average-reward agents for each of the training ending dates we sample before, during, and after the 2020 stock market crash, as described in Section 3.2. The starting date of the training data is Jan 01, 2015, for all agents. For the testing data, we do not use the data immediately after the market crash because the market experienced a long boom due to expansionary fiscal and monetary policies by the government to help the economy recover from the pandemic (Gravelle and Marples, 2021). To ensure that the market data for testing is representative of the market normality, we manually inspect the history of the DJIA index and select the testing data to be from Dec 01, 2021, to Jun 01, 2022, when the market had finished resuming from the crash. This allows us to evaluate their performance in a *normal* market when the market dynamics exhibit a similar paradigm to the market before the market crash.

Figure 3 provides a preliminary examination of how agents with different λ values respond to the presence of the market crash in training data. We calculate each agent's trading portfolio's current relative value as the ratio of the current portfolio value to the initial portfolio value. We then plot them by λ value, in groups of the training end time, to yield Figure 3. For readability, Figure 3 presents only the agents that give a median level of rate of return over the entire testing period at each choice of λ.

We would like to draw special attention to the changes to the portfolio value curve after the market crash has begun. It is noticed that the agents with $\lambda = 0.999$ and $\lambda = 0.95$ exhibit a great drop in portfolio value after the first 15 days of the market crash are added to the training data, whereas the agent with $\lambda = 0.5$ does not do so until nearly 45 days into the market crash. Although this naive direct inspection of the portfolio value curve gives some insight that may support our hypothesis, this approach has several caveats:

- Due to the stochastic nature of any deep algorithm, agent behavior may change abruptly in a way that is not meaningful for our analysis. An example is the agent with $\lambda = 0.75$ that demonstrates a considerable drop in portfolio value from the training ending date Jan 06 to Jan 21, even before the market crash or other market anomalies start.
- According to the CAPM model, trading portfolios with higher profits tend to have a higher

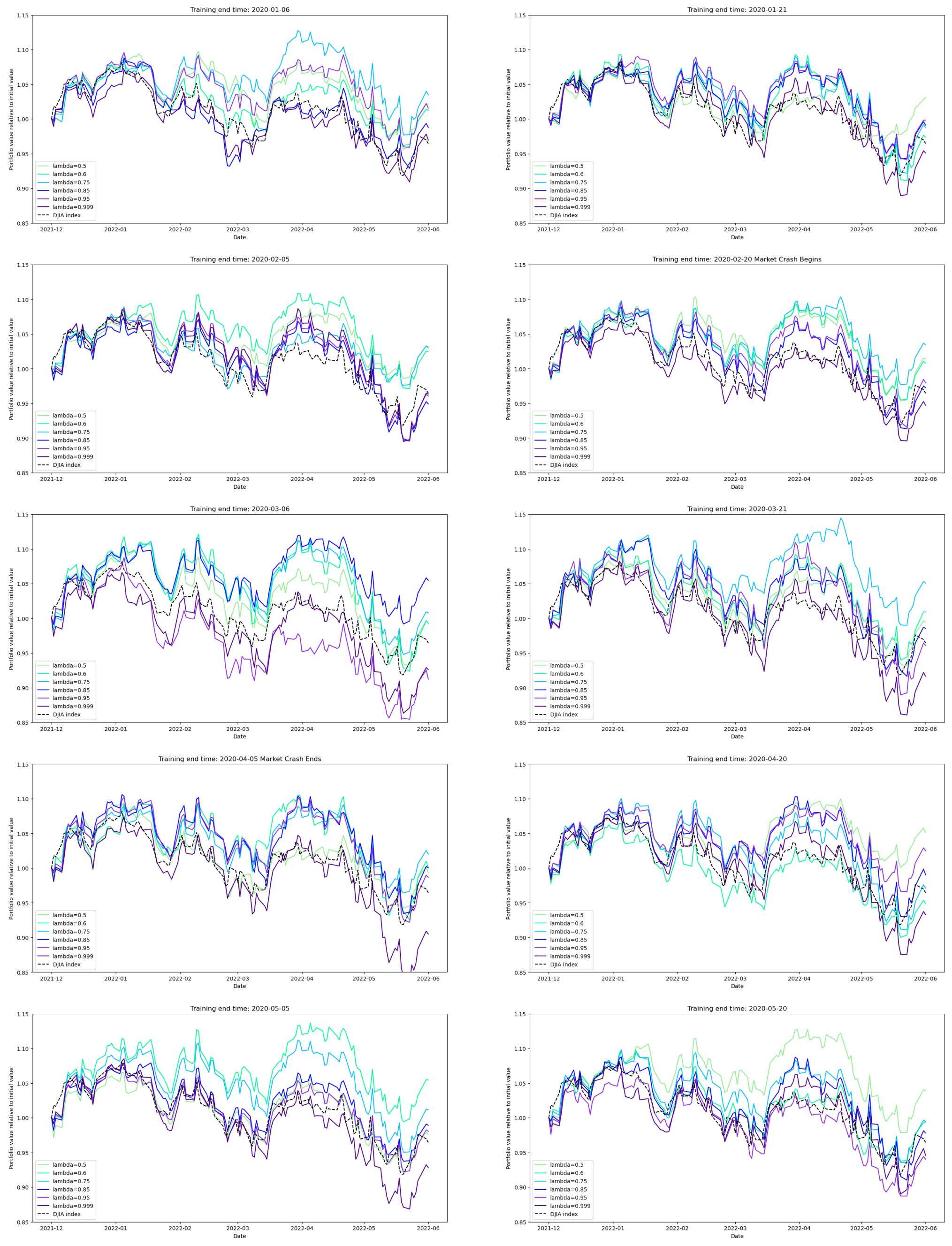

Fig. 3. **Plots of median-performing agents' portfolio value over time by training time and lambdas.** This plot shows the *naive* way of examining stock trading performance, which is to examine the time series of portfolio value. Here we present the models with a median rate of return in the sample of each (training-ending time, lambda) pair. One should pay special attention to changes in each agent's value curve, i.e., curves with the same color after the market crash data is added to training (training ending later than 2020-02-20). Agents with lambda 0.999 and 0.95 exhibit a great drop in performance after the first 15 days of market crash data are added into the training, while agents with lambda 0.5 don't do so until all 45 days of market crash data are added.

volatility. Thus, although some agents tend to have a high portfolio value, the current plot fails to quantify the variance level in the value. We need a new approach to control for this exogenous factor.

Therefore, we continue to analyze the alpha (α) and beta (β) coefficients of the sampled agents, as defined in Eq.2.17. By definition, α of an agent's portfolio measures its ability to generate excess return, while controlling for the risk premium from excess volatility. This allows us to solve the second caveat. Furthermore, unlike relative portfolio value, which is a time series, the alpha is merely a single statistic for each agent. We can conveniently plot and compare the average changes in α as market crash data is added to the training data, as demonstrated by the box plot in Figure 4. This addresses the first caveat.

We make a few observations regarding the changes in alpha values for agents trained with the market crash data. For agents with $\lambda = 0.999$ and $\lambda = 0.95$, we observe a drop in alpha from the training ending time of Feb 20 to March 06, when the first 15 days of market crash data get added to the training set. On the contrary, agents with smaller lambdas see drops delayed. For agents with $\lambda = 0.85$, we observe the decrease in average alpha when the second 15-day batch of market crash data is added, and for agents with $\lambda = 0.75$, the mean alpha does not drop significantly until the third 15-day batch of market crash data. For the lowest values of λ we experimented with, $\lambda = 0.5$ and $\lambda = 0.6$ agents do not demonstrate the dramatic downturn in mean portfolio alpha, even when all 45 days' data of the 2020 stock market crash is added to their training set.

These observations showcase that higher lambda values may lead to agents that are more sensitive to recent market changes. The drop in average alpha for agents with $\lambda = 0.999$ and $\lambda = 0.95$ after the inclusion of early crash data suggests these agents may be over-fitting to the recent market crash conditions rather than maintaining the strategy they learned that perform well under normal market conditions. On the contrary, agents with lower lambda values showing

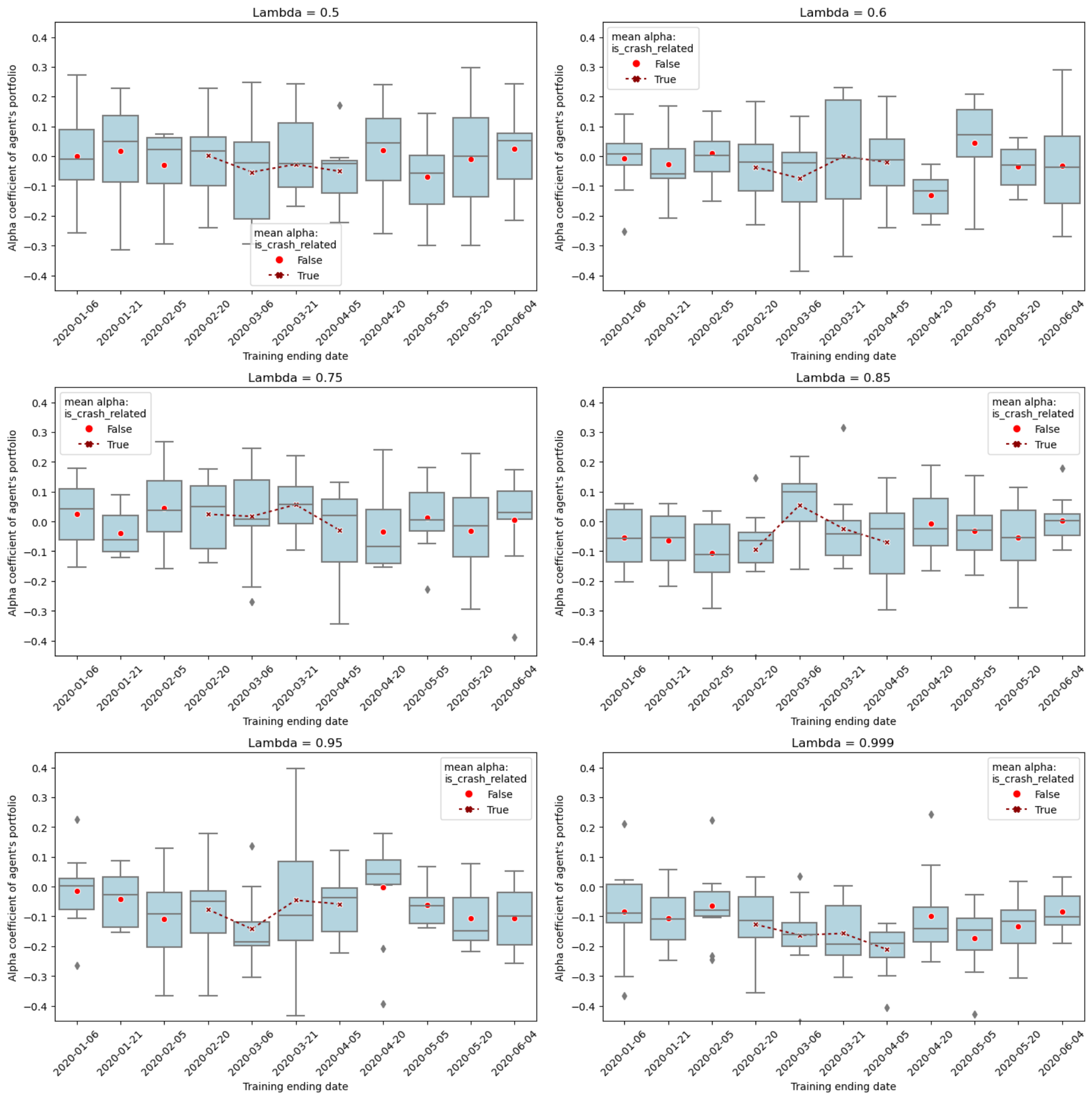

Fig. 4. **Box plot of alpha's of portfolios by agents trained with different ending dates by lambda values.** This plot demonstrates the changes in each stock-trading model's alpha when they are trained with different lambda values and when the market crash data gets added to the training set. Since the testing takes place with data in a normal market, a higher alpha means the model does better in preserving its learned trading strategy when market anomaly data gets added to the training set. The dotted line marks the dynamics of the average alpha as market crash data is added to training, for samples of models with different lambdas. We observe that the drop in average alpha happens for lambda 0.999 and 0.95 agents in the first 15-day batch of market crash, for lambda-0.85 agents in the 2nd batch, and for lambda-0.75 agents in the third batch. For lambda-0.5 and lambda-0.6 agents, we observe no significant drops in alpha. This might be indicative of smaller lambda values' effect to alpha preservation in abnormal training data, though we discuss the caveats of this experiment in Section 5.2.

delayed responses to the crash data might indicate a more conservative learning process. These agents seem to rely more on long-term trends rather than recent events, which could lead to a more stable, though potentially less reactive, performance in turbulent market conditions.

Nevertheless, we note that with a sample of only 10 models for each training-ending date, there are potential questions about the statistical significance of these findings. We will discuss this in Section 5.2.

5. Discussion

In this thesis, we have demonstrated the significant potential of Deep Reinforcement Learning (DRL) in revolutionizing investment science, particularly within the realm of stock trading strategies. Our investigation into the comparative performance of average and discounted reward agents, underpinned by the Proximal Policy Optimization (PPO) algorithm, reveals the potential of DRL to manage complex financial problems. However, it also highlights the challenges and returns in fine-tuning models to navigate the intricacies of the financial market.

5.1 Summary of findings

In experiments, we observed variations in performance, measured by portfolio rate of return, among the discounted-reward agents trained by different discount factors, γ. A significant portion of existing research within the intersection of DRL and finance has predominantly employed the discounted-reward frameworks, often defaulting to this setup without extensive exploration of alternatives. Our study challenges this norm by investigating the efficacy of average-reward agents in trading scenarios. Remarkably, our findings suggest that the average-reward trading agents not only deliver consistently high rates of return but also sidestep the complexities associated with selecting a γ. This simplification reduces the burden of hyperparameter tuning. It thus could make DRL more accessible and practical for a wider range of applications in finance, particularly for practitioners who may not have the extensive computing power to tackle the non-trivial task of hyperparameter tuning in DRL.

Concurrently, our investigation into lambda (λ) tuning for alpha preservation marks represents a step in understanding how to adjust average-reward reinforcement learning models for financial markets. Alpha, which measures a strategy's ability to outperform the market benchmarks, is a crucial metric for any trading model. Our experiments using the 2020 stock market crash demonstrated the potential to maintain an agent's alpha when abnormal market data is included in the training data. By adjusting for a lower λ, we can prevent the agents from catastrophically forgetting about the learned trading strategies in the event of market anomalies.

5.2 Future work

While this thesis presents significant explorations in the application of Deep Reinforcement Learning DRL to stock trading, it is crucial to acknowledge the limitations of our experiments and the scope for future research. One notable caveat of our study is the reliance on simplified data and the use of default hyperparameters, with the exception of λ. The real-world financial markets operate at a much finer granularity, often trading in seconds rather than days, and the dynamics at this scale can be vastly different from those observed in our experiments. Besides, as shown in Figure 2, we compared the portfolio return of our agents to the market portfolio of the DJIA stocks, i.e., the proportion of wealth invested in each stock is proportional to their market capitalization. To further assess the advantages of RL agents, future studies should consider using more complicated traditional strategies as the baseline models, such as the Black-Litterman model with historical data (Black and Litterman, 1992).

Moreover, the training process in our experiment may not fully resemble the application of RL in real-world stock trading. As Chong and Ng (2008) discussed, professional traders may use artificial technical indicators, such as Moving Average Convergence Divergence, to guide their predictions of price variations in the stock market. Such artificial indicators are not included in the training data of our experiments. Professional traders with access to substantial computing power may also engage in extensive hyperparameter tuning beyond what was explored in this study. The applicability of our findings about average-reward RL in such fine-tuned tasks remains an unanswered question. This highlights the need for further research that incorporates more complex market data and explores a broader range of hyperparameter optimizations to validate and potentially refine our findings.

For our second research question, our investigation into the impact of λ tuning on alpha preservation was constrained by limited computational resources. We only explore 6 λ values, cross-walked with training endpoints spaced out by 15-day intervals. Future studies should experiment with a more granular set of these parameters to quantify the effect of different λ values. Moreover, for each λ-endpoint pair, our random sample of models has a size of only 10, which may have impacted the statistical significance of our results. Such enhancements would provide a more robust quantitative analysis of λ's effect on the resistance of the average-reward agent to abnormal training data, offering clearer insights into the potential benefits and limitations of this approach.

Furthermore, we explored the 2020 stock market crash as an example of market anomalies. This market crash lasted for over one month, so we can detail how agents' behaviors evolve as this abnormal data gets added to the training set. We found that agents with lower lambdas preserve their alpha against longer periods of abnormal training data. However, it is worth questioning to what extent this crash is representative of broader market anomalies. To ensure the universality of our findings, future studies should either provide more extensive experiments on other market anomalies, or explore theoretical explanations of the effect of lowering lambda.

Looking ahead, future work in this field could address these limitations by leveraging more advanced computational resources to conduct experiments at a scale and complexity that more closely mirrors real-world trading conditions. Investigating the interplay between various hyperparameters, in addition to lambda, could uncover new dimensions of model optimization and performance enhancement. Moreover, a more granular approach to model training and evaluation, encompassing a wider range of market conditions and trading frequencies, would be instrumental in validating the efficacy of DRL models in diverse trading scenarios.

Acknowledgments

I would like thank my thesis advisor, Professor Shangtong Zhang, for providing me with detailed guidance in surveying reinforcement learning algorithms. I would also like to thank Professor Chen-Yu Wei for his valuable suggestions during the review of this thesis.

References

Black, Fischer and Litterman, Robert. (1992). Global portfolio optimization. *Financial Analysts Journal* **48**(5), 28–43.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie and Zaremba, Wojciech. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Buehler, Hans, Gonon, Lukas, Teichmann, Josef, Wood, B. Dan, Mohan, Baranidharan and Kochems, Jonathan. (2019, 3). Deep hedging: Hedging derivatives under generic market frictions using reinforcement learning. *Social Science Research Network*.

Chong, Terence Tai-Leung and Ng, Wing-Kam. (2008). Technical analysis and the london stock exchange: testing the macd and rsi rules using the ft30. *Applied Economics Letters* **15**(14), 1111–1114.

Evtimov, Ivan, Eykholt, Kevin, Fernandes, Earlence, Kohno, Tadayoshi, Li, Bo, Prakash, Atul, Rahmati, Amir and Song, Dawn. (2017). Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945* **2**(3), 4.

Fama, Eugene F. and French, Kenneth R. (2004). The capital asset pricing model: Theory and evidence. *The Journal of Economic Perspectives* **18**(3), 25–46.

Gravelle, Jane G. and Marples, Donald J. (2021, 2). Fiscal policy and recovery from the COVID-19 recession. *Technical Report* R46460.

Guan, Mao and Liu, Xiao-Yang. (2022). Explainable deep reinforcement learning for portfolio management: an empirical approach. In: *Proceedings of the Second ACM International Conference on AI in Finance*, ICAIF '21. New York, NY, USA: Association for Computing Machinery.

Jain, Abhinav, Patel, Hima, Nagalapatti, Lokesh, Gupta, Nitin, Mehta, Sameep, Guttula, Shanmukha, Mujumdar, Shashank, Afzal, Shazia, Sharma Mittal, Ruhi and Munigala, Vitobha. (2020). Overview and importance of data quality for machine learning tasks. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20. New York, NY, USA: Association for Computing Machinery. p. 3561–3562.

Kemker, Ronald, McClure, Marc, Abitino, Angelina, Hayes, Tyler and Kanan, Christopher. (2018, Apr.). Measuring catastrophic forgetting in neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence* **32**(1).

Kim, MyeongSeop, Kim, Jung-Su, Choi, Myoung-Su and Park, Jae-Han. (2022, 9). Adaptive Discount Factor for Deep Reinforcement Learning in Continuing Tasks with Uncertainty. *Sensors (Basel)* **22**(19), 7266.

Kirkpatrick, James, Pascanu, Razvan, Rabinowitz, Neil, Veness, Joel, Desjardins, Guillaume, Rusu, Andrei A., Milan, Kieran, Quan, John, Ramalho, Tiago, Grabska-Barwinska, Agnieszka, Hassabis, Demis, Clopath, Claudia, Kumaran, Dharshan *et al.* (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* **114**(13), 3521–3526.

Konda, Vijay R. and Tsitsiklis, John N. (2003, 1). OnActor-Critic algorithms. *Siam Journal on Control and Optimization* **42**(4), 1143–1166.

Kumar, Santosh, Kumar, Ankit, Singh, Kamred Udham and Patra, Sujit Kumar. (2023). The six decades of the capital asset pricing model: A research agenda. *Journal of Risk and Financial Management* **16**(8).

Li, Yuming, Ni, Pin and Chang, Victor. (2019, 12). Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing* **102**(6), 1305–1322.

Liu, Xiao-Yang, Yang, Hongyang, Chen, Qian, Zhang, Runjia, Yang, Liuqing, Xiao, Bowen and Wang, Christina Dan. (2022). Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance.

Ma, Xiaoteng, Tang, Xiaohang, Xia, Li, Yang, Jun and Zhao, Qianchuan. (2021). Average-reward reinforcement learning with trust region methods. *CoRR* **abs/2106.03442**.

Mahmood, A. Rupam, Korenkevych, Dmytro, Vasan, Gautham, Ma, William and Bergstra, James. (2018). Benchmarking reinforcement learning algorithms on real-world robots.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan and Riedmiller, Martin. (2013). Playing atari with deep reinforcement learning.

Perotto, Filipo Studzinski and Vercouter, Laurent. (2018). Tuning the Discount Factor in Order to Reach Average Optimality on Deterministic MDPs. In: *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Cambridge, United Kingdom.

Pitis, Silviu. (2019). Rethinking the discount factor in reinforcement learning: A decision theoretic approach.

Puterman, Martin L. (2014, 8). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Raffin, Antonin, Hill, Ashley, Gleave, Adam, Kanervisto, Anssi, Ernestus, Maximilian and Dormann, Noah. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research* **22**(268), 1–8.

Schulman, John, Moritz, Philipp, Levine, Sergey, Jordan, Michael and Abbeel, Pieter. (2015). High-dimensional continuous control using generalized advantage estimation.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec and Klimov, Oleg. (2017). Proximal policy optimization algorithms.

Sharpe, William F. (1964, 9). Capital asset prices: a theory of market equilibrium under conditions of risk. *The Journal of finance (New York. Print)* **19**(3), 425–442.

Silver, David, Huang, Aja, Maddison, Christopher, Guez, Arthur, Sifre, Laurent, Van Den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, Dieleman, Sander, Grewe, Dominik, Nham, John, Kalchbrenner, Nal, Sutskever, Ilya, Lillicrap, Timothy P., Leach, Madeleine, Kavukcuoglu, Koray, Graepel, Thore *et al.* (2016, 1). Mastering the game of Go with deep neural networks and tree search. *Nature* **529**(7587), 484–489.

Sorensen, Eric, Qian, Edward, Schoen, Robert and Hua, Ronald. (2004, 12). Multiple alpha sources and active management. *Journal of Portfolio Management - J PORTFOLIO MANAGE* **30**, 39–45.

Sutton, Richard S. (1988). Learning to predict by the methods of temporal differences. *Machine learning* **3**, 9–44.

Szepesvari, Csaba. (2010, 1). *Algorithms for reinforcement learning*. Morgan & Claypool Publishers.

Tadepalli, Prasad and Ok, DoKyeong. (1998). Model-based average reward reinforcement learning. *Artificial intelligence* **100**(1-2), 177–224.

Wei, Chen-Yu, Jafarnia-Jahromi, Mehdi, Luo, Haipeng and Jain, Rahul. (2021). Learning infinite-horizon average-reward mdps with linear function approximation.

Yang, Hongyang, Liu, Xiao-Yang, Zhong, Shan and Walid, Anwar. (2020). 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.