Multi-Agent Reinforcement Learning for Portfolio Management

M. Choi

TU Delft

Multi-Agent Reinforcement Learning for Portfolio Management

by

M. Choi

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday 29, August 2024 at 09:30 AM.

Cognitive Robotics

ROBECO
The Investment Engineers

Student number:	5401321	
Thesis duration:	November 13, 2023 – July 29, 2024	
Submission Date:	August 22, 2024	
Defense Date:	August 29, 2024	
Faculty:	Mechanical Engineering (ME)	
Department:	Cognitive Robotics (CoR)	
Supervisors:	Prof. Cosimo Della Santina,	TU Delft, Main supervisor
	Dr. Mustafa Mert Çelikok,	TU Delft
	Dr. Mike Chen,	Robeco B.V.
	Dr. Clint Howard,	Robeco B.V.
	Danny Huang,	Robeco B.V.

TU Delft

Acknowledgments

I would like to express gratitude to **Dr. Cosimo Della Santina** for approving this opportunity and being my supervisor and to **Dr. Javier Alonso-Mora** for being in my graduation committee.

Special thanks to **Dr. Mustafa Mert Çelikok**, who graciously accepted my request for supervision when I walked into his office and engaged in numerous discussions about deep learning, reinforcement learning, and multi-agent systems. His continuous support and inspiration through our weekly meetings were invaluable.

My interest in this work was sparked by one of Dr. Alonso-Mora's lectures on multi-agent robotic systems and reinforcement learning, which I found profoundly insightful. I am also grateful to **Dr. Mike Chen**, who proposed this project at the end of my summer internship at Robeco and encouraged me to explore this ambitious topic that bridges robotics and finance through the lens of intelligent decision-making. My sincere thanks go to **Dr. Clint Howard** and **Danny Huang** for their valuable feedback and critical insights, which greatly enriched this research. I would also like to extend my gratitude to **Dr. Frans Oliehoek** for his monthly insights into the theoretical aspects of multi-agent reinforcement learning.

Finally, I would like to thank my family, friends, and colleagues at Robeco for their support throughout this journey.

It has been a privilege to work on such an ambitious and fascinating project, and I thoroughly enjoyed the process, including overcoming numerous model crashes.

M. Choi
Rotterdam, August 2024

Abstract

Reinforcement learning (RL) is a powerful tool where the agents – or “robots” can learn from the environment based on their actions. Reinforcement learning approaches were found successful in combining predicting stock returns and portfolio allocation. Diversification is a critical element for achieving high portfolio returns with a lower level of risk. This work explores the application of reinforcement learning and multi-agent reinforcement learning (MARL) in portfolio management, emphasizing the applicability and increased portfolio performance via strategy diversification. A key contribution of this work is the development and evaluation of a MARL environment incorporating novel diversity measures, correlation, and total variation distance. The findings reveal that while fine-tuned single-agent RL models can demonstrate strong performance, roughly tuned MARL models with diverse agents reflecting a ”portfolio of portfolios” paradigm show improved action diversification and portfolio performances. The work also highlights the critical role of long-term robustness testing, algorithm- and problem-specific hyperparameter optimization, and the challenges of adapting MARL methods to financial contexts. This work contributes to the RL research in portfolio management by exploring the use of MARL in portfolio management and discussing the limitations and future work directions.

Contents

1 Introduction

Portfolio management (PM) is a fundamental responsibility for asset managers, who allocate capital on behalf of investors such as pension funds and insurance companies. The goal is to invest in a portfolio of assets such as stocks or bonds to achieve some predetermined objective like maximizing return and minimizing risk. Traditionally, this involves financial theories and fundamental analysis. However, integrating intelligent decision-making, the key pillar of robotics presents an innovative approach to these challenges. Recent advancements in reinforcement learning (RL) have been particularly promising, providing tools to optimize decision-making in portfolio management. RL agents can learn policies to dynamically adjust portfolios, aiming to enhance returns while managing risk, often quantified as portfolio volatility.

In portfolio management, the term *outperforming the market* usually refers to achieving (risk-adjusted) returns on investments that exceed a benchmark or index representing a broad market segment. For instance, if an investment portfolio earns a higher risk-adjusted return than a major market index, it is said to have outperformed the market. The benchmark used in this work is the Standard and Poor's 500, which is often called the S&P 500. It tracks the performance of 500 large publicly traded companies listed on the American stock exchanges.

Diversification is the method of allocating capital across a variety of assets to mitigate the impact of any single asset's risk on the overall portfolio. By diversifying, investors can achieve a more stable and robust financial outcome [49, 70]. Applying the principle of diversification to a multi-agent system in reinforcement learning, introducing diversity among agents' actions and policies may similarly enhance system robustness and performance. In a multi-agent environment, diverse actions may prevent the agents from converging to suboptimal behaviors and ensure a wider exploration of the state-action space. This may lead to discovering more effective portfolio strategies and better overall performance, much like how diversified investments can lead to more stable financial returns. In this thesis, the application of single- and multi-agent reinforcement learning in portfolio management is investigated, seeking to enhance the decision-making process in return prediction and asset allocation, potentially providing more robust solutions than benchmark and equally weighted portfolios.

1.1. Research Question and Goal

I aim to develop a robust multi-agent reinforcement learning portfolio management system to create diversified investment strategies inspired by real-world human portfolio managers [74]. The research questions are:

1. Single Agent Portfolio

RQ 1a. What is the proper training framework to develop a single-agent portfolio management strategy that the model outperforms the market during the out-of-sample period, specifically in terms of cumulative return and Sharpe ratio?

2. Strengthening the Strategy with Multi-Agent RL

RQ 2a. Can MARL models deliver higher risk-adjusted returns than a single-agent model?

RQ 2b. Will encouraging diversity between the actions deliver higher risk-adjusted returns?

The sub-questions are formulated to effectively address the research question 2. To answer the first question approaches from [73] are adapted and reproduced using single-agent Proximal Policy Optimization (PPO) models. For the second research question, the single-agent framework is extended to multi-agent Independent Proximal Policy Optimization (IPPO), followed by a series of experiments to evaluate its performance.

1.2. Main Contributions

In this thesis, I investigate specific applications of multi-agent reinforcement learning to the portfolio optimization problem. To the best of my knowledge, this thesis is the first study to apply Independent Proximal Policy Optimization (IPPO) agents to the portfolio optimization problem with portfolios with homogeneous sets of assets and continuous action spaces and, secondly, to investigate the effect of different diversity measures on the performance of IPPO agents in portfolio optimization. The primary contributions of this thesis are threefold: *Analysis of Single-Agent RL for portfolio management and evaluation*, *Investigation of Multi-Agent Reinforcement Learning for Portfolio Management*, and *Evaluation from Both Computer Science and Asset Pricing Perspectives*.

Analysis of Single-Agent RL for portfolio management and evaluation: This work successfully reproduces the baseline results of Sood et al. [73] and extends the evaluation by conducting long-term out-of-sample evaluation and action decomposition.

Investigation of Multi-Agent Reinforcement Learning for Portfolio Management: I developed a multi-agent portfolio management environment with IPPO agents controlling homogeneous portfolios with a continuous action space, building upon the concepts introduced by Jiang et al. [33] and J. Lee et al. [40]. Furthermore, I comprehensively examined the effect of diversity on the performance of IPPO agents in MARL portfolio optimization.

Evaluation from Both Computer Science and Asset Pricing Perspectives: The thesis integrates both financial asset pricing and computer science research methodologies, bridging the gap between these two fields and providing a balanced and comprehensive evaluation of RL for portfolio management studies. such as testing for a period longer than the training period. Previous work on RL for portfolio management problems usually falls into two categories: finance experts using RL to solve their problems and RL experts using finance as an application domain. These two groups have different expertise, and the two lines of work appear to have a gap. This thesis integrates both research methodologies in order to bridge the gap between RL research and finance approaches.

Contributions to further research and open-source projects: This thesis contributes to the multi-agent deep reinforcement learning research community. Two pull requests have been submitted to the BenchMARL library, an open-source MARL agent library, with one already merged[1], and another currently in progress[2] as of August 15, 2024. These contributions aim to aid and advance ongoing research in MARL, providing valuable resources and enhancements for the development and evaluation.

1.3. Financial Terms

An **asset** is a financial instrument or resource with economic value that can be bought and sold. A **portfolio** is a combination of multiple assets such as stocks, bonds, commodities, or other investments. **Returns** measure the financial value received from an investment over a specified period. **Simple (rate of) Return** represents the absolute change of an investment asset. For example, the investment of \$100 in an asset at time t and if it is valued at \$120 at time $T > t$, the simple return would be $0.2 = \frac{\$120}{\$100} - 1$.

1.4. Thesis Structure

- **Chapter 2** This chapter reviews existing work on reinforcement learning, multi-agent reinforcement learning, and portfolio management. It provides an overview of state-of-the-art techniques and applications, establishing the necessary context and background.

[1] https://github.com/facebookresearch/BenchMARL/pull/105
[2] https://github.com/facebookresearch/BenchMARL/pull/84

- **Chapter 3, Experimental Setup and Methodology:** This chapter details the methodology used in this research, including the software architecture, data and data processing, environment and agent design, the experimental setup, hyperparameters, and training frameworks. The focus begins with single-agent models to establish a baseline, as these models form the foundational units for the subsequent multi-agent approaches.
- **Chapter 4, Results:** Results of the experiments are presented, including model performance metrics, portfolio analytics, and action decomposition.
- **Chapter 5, Discussion:** Offers a reflective discussion on the techniques, the limitations of the models and experiments, and the broader implications of the research. It addresses challenges faced during the study.
- **Chapter 6, Conclusion:** Summarizes the key findings of the research, reiterates the contributions made, and synthesizes the results.
- **Chapter 7, Future Works:** Outlines potential avenues for future research, building on the insights gained from this study and suggesting ways to address identified limitations and explore new directions.

2

Preliminaries

This chapter reviews the relevant literature on reinforcement learning and portfolio management techniques, including both single-agent and multi-agent approaches. The chapter begins by discussing the fundamentals of reinforcement learning and traditional portfolio management strategies, followed by an examination of RL-based portfolio management methods. Finally, state-of-the-art RL and multi-agent RL techniques are presented. By exploring these approaches, this chapter aims to highlight the advancements and challenges in the field, providing a comprehensive overview of how modern RL techniques can be applied to solve the portfolio management problem.

2.1. Reinforcement Learning

Reinforcement Learning is a subfield of machine learning where an *agent* learns to make decisions through interactions with an environment, aiming to maximize reward. An *agent* is a computational entity that repeatedly perceives the environment, making decisions aiming for a maximum cumulative reward. The system outside the agent is called an *environment*, often modeled as a Markov Decision Process (MDP). Markov decision process formalizes a sequential decision-making process through state, action, and reward feedback loop [76].

Consider an MDP defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where:

- $\mathcal{S}$: the set of states,
- $\mathcal{A}$: the set of actions,
- P: the state transition probabilities where $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}$
- R: the reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, with immediate reward $r = R(s, a, s)$,
- γ: the discount factor $\gamma \in [0, 1)$ determines the weight of recent rewards relative to the future reward in an infinite time horizon problem between $t = 0$ and $t = \infty$. In finite-horizon objectives, $\gamma = 0$. As in many of the real-life cases, the notation will assume an infinite horizon objective.

An *agent* is a computational entity that repeatedly perceives the environment, making decisions aiming for a maximum long-term reward. The system outside the agent is often modeled as an *environment*. In each time step, the agent maps a state to the probabilities of consequences of choosing each possible action. The mapping from state to probability is represented as π, and $\pi(a|s)$ is the probability that $A_t = a$ given $S_t = s$. The agent's objective is to maximize the long-term cumulative (discounted) reward G in equation (2.1), not only the immediate reward R_t.

In each time step, the agent maps a set of state $\mathcal{S}$ to the probabilities of choosing each possible action $\mathcal{A}$. The policy, which maps the state to action, is represented as π, and $\pi(a|s)$ is the probability of selecting action a_t given s_t. The agent's objective is to find the optimal policy that maximizes the long-term cumulative (discounted) reward G in equation (2.1), not only the immediate reward R_t.

$$G_t = \sum_{t=0}^{\infty} \gamma^t R_{t+1} \tag{2.1}$$

2.1.1. Solving MDP

MDP provides a formal framework for modeling sequential decision-making problems. The optimal policy $\pi*(s)$ is defined as a policy π that maximizes the expected return, $\mathbb{E}_\pi$. $\mathbb{E}_\pi$ represents the expectation with respect to the distribution of trajectories generated by following policy π, starting from state s. Formally, the optimal policy $\pi^*(s)$ is defined as:

$$\pi*(s) = \arg\max_{\pi} \left(\mathbb{E}_\pi \left[G_t \mid S_t = s\right]\right). \tag{2.2}$$

State value Function (V^π)

The *state-value* function $V^\pi(s)$ for a policy π and state s is defined as the expected return when starting at state s and following the policy π onwards.

$$V^\pi(s) = \mathbb{E}_\pi \left[G_t \mid s_t = s\right] \tag{2.3}$$

The value function, 2.3 satisfies a recursion as shown in the equation 2.4, which is also called the Bellman Equation. It represents the relationship between the value of a state $V(s_t)$ and the successor state $V(s_{t+1})$, where the $s_{t+1} = s'$.

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s) \left[R(s',a,s) + \gamma V^\pi(s')\right] \tag{2.4}$$

Action value Function (Q^π)

The other function essential for solving MDP is the state-action value function, also known as the action value function or the Q -function. $Q^\pi(s,a)$ under policy π quantifies the expected cumulative reward when starting in state s, taking action a and then following policy π. It is expressed as:

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a\right] \tag{2.5}$$

Model-based reinforcement learning algorithms solve an MDP by finding an optimal policy π while learning the unknown transition $\mathcal{P} : \mathcal{S} \times \mathcal{A}$ and the reward $\mathcal{R}$ [23]. In this project, I focus on model-free reinforcement learning algorithms, which allow for solving MDP without explicitly modeling market dynamics. One of the fundamental components in model-free RL algorithms is the value function defined in equation 2.4, which is central to understanding the agent's long-term expected rewards.

There are three major approaches to model-free RL: Value-based methods that try to learn V or Q functions and then recover a policy from them, Policy-based methods that bypass this and try to optimize a policy directly from interaction, and Actor-Critic methods that combine both approaches. Value-based or action-value algorithms [64] like Q-learning and SARSA focus on determining the optimal values associated with states or state-action pairs, guiding the agent toward decisions that maximize cumulative rewards. In this thesis, I will focus on actor-critic algorithms.

2.1.2. Policy Gradient Methods

Policy gradients optimize the policy directly. A common approach to do so is by using a parameterized policy π_θ and optimizing its parameters with gradient-based methods.

$$\pi_\theta^{t+1} = \pi_\theta^t - \alpha \nabla_\theta J(\theta) \tag{2.6}$$

The objective is to learn the mapping from s to a. The policy parameters are learned based on the gradient of a scalar performance measure $J(\theta)$, with respect to the parameter θ. With direct parameterized policy optimization, policy gradient agents can explore continuous action spaces.

The performance objective of stochastic policy gradient in Sutton et al. [76], rephrased by [72] is :

$$\begin{aligned} J(\pi_\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s,a) R(s,a) da ds \\ &= \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta} \left[R(s,a)\right] \end{aligned} \tag{2.7}$$

$\rho(s)$ above denotes state distribution. Policy gradient algorithms adjust the parameters θ in the direction of the gradient $\nabla_\theta J(\pi_\theta)$:

$$\begin{aligned}\nabla_\theta J(\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) Q^\pi(s,a) da ds \\ &= \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta} [\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)]\end{aligned} \tag{2.8}$$

Deterministic policies associate each state with a single action, ensuring that, given the same state, the action is consistently identical. This is particularly advantageous in scenarios with a narrow answer space, where a fixed action corresponds to a specific state, as observed in games like chess, where optimal moves remain constant for a given board configuration, with an action distribution having $\sigma = 0$. On the other hand, **stochastic policies** map states to a probability distribution of actions $\pi : \mathcal{S}\times\mathcal{A} \to [0,1]$, allowing the consideration of uncertainty in the environment. In such policies, actions are drawn from a distribution characterized by mean (μ) and standard deviation (σ). For instance, in poker, where the probability of success may vary based on the opponent's move, a stochastic policy proves beneficial in adapting to uncertain conditions. In general, the policy in policy gradient is stochastic (i.e., $\pi(a \mid s, \theta) \in (0,1)$, for all s, a, θ as they are easier to optimize and denoted by $\pi_\theta : \mathcal{S} \to P(\mathcal{A})$. Silver et al. [72] extends the stochastic policy gradient into a deterministic policy gradient theorem with direct mapping from state to action with policy $\mu_\theta : \mathcal{S} \to \mathcal{A}$. ME Hall K

2.1.3. Actor-Critic Methods

Actor-critic methods integrate principles from both value-base d(TD) and policy-based(Policy gradient) learning methods [76].

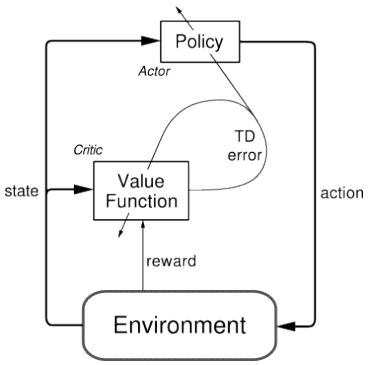


Figure 2.1: Actor-Critic Structure [76]

In actor-critic methods, the Actor learns to adjust parameters $\boldsymbol{\theta}$ in policy π_θ with policy gradient to update actions, and the critic evaluates the action by estimating value function parameterized by w, $Q_w(s,a) \approx Q_\pi(s,a)$ like in Figure 2.1. It starts with computing temporal difference error for action value:

$$\delta_t = r_t + \gamma Q_w(s', a') - Q_w(s,a). \tag{2.9}$$

Then the temporal difference δ_t is used to update the Q function parameters w:

$$w \leftarrow w\alpha_w \delta_t \nabla_w Q_w(s,a). \tag{2.10}$$

2.1.4. Deep RL

Traditional methods of solving RL problems involved storing Q values for each state-action pair, where the agent updated these values based on its chosen strategy (e.g., Q-learning or SARSA) in a tabular form to approximate the value function. The linear approximation approaches using Q-table becomes impractical when working with higher dimensional decision-making scenarios. In addition to state and action space, approximating the parameters for Q_θ, V_θ functions, and policies π_θ are more complex in large and continuous problems. The evolution from traditional approaches to deep reinforcement learning using neural networks [52] marked a significant advancement in solving RL problems. Deep neural networks (DNN) utilize nonlinear activation functions to approximate hidden relationships in data.

Neural networks are composed of several layers: input, hidden, and output layers. The hidden layers consist of connected neurons or nodes that apply weights and biases during computation. Weights and biases are the learnable parameters of the neural network, and the algorithms learn these parameters to optimize based on certain loss functions [24]. The capabilities of neural networks enabled the RL agents to learn abstract and high-dimensional representations and approximate value functions and parameters better. This allowed for a more efficient and scalable representation of state and action values, especially in complex scenarios where traditional Q-tables fell short. [52] and [53] successfully integrated CNN for learning feature representations of high dimensional states and to approximate Q with Deep Q Network (DQN). The DQN agent could play video games like Atari better than a human player by integrating DNN with Q-learning.

Deep Actor-Critic Method: Proximal Policy Optimization (PPO) [68]
The predecessor of PPO, the Trust Region Policy Optimization(TRPO) [66] aims to update a new policy in a way that it does not deviate too far from the previous policy. The probability ratio between the current and old policy is the probability of taking an action a_t at state s_t with policy π_θ with respect to the previous policy.

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{2.11}$$

If the divergence between the old and current policy, $r(\theta)$ is higher than 1, the action a_t at state s_t is more likely in the current π than π_{old} and if $0 < r(\theta) < 1$, the action a_t at state s_t is less likely in the current π than π_{old}.

The KL divergence between (policy) distributions can be expressed as:

$$D_{\text{KL}}(\pi_\theta \mid \pi_{\theta_{\text{old}}}) = \sum_i \pi_\theta(i) log \frac{\pi_\theta(i)}{\pi_{\theta_{\text{old}}(i)}} \tag{2.12}$$

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}(\theta_k, \theta) \tag{2.13}$$

The objective of TRPO is to optimize the following:

$$\max_\theta \left(\mathbb{E}_{s_t,a_t}\left[r_t(\theta)A\left(s_t,u_t\right)\right]\right) \quad \text{subject to} \quad \mathbb{E}_{s_t,a_t}\left[\text{KL}\left(\pi_{\theta_{old}},\pi_\theta\right)\right] \leq \delta \tag{2.14}$$

A constraint on Kullback–Leibler divergence was introduced to ensure this so that the difference between the two policies does not exceed a threshold δ, which complicates the computation. Instead, PPO introduces clipping directly in the objective function.

$$\mathcal{L}^{CLIP}(\theta) = \min\left(r_t(\theta)A_t^{\text{adv}}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t^{\text{adv}}\right) \tag{2.15}$$

So that the final objective can be a lower bound of the probability distribution based on the ratio ϵ, the introduction of clipping balances between performance and comprehension will make PPO converge better while avoiding policy updates that are too large.

2.2. Multi-Agent RL

Achievements in single-agent reinforcement learning serve as a precursor to the active exploration of multi-agent reinforcement learning (MARL). MARL is a sub-field of RL with several learning agents interacting with each other. Deepmind's AlphaStar [82] achieved grandmaster-level proficiency in the intricate real-time strategy game StarCraft II. OpenAI Five [58] has demonstrated capabilities in navigating the intricate dynamics of multiplayer games such as Dota 2, successfully incorporating elements of incomplete information, cooperation, and competition. Beyond gaming scenarios, MARL has transcended into achieving human-level performances in various domains, including manufacturing scheduling [18], and applications in transport system [71, 89].

In important model in MARL is the multi-agent generalization of an MDP, known as the Multi-agent MDP [10]. Multi-Agent MDP can be expressed as a stochastic game with N agents [46], expressed as a tuple: $(\mathcal{S}, \mathcal{A}_1, \ldots, \mathcal{A}_N, P, R_1, \ldots, R_N, \gamma)$. $\mathcal{A} = \mathcal{A}_1 \times \ldots \mathcal{A}_N$ is the joint action space of agent $i \in N$, and R_i is their corresponding reward function $R^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$. The transition is expressed as $P : \mathcal{A}_1 \times \ldots \times \mathcal{A}_N \times \mathcal{S} \mapsto \mathbb{P}(\mathcal{S})$.

Distributed Training Decentralized Execution (DTDE)
Modeling a multi-agent system in a distributed manner may seem like the most straightforward approach considering the nature of the independent agents. In a fully distributed training and execution setup, learning of the policy and action selection of each agent is fully segregated from that of other agents. Agent updates its policy based on individual observations specific to the agent. There is no centralized controller, and agents do not have access to global information. With the individual state action space, distributed methods scale poorly with an increasing number of agents and complicate the learning [27]. Furthermore, the distributed scheme suffers from non-stationarity as the individual agents are unaware of the other agents' actions and their impact on the environment when making decisions for their future actions, which could lead to non-convergence.

2.2.1. Independent MARL

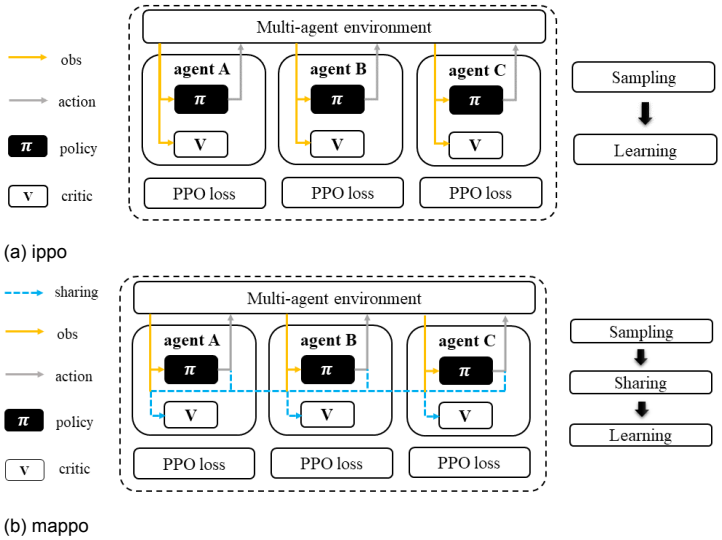


Figure 2.2: Visualization of IPPO and MAPPO [30]. The key difference is in knowledge sharing between the agents.

Independent algorithms assume independent learning and assume the other agents as part of the environment [11]. Each independent learner does not consider other agents' existence and treats the environment as stationary. The assumption makes the local policy of the learners stationary; however, the performance of this policy globally is affected by other players' actions. In an example of a rock-paper-scissors game, if a player assumes the environment is stationary, this means the opponent is not adaptive. Then they may commit to a deterministic policy (e.g., always rock, always best-respond to the previous game, etc.). However, in reality, an adaptive opponent would learn to exploit this behavior. Therefore, in this scenario, the optimal policy would always be stochastic since, in reality, the environment is non-stationary due to the opponent's adaptation. Local deterministic policy will never be optimal when you assume independent learning. Thus a stochastic approach is required to find an optimal policy [55]. Independent Q-learning (IQL, [78]) is a MARL extension of Q-learning where each agent has its Q function. Another example of an independent multi-agent algorithm is Independent DDPG, a natural extension of DDPG to a multi-agent system. policy network and Q network. Each agent has its own DDPG network with an observation and action space that samples data. It is capable of learning independently without requiring knowledge sharing.

2.2.2. Diversity in Multi-Agent Systems

Diversity in multi-agent systems refers to the variation in behavior or strategies [86] or policies [8] among agents within the environment. Introducing diversity allows agents to explore a broader range of strategies [59], leading to better generalization and adaptability. In cooperative environments, diverse agents can complement each other, resulting in improved team performance, while in competitive scenarios, diversity can promote innovation and adaptability [41, 88]. Understanding and measuring diversity is key to unlocking these benefits. Multi-agent systems' diversity further depends on the degree of coop-

eration between the agents, environment-agent interaction style, and additional information availability from the environment. McKee et al. [50] explored the relationship between generalization and diversity in MARL. They found that generating diverse training enhances agent performances on new and unseen levels and introduced an environment-agnostic measure of behavioral diversity. The most common measure in multi-agent systems with physical robots is by using Euclidean distance. Li et al. [41] optimized the diversity via regularized L1-norm that promotes learning sharing between the agents.

2.3. Portfolio Management

Investors aim to maximize future returns by selecting assets with high expected returns and optimally allocating these assets within a portfolio. Consider constructing a portfolio of n assets. Assuming a full investment scenario where the weights of the portfolio sum to 1, $\sum_{i=1}^{n} w_i = 1$. If r_i denotes the return on asset i, the return of a portfolio R_p with N assets is a weighted combination of the constituent assets' returns r_i.

$$R_p = \sum_{i=1}^{N} w_i r_i \tag{2.16}$$

2.3.1. Portfolio Optimization

The challenge in portfolio management is to balance two conflicting objectives: maximizing the expected value of portfolio returns while minimizing risk, often quantified by the standard deviation of the returns. The mean-variance model, introduced by Markowitz [49], is the fundamental mathematical framework in portfolio optimization. It aims to find an optimal set of weights that minimizes volatility for a given level of risk under the assumption that an investor prefers a portfolio with lower risk given the same expected return. It shows that investors can reduce the overall risk of the portfolio by investing in various assets. The intuition behind this is that the overall portfolio variance is reduced when assets with low or negative correlations are combined. Diversification lowers the impact of any single asset's poor performance, thus enhancing the stability and predictability of returns.

2.3.2. Asset Pricing

Asset pricing models aim to estimate the required or expected rate of return of assets by understanding why assets have different average returns and identifying the factors driving changes in returns [14]. Studies focus on determining the risk premium, which is the expected return of an asset over the risk-free rate as compensation for bearing risk. In traditional asset pricing models, such as the Capital Asset Pricing Model (CAPM) Lintner [45], Mossin [56], and Sharpe [69], which is built on the Markowitz framework, the expected return of any asset is equal to the return on the risk-free asset plus the asset's market beta multiplied by the market risk premium,

$$E\left[R_{i,t}\right] = R_{f,t} + \beta_i \left(E\left[R_{m,t}\right] - R_{f,t}\right). \tag{2.17}$$

where the security's beta is given by

$$\beta_i = \frac{\mathrm{Cov}\left(R_{i,t}, R_{m,t}\right)}{\mathrm{Var}\left(R_{m,t}\right)}. \tag{2.18}$$

Let $R_{i,t}, R_{f,t}$, and $R_{m,t}$ represent the returns of security i, the risk-free asset, and the market portfolio, respectively, in period t. The expectation operator is denoted by $E[\cdot]$. The term $E\left[R_{m,t}\right] - R_{f,t}$ is the market risk premium and the covariance between the return of asset i and the market return during period t is given by $\mathrm{Cov}\left(R_{i,t}, R_{m,t}\right)$ while $\mathrm{Var}\left(R_{m,t}\right)$ denotes the variance of the market return in the same period [6]. More advanced models, such as the Fama-French three-factor model [22], incorporate multiple risk factors. Similarly, asset pricing studies focus on identifying factors that explain the risk premium [12].

2.4. RL for PM

Machine learning techniques help unravel complexity in financial signals by disentangling complex nonlinear associations, reducing degrees of freedom, and adeptly handling redundant variations. They

provide a systematic approach to tasks such as identifying predictive signals or timing the market, making portfolio choices, and risk management [25, 63]. Studies found predictive signals related to stock returns [12, 25, 81] using deep learning techniques, such as neural networks. Moreover, Gu et al. [26] used an Autoencoder model for risk-adjusted return prediction, and Heaton et al. [28] used deep neural networks for optimal portfolio selection. Reinforcement learning finds diverse applications in financial markets, from utility theory, dynamic asset allocation, and consumption to derivatives pricing and hedging [5, 65]. The advancement of deep neural networks and deep reinforcement learning has prompted a surge in studies exploring their integration into financial markets. However, unlike game-playing or robot control domains, where RL has demonstrated tremendous success, the financial market poses unique challenges. The singular nature of the market allows no opportunity to create multiple games. Developing realistic synthetic financial markets is a subject of its own. Furthermore, the financial market's inherent volatility and non-stationarity, coupled with a low signal-to-noise ratio, complicate the identification of market dynamics. Historical market information often proves irrelevant to its present and future states, rendering the problem inherently stochastic [31].

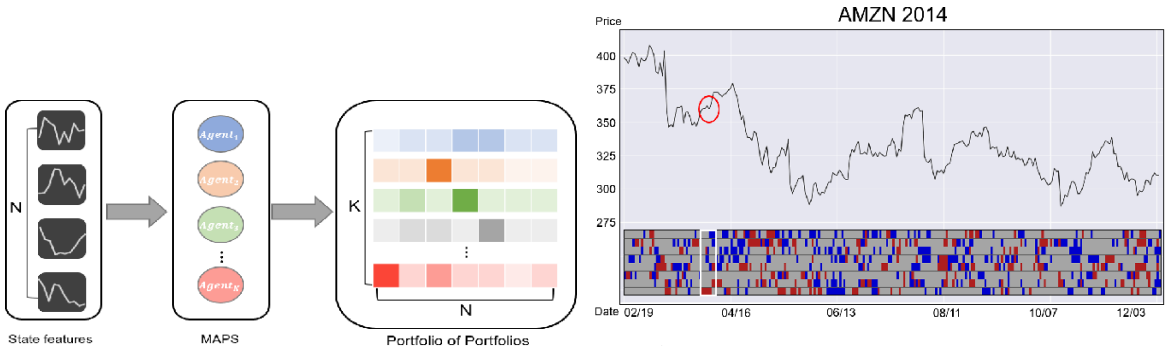
Portfolio allocation decisions with the aim of outperforming the market is one of the primary tasks of asset managers. RL application in portfolio management has been explored due to its ability to combine both return prediction and portfolio optimization. Neuneier [57](1997) utilized Q-Learning to optimize asset allocation, showing the adaptability of RL in addressing the complexities of portfolio decisions. With deep reinforcement learning, it became possible to solve problems with larger state-action space and applications have been extensively explored for its integration into portfolio management strategies, as observed in the work by [2, 33, 43, 60, 83, 84, 85, 87].

2.4.1. MARL for PM

Traditional single-manager funds tend to exhibit net exposures associated with the manager's particular strategy. To avoid the concentration of highly correlated ideas and strategies, human investors started allocating funds to investment vehicles that do not depend on the capability of a single investment manager [38]. Multi-manager platforms are investment organizations that collectively operate multiple specialized fund managers and strategies as a unified entity. Within these platforms, individual units are assigned local performance responsibilities. The decision-making process within multi-manager funds can either be centralized or decentralized, and the investment approach may be coordinated or operated independently. They are structured as all-weather investments, aiming to generate consistent absolute returns and appealing risk-adjusted returns regardless of the market conditions[74]. Before the breakthrough of deep reinforcement learning by [53], J. W. Lee et al. [39] (2007) proposed to incorporate a multi-agent system in portfolio management problems using four Q-learning agents to divide and conquer the market. Two agents specialized in picking the buy signal, and the other two specialized in holding. Their actions were driven by the objective of maximizing the portfolio value. This study presents the multi-agent approach, in which each agent has its specialized capability and knowledge.

J. Lee et al. [40] introduced a cooperative multi-agent framework where each agent functions as an independent portfolio manager, formulating its unique portfolio. Figure 2.3a describes the design of the problem, which is to have a "portfolio of portfolios" by making each K agent a portfolio manager of its own. The agents share the price information and share a common global reward while consequently optimizing the local policy to maximize local returns through discrete actions on how many shares to to long, neutral, and short.

To ensure the diverse actions between the agents, the study introduced a global training loss function that penalizes the correlation of actions between the agents. Additionally, [75] proposed to use two other diversity metrics for portfolio management: entropy [35] and effect number of bets [36]. Figure 2.3b illustrates how agents behave differently given the same shared state about the price information. Furthermore, the authors compared their strategy with well-known market factors like momentum and also compared with the market benchmark, the Russell 3000 index. Their experiments with 12 years of out-of-sample back tests outperformed all existing strategies. It's noteworthy to mention that while the authors achieved impressive results, replicating their experiment poses challenges due to the lack of clear presentation of hyperparameters, such as the number of stocks assigned to each agent and other implementation details.



(a) Visualization of Multi-Agent Portfolios

(b) Different market timing among the agents

Figure 2.3: J. Lee et al. [40]

3

Experimental Setup and Methodology

3.1. Empirical Settings

The choices made around the empirical settings are designed to enhance the model's learning process and to address the research question. I have adapted approaches from both computational sciences and finance.

3.1.1. Data Choices

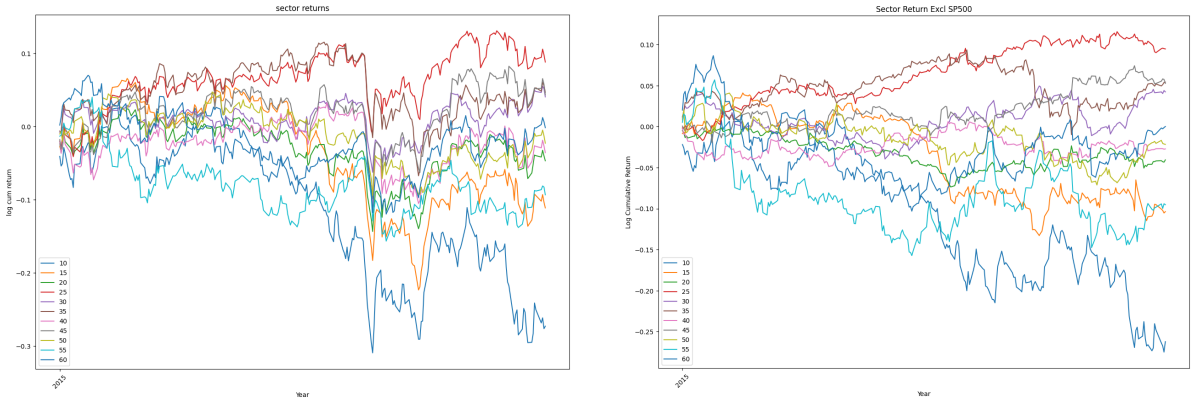
In reinforcement learning studies, a common practice when selecting investment instruments is to work with 10-30 different stocks [33, 37, 43, 84, 87]. The action space of RL agents grows exponentially with the number of stocks, which forces these studies to limit the scope of their portfolios. While this approach may be practical for individual investors, it introduces a significant limitation: the lack of diversification in assets. For example, a single volatile event from one company can disproportionately influence the model's performance, exposing it to higher risk from the lack of diversification. Moreover, the selection of stocks can have a significant influence on the outcome of the model. In contrast, it is a standard approach for empirical asset pricing studies to adopt a cross-sectional analysis that evaluates thousands of stocks, comparing their relative performances at a single point in time. Using sector indices enables leveraging the benefits of both approaches. Sector indices represent the performance of 11 industries, such as technology, healthcare, or energy. The indices are market-cap weighted, meaning larger companies within the sector have a proportionally greater influence on the index's performance. It offers a broader perspective on market movements while working within a limited number of instruments. Sector indices contribute to stable policy learning by reducing individual event volatility and offering general insights into market trends. However, constructing a long-only portfolio with sector indices presents challenges in deviating from the market index (S&P 500), potentially restricting excess returns but ensuring a more robust model. Thus, I used sector indices of the S&P 500.

Data Preprocessing

I use data between 2006-2013 for training, 2014 for validation, and 2015-2023 for out-of-sample testing. This data split approach differs from the standard data split approach in the machine learning field, where the majority of the data is typically used for training. Instead, prior studies on machine learning in asset pricing, such as Gu et al. [25] and L. Chen et al. [12], employed out-of-sample testing periods that are equal to or longer than the training period. Financial markets constantly evolve, and strategies that worked in 2005 may become irrelevant by 2009. Therefore, extended out-of-sample testing is crucial to prevent overfitting and to ensure the model's relevance in diverse market conditions. While empirical asset pricing studies typically use paywalled data from academic or commercial sources such as CRSP[1] and Compustat[2], publications focused on computer science aspects [1, 3, 13, 73] often employ return data from yfinance[3]. The reason for this preference is likely the emphasis on accessible

[1] https://www.crsp.org/
[2] https://wrds-www.wharton.upenn.edu/pages/grid-items/compustat-global-wrds-basics/
[3] https://pypi.org/project/yfinance/

open-source data and reproducibility within the machine learning community. However, a primary criticism of yfinance data is its low completeness, particularly when dealing with individual stocks that have undergone listing changes such as delisting and stock splits. Despite this, for index data that do not undergo such activities, yfinance provides complete data for the selected period. Therefore, this study uses yfinance library to leverage its accessibility and ensure the reproducibility of the research. The tickers on yfinance used for the experiments were *SP500TR* (S&P 500), *GSPE* (Energy), *SP500-15* (Materials), *SP500-20* (Industrials), *SP500-25* (Consumer Discretionary), *SP500-30* (Consumer Staples), *SP500-35* (Health Care), *SP500-40* (Financials), *SP500-45* (Information Technology), *SP500-50* (Communication Services), *SP500-55* (Utilities), and *SP500-60* (Real Estate).



(a) Cumulative log returns of each sectors

(b) Cumulative log returns of each sectors after subtracting the market

Figure 3.1: Comparison of Raw return vs. Market return

To ensure that the model learns the sector signal itself rather than the broader market movement, the market S&P 500 returns have been subtracted from the sector index returns. Figure 3.1 shows that the raw returns of the indices move in tandem with each other and the market. However, after subtracting the market signal, the sector-specific returns exhibit more distinct patterns. Therefore, the return of S&P 500 was subtracted from the returns of sector indices as an input of the model.

3.1.2. Assumptions

Two key assumptions when of the environment for the problem are the following:

- **Full Liquidity:** Assumes that the market is efficient. The liquidity of all market assets is sufficiently high, enabling immediate execution of each trade at the last price when an action is placed during the back-test period.
- **No Market Impact:** Assumes that the trading action made by the agent exerts no influence on the market dynamics (i.e., increasing the price of the asset) during the back-test period.

3.2. Portfolio Management Problem Setup

The portfolio management problem is formulated as a Markov Decision Process. The design of the experiment aims to model the step-by-step determination of the optimal weights to maximize the expected reward. The problem is represented as a discrete-time MDP and is transformed into an optimal control problem wherein the system under control is a portfolio comprising various investment components. The control involves adjusting the weights assigned to each asset.

3.2.1. State Space

The main component of the state space is the log return of the sector indices. With the closing price of an asset at time t as p_t, the one period return is defined as $r_t = \frac{P_t - P_{t-1}}{P_{t-1}}$.

Using percentage changes for returns may appear straightforward, but it displays asymmetry. Logarithmic returns, on the other hand, are symmetric on both the upside and downside, i.e., 10% gain

followed by 10% loss resulting in a net change of 0%, making comparison between returns across different periods easier. Furthermore, log returns can be added over time instead of multiplications when calculating cumulative returns.

Like the setting of Jiang et al. [33], the observation state includes past T day market information.

Observation state at t includes the weight of each asset in the portfolio of the agent based on $\mathbf{a}_{t-1}$ and past $T = 60$ days' historical sliding returns of N assets. As new market information is updated each day, t, the updated information for the day t is fed into the agent via the step function 1 as observation, along with the Differential Sharpe Ratio [54] reward.

$$\begin{bmatrix} w_{t-1,1} & w_{t-1,2} & \cdots & w_{t-1,N} \\ \rho_{1,t} & \rho_{2,t} & \cdots & \rho_{N,t} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1,t-T+1} & \rho_{2,t-T+1} & \cdots & \rho_{N,t-T+1} \end{bmatrix} \tag{3.1}$$

The 2-D matrix (3.1) is flattened and concatenated with volatility indicators $\left[vol_{20}, \frac{vol_{20}}{vol_{60}}, VIX_t\right]$, making the observation vector $\mathbf{s_t}$ that is compatible with MLP layers. vol_{20} is the 20-day rolling window standard deviation of the daily S&P500 index returns. vol_{60} is the same measure with 60-day window and VIX_t is the standardized value of the VIX index as in [73].

3.2.2. Action Space

The RL portfolio manager agent outputs the vector of action $\mathbf{a}_t = \left[a_t^1, a_t^2, \ldots, a_t^N\right]^T$ with $a_t^i \in [-1, 1]$ for all $i = 1, 2, \ldots, N$ at time t. The softmax function is applied to all $\mathbf{a}$ at a time t such that a new rebalanced portfolio weight vector $\mathbf{w}_t$ for the period between step t and $t + 1$ as:

$$w_t^i = softmax(a_t^i) = \frac{\exp(a_t^i)}{\sum_{j=1}^{N} \exp(a_t^j)} \quad \text{for} \quad j = 1, 2, \ldots, N \tag{3.2}$$

such that

$$\sum_{i=1}^{N} w_{t,i} = 1. \tag{3.3}$$

3.2.3. Reward

The objective of the agent is to maximize the risk-adjusted return. While the Sharpe Ratio is a commonly used measure for risk-adjusted returns, it is calculated over the long term and cannot be directly converted into a reward per time step. To address this limitation, [54] proposed the Differential Sharpe Ratio to represent the Sharpe Ratio as an additive sum of single-step reward. RL for PM studies [60, 73] used it as a reinforcement learning agent reward. First, the exponential moving average of the Sharpe ratio is formulated as:

$$S_t = \frac{A_t}{K_t (B_t - A_t^2)^{1/2}} \tag{3.4}$$

with $A_t = \frac{1}{t}\sum_{i=1}^{t} R_i$ and $B_t = \frac{1}{t}\sum_{i=1}^{t} R_i^2$ $K_t = (\frac{t}{t-1})^{1/2}$, which recursively estimates the exponential moving average of the returns and standard deviation of returns on timescale $\frac{1}{\eta}$. S_t is expanded to first order in η:

$$S_t \approx S_{t-1} + \eta D_t|_{\eta=0} + O\left(\eta^2\right) \tag{3.5}$$

Where D_t is the Differential Sharpe Ratio:

$$D_t = \frac{\partial S_t}{\partial \eta} = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{\left(B_{t-1} - A_{t-1}^2\right)^{3/2}} \tag{3.6}$$

with

$$A_t = A_{t-1} + \eta \Delta A_t \tag{3.7a}$$
$$B_t = B_{t-1} + \eta \Delta B_t \tag{3.7b}$$
$$\Delta A_t = R_t - A_{t-1} \tag{3.7c}$$
$$\Delta B_t = R_t^2 - B_{t-1} \tag{3.7d}$$

With $\eta = 252$ trading days in one year, the Differential Sharpe Ratio D_t between periods $t-1$ and t is used as a reward for the agent, similar to [73].

Table 3.1: Key components of the Single Agent Model

Key Components	Attributes
Observation $\mathbf{s}_t$	Previous period $t-1:t$ portfolio weights Daily return of past 60 days Volatility information
Action $\mathbf{a}_t$	New portfolio weights for period $t:t+1$
Reward r_t	Differential Sharpe Ratio

3.2.4. Environment

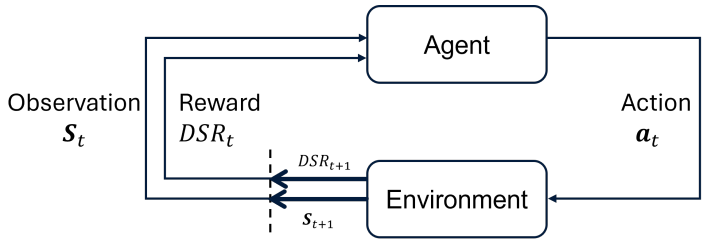


Figure 3.2: Visualization of Single-Agent PM Framework

The environment simulates the stock market based on the methodologies described in [33, 47]. Observations are provided from historical returns, and the agent's actions consist of the new portfolio weights for the subsequent time $t+1$. environment's step function (Algorithm 1) provides market information to the agent. The environment receives action a_{t-1} and returns state s_t and reward r_t for the next time step. By interacting with the environment, the agent iteratively collects experience and learns a policy to maximize rewards. The observation based on historical market data is configured using the Gymnasium [80] library.

Algorithm 1 Step Function in single-agent portfolio management environment

Input : Action at time $t-1$, a_{t-1} from the policy
Output: Set of State and Reward at time t, $\mathbf{s}_t$, r_t, and *done* = False
$t \leftarrow t+1$ Set *done* to True if $t \leq T-1$
while *not done* **do**
  $\mathbf{w}_{t-1} \leftarrow$ softmax($\mathbf{a}_{t-1}$)
  Calculate Differential Sharpe Ratio at time t with $\mathbf{w}_{t-1}$ and $(\mathbf{ret}_t)+1$
  $r_t \leftarrow$ Differential Sharpe Ratio$_t$
  $\mathbf{s}_t \leftarrow$ Concatenate[$\mathbf{w}_{t-1}$, log return array between $[t-60, t]$, and volatility index]

3.3. Single-Agent PPO

There are various implementations of PPO beyond the original paper by Schulman et al. [68], including PPO2 from the same authors. Reproducibility of PPO has been challenging due to numerous implementation details, making it difficult to implement the algorithm from scratch, as highlighted in the article

"37 Implementation Details of PPO" [32]. Multiple RL for PM studies [2, 43, 48, 85] have used PPO To ensure standardized and reproducible research, I use the Stable Baselines3 [62] version of PPO, with agent hyperparameters from Sood et al. [73], shown in the Table 3.2.

Table 3.2: Stable Baselines3 PPO Agents hyperparameters

Hyperparameters	Values
Gamma	0.9
GAE Lambda	0.9
Clip Range	0.25
Learning Rate	3e-4 annealed to 1e-5

3.3.1. Rolling Window Training

Sood et al. [73] employs a unique agent training methodology. They train a population of agents with different random seeds for each training window and continue this process iteratively, as described in Figure 3.3. During the first training window, five agents are initialized with different seeds, and the agent with the highest mean average return during the validation period of that window is selected as the initial agent for the next training window. This process is repeated until the 10th training window, resulting in the generation of 50 agents with different random seeds.

This method can be considered a form of multi-agent learning, as it involves multiple agents. However, the agents do not receive feedback from the actions of other agents during learning or within the environment. This training approach can also be seen as a type of population-based training or evolutionary algorithm, in the sense that it involves optimizing a population of models by periodically selecting the best-performing individuals and exploring new hyperparameters based on them, and the best learner survives and is used in the next generation of training. This training method is used to establish the single-agent baseline results.

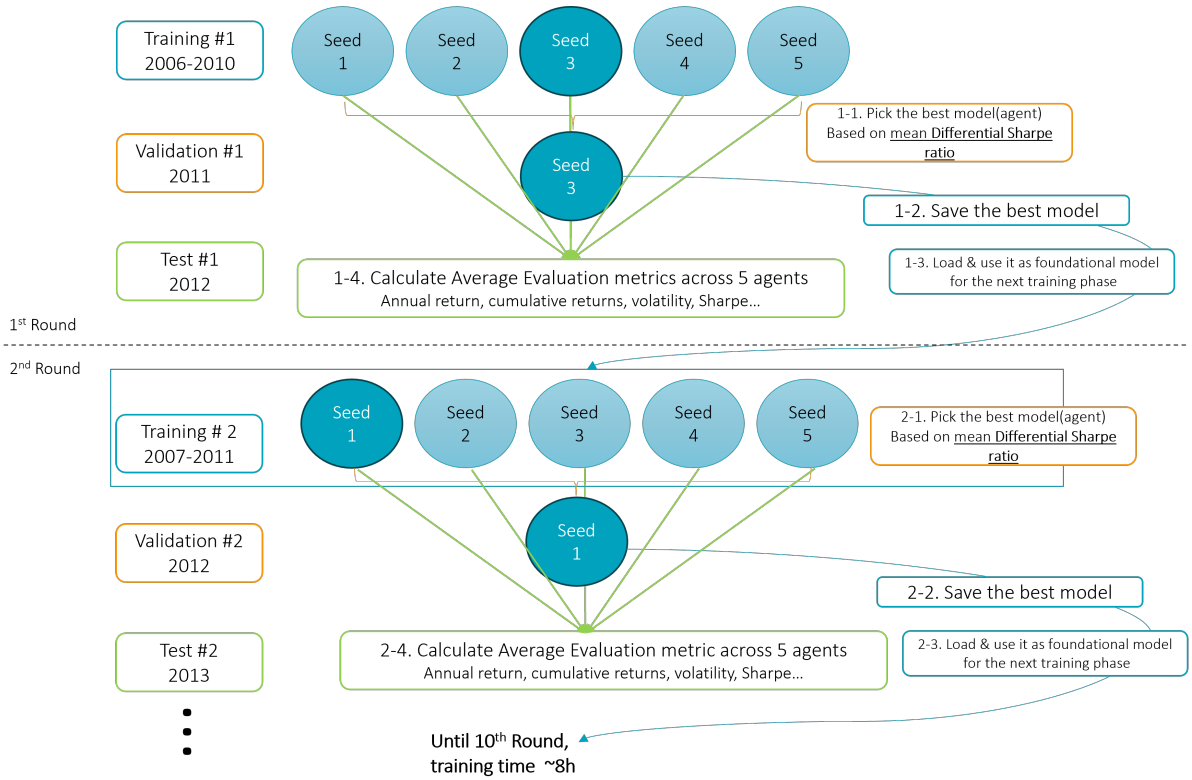


Figure 3.3: Visualization of multiple seed training in [73]

3.4. Single-Agent to Multi-Agent

In this section, the single-agent reinforcement learning framework is extended to a multi-agent setup to investigate the impact of introducing diversity among agents and assess whether this enhances overall strategy robustness. Specifically, the multi-agent approach aims to diversify strategies among agents by incorporating diversity measures that encourage agents to behave differently from each other.

3.4.1. Diversity Measure

The diversity penalty, similar to the Global loss in J. Lee et al. [40], is reflected in the environment instead of the agent's loss function. In addition to using correlation as a diversity measure, Total Variation (TV) distance is employed as an alternative metric to further explore and enhance portfolio diversity by capturing differences in the distribution of returns.

TV distance is defined as:

$$TV(A,B) = \frac{1}{2}||A - B||_1 \tag{3.8}$$

Where A and B are two probability distributions. TV distance measures the difference between two probability distributions and captures the extent to which the two distributions diverge from each other and is used in diversity in MARL studies [41]. Portfolio weights that sum up to 1 can be considered probability distributions, making TV distance a meaningful measure for assessing how similar or different portfolios are.

The diversity penalty is incorporated into the reward calculation as follows:

$$r_{t,k} = r_{t,k}^{init} \cdot \kappa_t \tag{3.9}$$

where the multiplier κ_t includes the diversity penalty and the weight of the penalty lambda.

$$\kappa_t = \begin{cases} 1 - |DP_t \cdot \lambda| & \text{if } r_{t,k}^{init} \geq 0 \\ |DP_t \cdot \lambda| & \text{if } r_{t,k}^{init} < 0 \end{cases} \tag{3.10}$$

For example, if $r_{t,k}^{init} = 5$ and $\lambda = 0$, the final reward $r_{t,k}$ remains the same as the initial reward, which is the differential Sharpe ratio. When the absolute value of the diversity penalty DP_t is close to 1, it indicates high similarity in actions, thereby reducing the final reward. If $\lambda = 1$ and $DP_t = 0.9$, the agent receives a reward of 0.5, indicating high penalty and when $\lambda = 1$ and $DP_t = 0.1$ the final rewards becomes 4.5. Conversely, if $r_{t,k}^{init} = -5$, $\lambda = 1$, and $DP_t = 0.9$, the initial reward becomes -4.5. If the similarity of the actions is low, such that $DP_t = 0.1$, the final reward is -0.5, thus resulting in a higher reward.

J. Lee et al. [40] introduced a global training loss function that penalizes the correlation of actions between agents, defined as:

$$GLoss_k = \sum_{k=1, i \neq j}^{K} \left[Corr(\eta_k, \eta_j) \right] \tag{3.11}$$

where η_i represents the agent's confidence level in its action for the subsequent time step. The independent Q-learning agents in their study optimize a loss function that is a weighted sum of the local Q-learning loss $LLoss_k$ and the global loss:

$$Loss_k = (1 - \lambda)\, LLoss_k + \lambda GLoss_k \tag{3.12}$$

PPO is capable of achieving higher RL reward than Q-Learning [68] and is suited for environments where continuous action spaces are required. The loss objective of PPO is mathematically more complex than that of DQN, as it includes parameters such as the divergence between the old and current policy and a clipped objective. Due to this complexity, incorporating the global loss directly into the PPO loss function is more challenging. Consequently, the diversity penalty was reflected within the environment rather than integrated into the PPO loss function. Furthermore, [50] states that the diversity in the environment itself is a good source of diversity. The diversity penalty was directly scaled and applied to the reward in the environment. This approach was chosen because the reward differential Sharpe ratio has a wider range, with values potentially exceeding 100. Given that the range of correlation is

$[-1, 1]$, Total Variation distance is $[0, 1]$, and cosine similarity is $[0, 1]$, to increase the influence of the penalty while encouraging diverse actions, the absolute diversity penalty, scaled by the weight for the penalty λ, is subtracted from 1 and finally multiplied with the initial differential Sharpe ratio reward of each agent, as shown in Equation 3.9. This modification from [40] aligns with the goal of leveraging PPO's advantages in handling continuous actions while effectively encouraging diverse actions among the agents.

3.4.2. Multi-Agent Environment

In the step function in multi-agent environments, all agents receive the same set of market information, as they are assumed to have access to a homogeneous set of assets. The procedure for receiving actions from the agents and presenting observations is identical to that of the single-agent environment. After the observation and calculation of the DSR of each agent, the diversity of the actions of the agents is measured via the selected diversity measure. The global diversity measure between the agents is calculated and used as a scaling penalty factor for the given differential Sharpe ratio. The penalty, scaled by the weight λ of the diversity penalty, is then applied to the final reward.

Algorithm 2 Step Function in multi-agent portfolio management environment with K agents

Input : Action at time $t-1$, a_{t-1} from the policy
Output: State and Reward at time t, $\mathbf{s}_t$, r_t, and *done*
$t \leftarrow t+1$
Set *done* to True if $t \leq T-1$
while *not done* **do**
  for *Agent*$_k$ *in* $\mathbf{K}$ **do**
    $\mathbf{w}_{t-1,k} \leftarrow \text{softmax}(\mathbf{a}_{t-1,k})$
    Calculate Differential Sharpe Ratio at time t with $\mathbf{w}_{t-1,k}$ and $\mathbf{ret}_{t,k}$
    $r_{t,k}^{init} \leftarrow$ Differential Sharpe Ratio$_{t,k}$
    $\mathbf{S}_{t,k} \leftarrow$ Concatenate$[\mathbf{w}_{t-1,k}, \rho[t-60, t]$, Volatility info$]$
  $DP_t \leftarrow \sum_{k=1, i \neq j}^{K} \left[\text{diversity\_measure}\left(\mathbf{w}_{t,k}^{init}, \mathbf{w}_{t,j}^{init}\right)\right]$; `// Calculate Diversity Penalty`
  $\kappa_t \leftarrow$ Eq. 3.10
  $r_{t,k} \leftarrow r_{t,k}^{init} \cdot \kappa_t$; `// λ: diversity penalty weight`

To experiment with agent diversity and enhance performance through inter-agent feedback, I used the PettingZoo [79] library to extend the single-agent RL environment to a multi-agent RL environment. PettingZoo allows the creation of a MARL environment in Gymnasium style. The environment was implemented as a parallel environment where the timings of observation and reward between the agents are synchronous. PettingZoo was selected over alternatives such as Ray RL [42] and TorchRL due to its comprehensive documentation for creating customized environments and its compatibility with NumPy, thus eliminating the need for transitioning to Jax.

3.4.3. Multi-Agent Learning

Independent algorithms assume independent learning and assume the other agents as part of the environment. Each independent learner does not consider other agents' existence and treats the environment as stationary. Actions of the other agents are considered as part of the environment.

The IPPO algorithm [17] uses clipped PPO loss (equation 2.15) as an objective, which stabilizes training by limiting policy updates. Generalized Advantage Estimation (GAE) [67] is employed to efficiently teach the agent from local observations.

Algorithm 3 IPPO Training

Initialize policy parameters for k agents $\{\pi_{\theta_k}\}_{k\in\{0,1,\ldots,K-1\}}$
for *iteration* $= 0, 1, \ldots n_episodes$ **do**
  Run policies $\{\pi_{\theta_{\text{old},i}}\}_{i\in\{0,1,\ldots,N-1\}}$
  Collect $(\mathbf{s}, \mathbf{a}, \mathbf{r})$ from experience
  for *epoch* $= 0, 1, \ldots$, *num_iters - 1* **do**
    for *minibatch* $= 0, 1, \ldots$, *minibatch_size - 1* **do**
      for $k = 0, 1, \ldots, K-1$ **do**
        Compute Value function GAE A_k^{adv}
        $\mathcal{L}_k \leftarrow \mathcal{L}^{CLIP}(\pi_{\theta_{\text{old},k}}, \pi_{\theta_k})$ (2.15)
        Update π_{θ_k} by minimizing $\mathcal{L}_k$ via stochastic gradient ascent with Adam
      end
    end
  end
  $\{\pi_{\theta_{\text{old},k}}\}_{k\in\{0,1,\ldots,K-1\}} \leftarrow \{\pi_{\theta_k}\}_{k\in\{0,1,\ldots,K-1\}}$
end

The non-independent MAPPO algorithm [88], which uses shared observations and shared inputs to the value function, has outperformed independent MARL algorithms in cooperative tasks like Google Research Football and the StarCraft Multi-Agent Challenge. Its strength lies specifically in scenarios where maximizing shared input to the value function is critical. Therefore, for non-independent multi-agent algorithms to be effective, each agent should have a distinct state space and carefully curated shared information. However, in the current multi-agent portfolio management scenario, all agents operate with the same set of assets and thus have a significant overlap in information. This overlap means that the agents already receive a high degree of shared, centralized information, which may reduce the need for further specialization in their state spaces.

MARL Software Implementation
While stable-baselines3 is a well-established library with extensive documentation and community contributions, there is currently no equivalent for multi-agent reinforcement learning. BenchMARL[7] stands out due to its design, which is made for experiments, and compatibility with customized environments, a feature that is more challenging than other libraries. BenchMARL is based on TorchRL[9], which allows for the use of a PettingZoo wrapper to convert a custom PettingZoo environment into a TorchRL environment compatible for use with BenchMARL agents, enhancing its flexibility and usability.

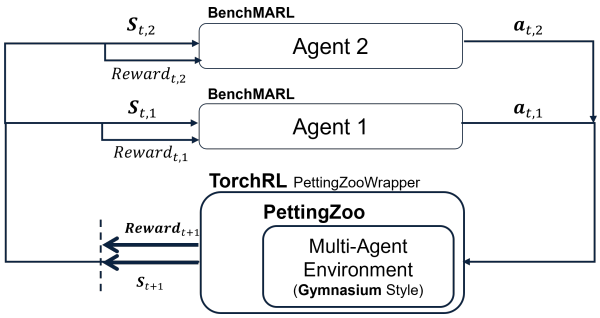


Figure 3.4: Visualization of Multi-Agent PM Framework

3.4.4. Hyperparameters

Hyperparameter Tuning
Reinforcement learning models are highly sensitive to hyperparameter choices, which can significantly influence their final performance. Hyperparameters are often problem-specific, which makes it a non-intuitive task [20, 29, 44]. For hyperparameter tuning, a hyperparameter sweep from Weights and

Biases (W&B) [4] like in Figure 3.5 is used to train models over a grid of hyperparameters. Multiple models are trained to test various combinations of hyperparameters, focusing on those that show significant effects. Although more advanced hyperparameter tuning methods, such as random search or Bayesian optimization, exist, grid search was used in this study for its simplicity in roughly finding the optimal hyperparameters and experimenting with different diversity weights. The purpose of this sweep was to identify the set of hyperparameters that yielded the best performance during the out-of-sample validation period, which is the year 2014. The model's performance was evaluated using the mean of the differential Sharpe ratio (DSR) reward, which will be explained in the following section, from the start to the end of the evaluation period, as in the equation 3.13.

$$\text{Evaluation Period Mean Reward} = \frac{1}{T}\left(\sum_{t}^{T}(dsr_t)\right) \tag{3.13}$$

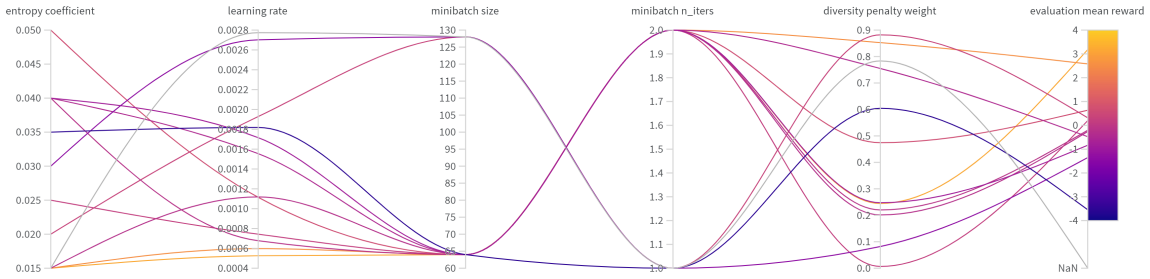


Figure 3.5: Hyperparameter sweep example

Table 3.3: IPPO hyperparameters in scope for the sweep

Category	Hyperparameters	Values in Scope
Optimizer(SGD)	Learning Rate	[0.0001,0.001]
	Minibatch Size	[64, 128]
	Minibatch number of iterations	[2,4,8]
Nerual network	Number of layers	2, 3
	Size of layers	64, 128
	Random seeds	0,1,2,3
PPO	Entropy coefficient	[0.002,0.005]
Diversity	Type of diversity penalty	Correlation, TV Distance
	Weight of the penalty	[0,1]

The primary choice for the hyperparameter ranges is based on [20, 32, 68]. For learning, the optimizer learning rate and its scheduling were included in the tuning scope. Studies by [21] and [4] found that annealing the learning rate for PPO improved performance. Neural network hyperparameters were also considered, including the number of layers and units per layer. The neural network's architecture significantly impacts the model's ability to learn and generalize, making it essential to tune these parameters carefully. During the initial experiments, an improper choice of the entropy coefficient severely damaged model performance, similar to observations in [20]. Consequently, the entropy coefficient was included in the sweep objectives. To assess the impact of diversity weight in the models, the weight λ of the diversity penalty κ, acting as a scaling factor to the differential Sharpe ratio, was also included in the sweep. Values ranging from 0 to 1, with increments of 0.1, were tested. Additionally, different random seeds, $[0, 1, 2, 3]$, were included in the scope to see its effect, as deep RL is also sensitive to random seeds.

[4] https://docs.wandb.ai/guides/sweeps

3.4.5. Contribution to Open-Source Project

I implemented a systematic hyperparameter optimization approach utilizing Weights & Biases sweeps, integrated with Hydra configuration management [5] and the BenchMARL library. This framework facilitated an extensive automatic exploration of various hyperparameter configurations. Upon reviewing the existing BenchMARL library, I identified a lack of a structured framework for hyperparameter optimization. To address this, I submitted a pull request [6] with the hyperparameter sweep code and detailed implementation and usage instructions. Additionally, the developed software implementation framework for connecting MARL agents with the custom environment in section 3.4.3.1 has been contributed [7] to the same library. As a result, I provided a more efficient hyperparameter optimization method and straightforward experimentation with customized environments to the MARL research community.

3.5. Evaluation

During the out-of-sample testing period, trained policies are evaluated with four portfolio metrics. Initially, a trained policy is retrieved from a checkpoint file. This policy is then deployed deterministically into an environment where sector returns during the testing period serve as observations. Subsequently, the agent outputs the actions based on the observation. These actions are interpreted as portfolio weights for the testing period and are used to evaluate the performance of the portfolios constructed. The metrics used for analyzing portfolios include absolute return, Sharpe ratio, maximum drawdown, and turnover, which are explained in more detail below.

Absolute Cumulative Return

The rate of return of a portfolio R_p with N assets is a weighted combination with weight w_i of the constituent asset's returns r_i.

$$r_{p,t} = \sum_{i=1}^{N} w_t^i r_t^i \tag{3.14}$$

Cumulative return between period t and T is calculated by compounding the single period returns over time. With rate of return $r_{p,k}$ at time k, the cumulative return r_k between period t and T is given by:

$$r_{t \to T} = \prod_{k=t+1}^{T} (1 + r_k) - 1 \tag{3.15}$$

Sharpe Ratio

Sharpe Ratio [70] assesses the risk-adjusted performance of an investment portfolio. It quantifies the excess return per unit of volatility σ_p, indicating how well an investment return performs for the associated risk. Excess return is calculated as the difference between the portfolio's return R_p and the risk-free rate R_{rf}, which is the return from investing in an asset that is considered risk-free, like the interest rate or a government bond yield. A higher Sharpe ratio indicates a higher risk-adjusted return. The relationship between the volatility of the portfolio with N number of assets σ_p and portfolio weight w_i, volatility between the assets $\sigma_{i,j}$ is:

$$\sigma_p^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} w_i \sigma_{i,j} w_j \tag{3.16}$$

With the portfolio volatility, the annualized Sharpe ratio is:

$$\text{Sharpe Ratio} = \frac{\mathbb{E}[R_p - R_{rf}]}{\sigma_p} \tag{3.17}$$

[5] https://hydra.cc/docs/intro/
[6] https://github.com/facebookresearch/BenchMARL/pull/105
[7] https://github.com/facebookresearch/BenchMARL/pull/84

Maximum Drawdown
Maximum Drawdown measures the greatest percentage drop in a portfolio's value from its highest point (peak) to its lowest (trough). It represents the peak-to-trough loss experienced before a recovery to a new high occurs.

$$\text{MDD}(\%) = \frac{\text{Trough Value} - \text{Peak Value}}{\text{Peak Value}} \tag{3.18}$$

Portfolio Turnover
Portfolio turnover, as described in [16], measures the total absolute value of rebalanced weights across N assets over T − 1 trading periods, normalized by the total length of trading periods.

$$\text{Turnover} = \frac{1}{T-1} \sum_{t=1}^{T-1} \sum_{i=1}^{N} \left(\left| w_{t+1}^{i} - w_{t}^{i} \right| \right) \tag{3.19}$$

4

Results

4.1. Single Agent Baseline

4.1.1. Environment and Algorithm Validation

To validate both the environment and the RL algorithm, the PPO agent is initially trained on a simplified learning case where the future returns of the assets, $\mathbf{r}_{t+1}$ are provided as the state $\mathcal{S}_t$, and the reward is defined as the logarithm of the portfolio return. This experiment addresses two key questions: whether the environment correctly presents observations to the agent and whether the agent effectively learns the simplest strategy. In this full foresight scenario, the optimal strategy for the agent is to allocate all weights to an asset with the highest expected return at $t+1$, so that the agent can reach the maximum possible return and immediate RL reward. The validation experiment demonstrates that the agent can identify the maximum return scenario, but only when the action space is adjusted to a broader range, e.g., $[-300, 300]$, rather than a narrower range, e.g., $[-1, 1]$. This adjustment is necessary due to the softmax function applied to the action outputs. In a case where the agent produces an extreme action, where one asset gets 1 and the rest 10 gets -1, e.g. $\mathbf{w}_t = [1, -1, -1 \ldots -1]^T$, the softmax function normalizes the actions resulting results in a weight of 42% for the asset with the action output of 1, rather than the desired 100%. Scaling the action space has the same effect as adjusting the temperature T of the softmax function, which modulates the scaling of action probabilities.

$$\frac{\exp \frac{z_i}{T}}{\sum_j \exp \frac{z_j}{T}} \tag{4.1}$$

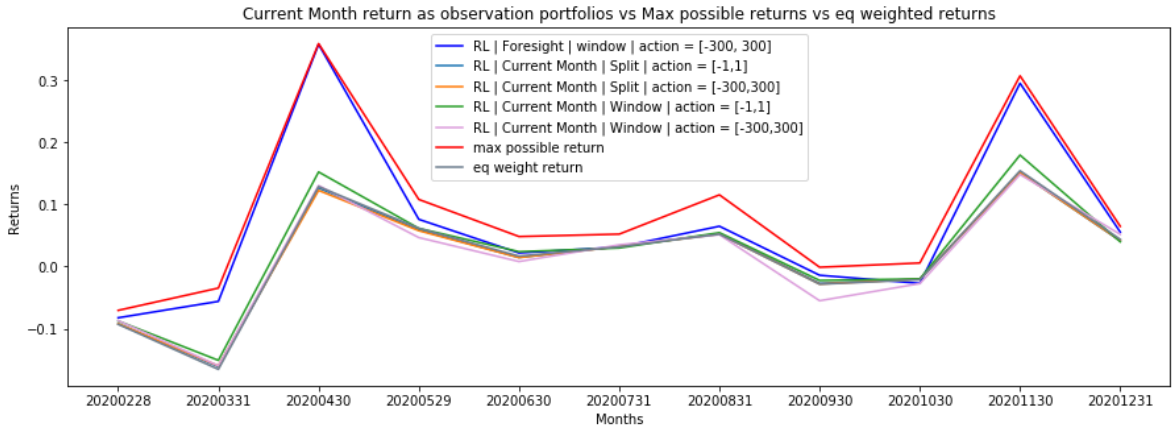


Figure 4.1: Comparison of the foresight PPO agent's performance with an action space of $[-300, 300]$ and the maximum possible return.

The results confirm that the PPO agent matches the maximum possible return with the expanded action space, thereby validating the environment's effectiveness. For all subsequent experiments without foresight, the action space was reverted to $[-1, 1]$ to constrain weight changes and potentially enhance the Sharpe ratio by limiting large weight changes.

4.1.2. Single-Agent Replication Result

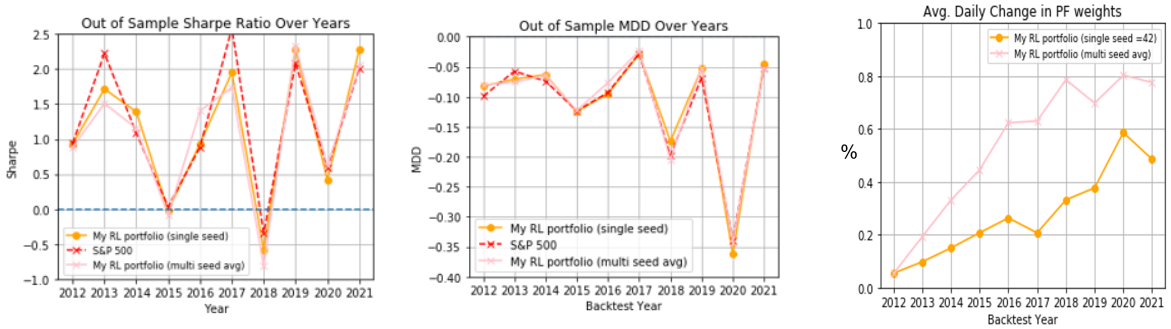
The single-agent model is reproduced to establish a baseline, serving as a reference point for comparison when extending to multi-agent models. The empirical setting for the reproduction is taken from [73] as detailed in Table 3.2. The original study and the replication Setting A train 50 agents over 10 rolling windows using 5 different seeds. While the seeds used in the original study are unspecified, Setting A uses the seeds 0, 1, 2, 3, and 4. Setting B trains 10 agents over 10 rolling windows using a single seed, 42, in this experiment. The metrics in Table 4.1 are calculated following the same approach as the original study, averaged across 10 testing periods after each training window. The exception is the maximum drawdown, which is reported as the most negative value observed in any period.

Type	**Sood et al. [73]**	**Replication Setting A**	**Replication Setting B**
Training Method	Window Training	Window Training	Window Training
Seed	Unknown 5 seeds	0,1,2,3,4	42
Cumulative Returns	0.1195	0.1252	0.1236
Sharpe	1.1662	1.07834	1.1305
Max Drawdown*	-0.3296	-0.3465	-0.3625
Annual Volatility	0.1249	0.1374	0.1386

Table 4.1: Single-Agent reproduction, sample period: 2006-2021, evaluation: [2012-2021]

Table 4.1 presents the portfolio statistics of the agents. The cumulative returns are similar, showing less than a 5% difference relative to the original study. The Sharpe ratio of the replicated models deviated by no more than 8% relative to the original paper. These differences are to be expected, as the original seeds used in the paper are unknown.

However, when decomposing the results by year, notable differences emerge between the models with Setting A (pink line in 4.2) and Setting B (orange line in 4.2), particularly in the average daily change in portfolio weights. As illustrated in Figure 4.2 c, there is a significant gap in the average daily change in portfolio weights between the two settings. The observed similarity in the Sharpe ratios, as shown in Figures 4.2 a and b, and potentially the results in Table 4.1 may stem from the fact that the experiment involves around 11 subcomponents of the market benchmark, S&P 500, where the returns of the assets are largely correlated to the benchmark. Additionally, the softmax normalization function also suppresses the significant deviations of portfolio statistics of the agents, as it constrains asset weights within the range of [5%, 42%], as mentioned in the previous section.



(a) Sharpe ratio over time

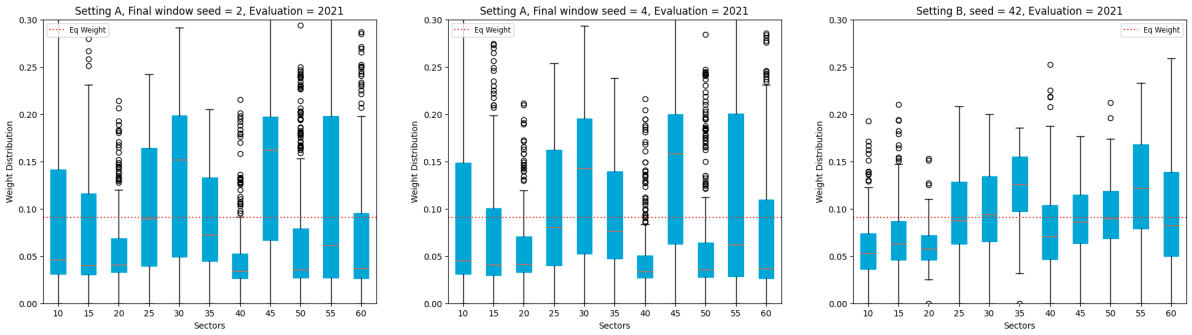
(b) MDD over time

(c) Portfolio turnover

Figure 4.2: Three images side by side, resized to one-third each

When decomposing the weights of the portfolios derived from settings A and B, it is possible to

see the difference in seed. In the training of models for Figure 4.3a and 4.3b, the policy followed the training described in Section 3.3.1, where the policy with the best seed, among the seeds $[0, 1, 2, 3, 4]$ was carried forward into the subsequent rolling window training. The difference in the training of the agents in Figure Figure 4.3a and 4.3b are in the seed of the 10th window. The agent in Figure 4.3a was trained with seed $= 2$ on the 10th window, while the agent in Figure 4.3b used seed $= 4$. With a small difference in seeds during the final training round, it is possible to see that the results of the policy in Figure 4.3a and 4.3b showed minimal variation to each other in weight distribution. This indicates that the policy, which remained consistent up until the 9th rolling window, was largely unaffected by the change in seed during the final round of training. Figure 4.3c shows the results from an agent trained with the same seed (42) across all 10 rolling window training rounds. Notably, there is significant variation in weight distribution between the policies from Setting A and Setting B.



(a) Setting A, Seed 2 (b) Setting A, Seed 4 (c) Setting B, Seed 42

Figure 4.3: One year OOS period (2021) Weight distribution comparison. The X-axis represents the 11 sectors, while the Y-axis shows the weight of these sectors in the portfolios over the evaluation period.

4.1.3. 1 year vs. 9 year Out-of-Sample Testing

The S&P 500 achieved a return of 28.70% and a Sharpe ratio of 1.99 in the out-of-sample test year 2021. Although all single-agent PPO policies outperformed the S&P 500 in terms of the Sharpe ratio, only a subset of these models demonstrated superior performance in terms of total return. The impact of different seeds on the average daily weight change is evident, with policies in Setting A showing higher average daily weight changes compared to those in Setting B, as illustrated in Figure 4.3. Additionally, the results from the train-test split model, trained on data from 2006 to 2019 without using a rolling window, revealed the best performance in both total return and Sharpe ratio. However, it is important to consider that this superior performance may be influenced by the random seed or potentially more favorable market conditions for certain policies.

Table 4.2: 1-year (2021) OOS Portfolio Results of Single Agent PPO agents

Model-Training Type (Seed)	Total Return	Sharpe	MDD	Avg. Daily Weight Change
PPO- Setting B (42)	0.3015	2.2786	-0.0453	0.4603
PPO- Setting A (0)	0.2882	2.0874	-0.0538	0.7263
PPO- Setting A (1)	0.2785	2.0373	-0.0529	0.7295
PPO- Setting A (2)	0.2798	2.0540	-0.0523	0.7325
PPO- Setting A (3)	0.2760	2.0376	-0.0507	0.7373
PPO- Setting A (4)	0.2727	2.0202	-0.0494	0.7414
Train-Test Split (42)	0.3583	2.4968	-0.0502	0.6058

To evaluate the policy over a longer out-of-sample period, the models were trained on data from 2006 to 2013 and tested on data from 2015 to 2023. During this period, the S&P 500 achieved a return of 131% with a Sharpe ratio of 0.70. No RL policy outperformed the S&P 500 in terms of Sharpe ratio. However, policies with certain seeds (0 and 1) were able to slightly surpass the market return.

Table 4.3: 9 year (2015-2023) OOS Portfolio Results of Single Agent PPO agents

Model-Training Type (Seed)	Total Return	Sharpe	MDD	Avg. Daily Weight Change
PPO- Setting A (0)	1.2960	0.6182	-0.3193	0.5502
PPO- Setting A (1)	1.2972	0.6204	-0.3202	0.5695
PPO- Setting A (2)	1.2254	0.5991	-0.3214	0.5699
PPO- Setting A (3)	1.2309	0.5968	-0.3306	0.5590
PPO- Setting A (4)	1.2044	0.5888	-0.3389	0.5841
Train-Test Split (42)	0.9295	0.4954	-0.3864	0.6240

Plotting the log portfolio returns of the policies over time reveals that the policies remain close to the market performance in the years following the end of the training period(2015-2016). However, over a longer horizon, the policies tend to diverge more significantly from the market. The train-test split model behaves similarly to an equal-weight strategy during the extended evaluation period.

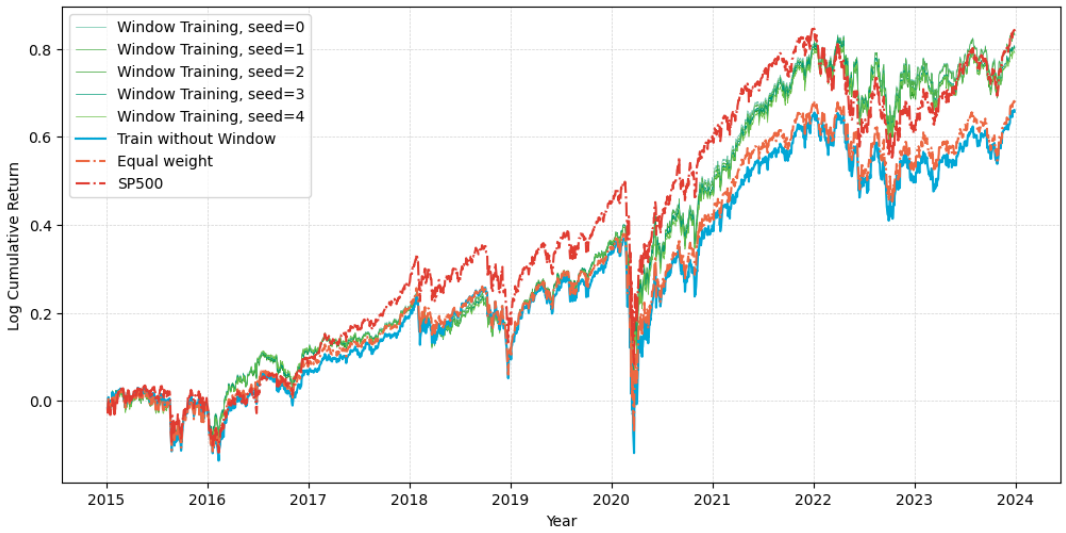


Figure 4.4: Single-agent baseline

Rolling window models may exhibit greater generalizability over the long term, adapting more flexibly to changing market conditions. In contrast, the train-test split model tends to perform better in capturing specific market conditions in the near future. This advantage may be attributed to its ability to leverage more recent data without the influence of older training periods. However, it is also important to consider that these results could be influenced by the choice of seed and prevailing market conditions during the testing period.

4.1.4. Limitation of the Single Agent Method

The single-agent method, while demonstrating the potential to outperform the benchmark, reveals significant limitations in robustness to hyperparameters, seed sensitivity, and long-term out-of-sample test performance. Analyzing the performance and behavior by decomposing the weights outputted by the policies of the single-agent model shows that, although it achieves results close to benchmark indices like the S&P 500 and equal-weight portfolios in short-term evaluation, it falls short in terms of long-term stability and generalizability.

Given these findings, it is imperative to explore multi-agent reinforcement learning models that can address these shortcomings, especially in longer-term out-of-sample risk-adjusted returns. Introducing a diversity term aims to develop a more robust policy. By employing multiple agents, each with potentially different strategies and perspectives, MARL is expected to enhance diversification and improve long-term out-of-sample performance.

4.2. Multi-Agent Models

4.2.1. Multi-Agent Results

The multi-agent reinforcement learning experiments were designed to evaluate the impact of introducing diversity measures on the agents' policies and their performance over an extended out-of-sample evaluation period, $[2015 - 2023]$. As specified in equations 3.9 and 3.10, the diversity measures were integrated into the reward function to encourage diverse actions among agents.

Correlation as Diversity Penalty

Without any correlation penalties, the agents' actions exhibited a high average absolute correlation of 81%. Figure 4.5 illustrates that as the weight of the diversity penalty λ increases, the absolute correlation of the actions between agents decreases, evident from the top to the bottom of the heatmap. This trend demonstrates the effectiveness of incorporating diversity measures, as higher λ values lead to more diverse actions among agents. The observed reduction in correlation as λ increases implies that encouraging diversity among agents leads to more distinct investment strategies.

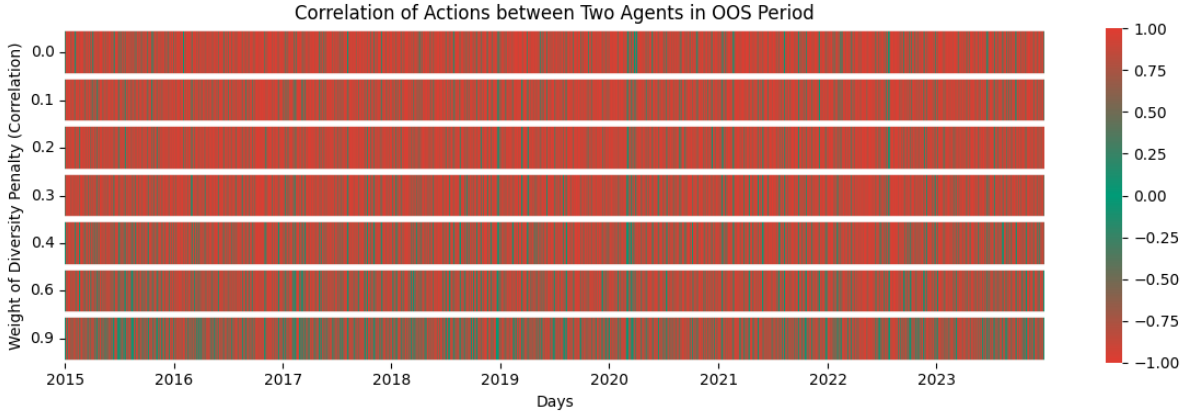


Figure 4.5: Correlation of weights between agents over time (x-axis) for different correlation penalty weights λ (y-axis), with a color scale where higher absolute correlation values are depicted in red and lower values in green.

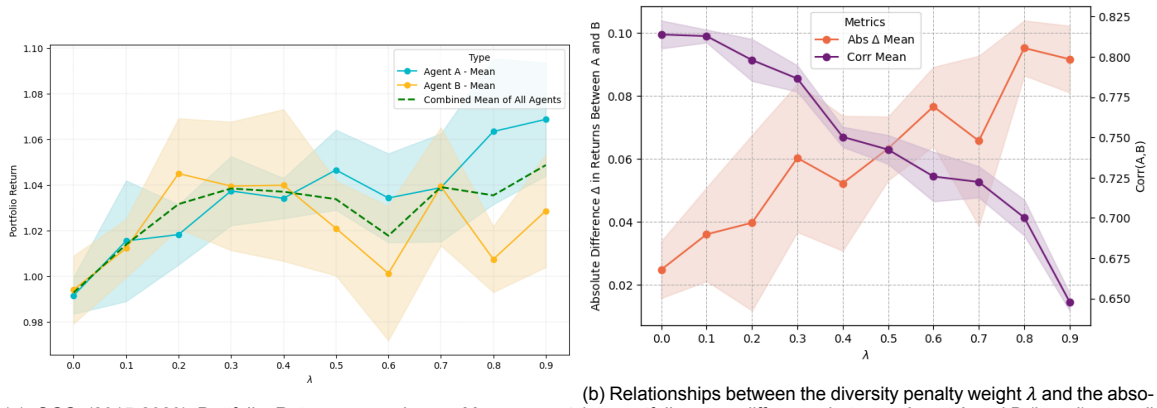
Figure 4.6a illustrates the variation in 9-year out-of-sample portfolio returns as a function of the diversity penalty, λ, for two concurrently trained agents, Agent A and Agent B. The plot aggregates results across four different seeds $[0, 1, 2, 3]$, with the mean return across all seeds represented by the solid line, and the shaded region indicates the standard error calculated from the four seeds. The left y-axis of Figure 4.6b represents the absolute difference in portfolio returns between Agent A and Agent B, Δ with 4 seeds. Policies with higher λ values tend to show greater differences between the two agents' returns. The y-axis on the right side represents the average correlation between two agents and demonstrates that as λ increases, the correlation between the agents decreases, while the absolute difference in returns, Δ, shows an upward trend.

Table 4.4: Portfolio statistics of the agents trained using correlation as the diversity penalty; each row is computed from 8 agents (2 agents per training run, averaged over 4 seeds).

DP=Corr	**Portfolio Return**		**Sharpe Ratio**		**Max Drawdown**		**Daily Turnover**	
λ	μ	σ	μ	σ	μ	σ	μ	σ
0	0.9928	0.0222	0.5208	0.0067	-0.1201	0.0185	0.3466	0.0098
0.1	1.0138	0.0385	0.5281	0.0124	-0.1226	0.0148	0.3432	0.0059
0.2	1.0316	0.0388	0.5336	0.0127	-0.1171	0.0139	0.3372	0.0062
0.3	1.0384	0.0418	0.5354	0.0149	-0.1187	0.0200	0.3328	0.0043
0.4	1.0370	0.0451	0.5358	0.0153	-0.1083	0.0207	0.3268	0.0070
0.5	1.0337	0.0381	0.5353	0.0140	-0.1109	0.0239	0.3190	0.0044
0.6	1.0177	0.0494	0.5302	0.0169	-0.1158	0.0251	0.3158	0.0045
0.7	1.0390	0.0459	0.5362	0.0166	-0.1162	0.0157	0.3071	0.0068
0.8	1.0354	0.0547	0.5361	0.0184	-0.1123	0.0254	0.2996	0.0091
0.9	1.0486	0.0505	0.5409	0.0170	-0.1048	0.0172	0.2927	0.0053

The portfolio statistics presented in Table 4.4 further indicate that as the diversity penalty, λ, increases, both portfolio return and Sharpe ratio improve. It is noteworthy that daily turnover and portfolio

return exhibit a strong negative correlation with λ, with a correlation coefficient of -0.99. The decline in daily turnover is particularly significant from a practical standpoint, indicating reduced transaction costs.



(a) OOS (2015-2023) Portfolio Return comparison. Mean over 4 seeds, DP = corr. Values in Table 4.5

(b) Relationships between the diversity penalty weight λ and the absolute portfolio return difference between Agent A and B (in red), as well as the correlation between the agents(in purple).

Figure 4.6: Results of Portfolios Derived from Correlation as Diversity Penalty Scenario. Lines are the mean of the metrics, and standard error is represented in a shaded area.

Table 4.5: λ and Correlation Between Agents

λ	Seed=0	Seed=1	Seed=2	Seed=3	μ	σ
0	0.8205	0.8298	0.8160	0.7894	0.8139	0.0150
0.1	0.8101	0.8244	0.8112	0.8058	0.8129	0.0069
0.2	0.7881	0.8197	0.8186	0.7658	0.7980	0.0225
0.3	0.7629	0.7916	0.8001	0.7916	0.7865	0.0141
0.4	0.7647	0.7366	0.7441	0.7559	0.7503	0.0108
0.5	0.7362	0.7264	0.7680	0.7393	0.7425	0.0155
0.6	0.7362	0.6825	0.7293	0.7549	0.7257	0.0267
0.7	0.6993	0.7298	0.7158	0.7445	0.7223	0.0167
0.8	0.7222	0.7086	0.7002	0.6700	0.7003	0.0192
0.9	0.6609	0.6391	0.6391	0.6526	0.6479	0.0093

Table 4.6: λ and Δ of portfolio return between agents

λ	Seed=0	Seed=1	Seed=2	Seed=3	μ	σ
0	0.0436	0.0095	0.0216	0.0465	0.0248	0.0180
0.1	0.0505	0.0395	0.0407	0.0476	0.0360	0.0297
0.2	0.0899	0.0982	0.0228	0.0576	0.0397	0.0558
0.3	0.0044	0.0573	0.0063	0.0542	0.0603	0.0471
0.4	0.0984	0.0787	0.0927	0.0602	0.0522	0.0428
0.5	0.0319	0.0656	0.1227	0.1249	0.0633	0.0203
0.6	0.1097	0.0755	0.0873	0.1325	0.0766	0.0251
0.7	0.0786	0.1033	0.0009	0.0114	0.0657	0.0540
0.8	0.0102	0.0125	0.0425	0.0840	0.0952	0.0174
0.9	0.0803	0.0041	0.1197	0.1051	0.0917	0.0212

Total Variation Distance as Diversity Penalty

The average Total Variation (TV) Distance between agents without any diversity penalties was 0.07. In Figure 4.7 and Table 4.8, it is possible to see that the TV distance between the agent's actions over time increases as λ increases.

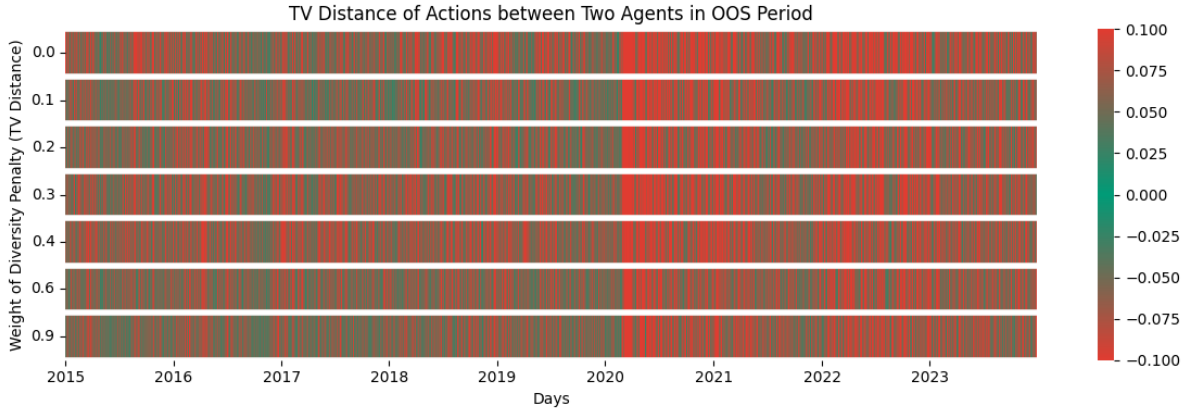
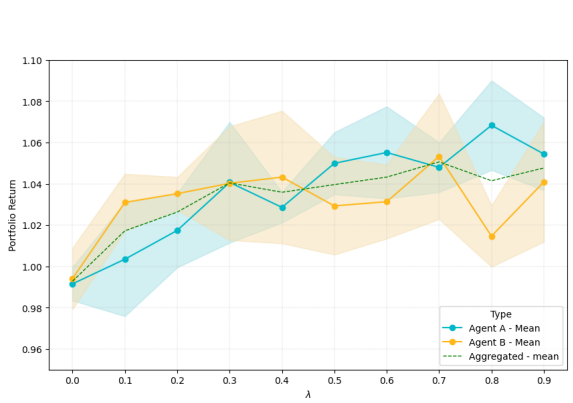


Figure 4.7: The heatmap of the total variation distance of weights between agents over time (x-axis) for different TV Distance penalty weights λ (y-axis), higher absolute TVD values are depicted in red and lower absolute values in green.

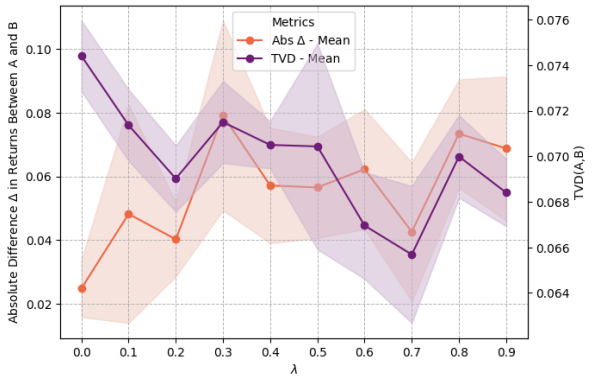
Table 4.7: Portfolio statistics using TV Distance as the diversity penalty; each row is computed from 8 agents (2 agents per training run, averaged over 4 seeds).

DP=TVD	**Portfolio Return**		**Sharpe Ratio**		**Max Drawdown**		**Daily Turnover**	
λ	σ	σ	μ	σ	μ	σ	μ	σ
0	0.9928	0.0222	0.5208	0.0067	-0.1201	0.0185	0.3466	0.0098
0.1	1.0172	0.0430	0.5288	0.0144	-0.1171	0.0085	0.3488	0.0056
0.2	1.0263	0.0274	0.5296	0.0105	-0.1191	0.0172	0.3534	0.0070
0.3	1.0404	0.0526	0.5375	0.0169	-0.1107	0.0195	0.3516	0.0054
0.4	1.0359	0.0438	0.5363	0.0132	-0.1122	0.0158	0.3480	0.0069
0.5	1.0396	0.0383	0.5365	0.0111	-0.1122	0.0189	0.3490	0.0085
0.6	1.0432	0.0395	0.5340	0.0135	-0.1218	0.0200	0.3474	0.0077
0.7	1.0506	0.0429	0.5382	0.0148	-0.1160	0.0160	0.3452	0.0064
0.8	1.0415	0.0448	0.5313	0.0154	-0.1179	0.0238	0.3424	0.0092
0.9	1.0477	0.0450	0.5345	0.0176	-0.1157	0.0248	0.3413	0.0071

The relationship between the difference in portfolio returns (Δ) and the diversity penalty illustrated in Figure 4.8b is less distinct compared to the case when the correlation is used as the diversity metric. Nevertheless, it remains observable, in both Figure 4.8a and Table 4.7, that increasing the weight of the diversity penalty in reward during training also increased the out-of-sample return and Sharpe ratio of the portfolios created.



(a) OOS (2015-2023) Portfolio Return comparison. Mean over 4 seeds, DP = TVD. Values in Table 4.8



(b) Relationships between the diversity penalty weight λ and the absolute portfolio return difference between Agent A and B (in red), as well as the correlation between the agents(in purple).

Figure 4.8: Impact of diversity penalty TV Distance to portfolio return. Lines are the mean of the metrics, and standard error is represented in a shaded area.

Table 4.8: λ and TVD Between Agents

λ	**Seed=0**	**Seed=1**	**Seed=2**	**Seed=3**	μ	σ
0	0.0743	0.0708	0.0740	0.0784	0.0744	0.0027
0.1	0.0722	0.0709	0.0674	0.0750	0.0714	0.0027
0.2	0.0671	0.0687	0.0670	0.0732	0.0690	0.0025
0.3	0.0676	0.0763	0.0708	0.0713	0.0715	0.0031
0.4	0.0677	0.0723	0.0717	0.0704	0.0705	0.0018
0.5	0.0640	0.0833	0.0702	0.0642	0.0704	0.0078
0.6	0.0676	0.0730	0.0656	0.0617	0.0670	0.0041
0.7	0.0625	0.0747	0.0633	0.0623	0.0657	0.0052
0.8	0.0659	0.0747	0.0692	0.0701	0.0700	0.0031
0.9	0.0704	0.0691	0.0641	0.0701	0.0684	0.0026

Table 4.9: λ and Δ of portfolio return between agents

λ	**Seed=0**	**Seed=1**	**Seed=2**	**Seed=3**	μ	σ
0	0.0436	0.0019	0.0390	0.0694	0.0248	0.0180
0.1	0.0425	0.0114	0.0075	0.0604	0.0482	0.0686
0.2	0.0593	0.0250	0.0228	0.0300	0.0402	0.0233
0.3	0.0449	0.0773	0.0325	0.0589	0.0792	0.0597
0.4	0.0920	0.0101	0.0751	0.0347	0.0571	0.0362
0.5	0.0319	0.1495	0.0669	0.1575	0.0565	0.0318
0.6	0.1111	0.0719	0.0694	0.0959	0.0623	0.0377
0.7	0.0396	0.1105	0.0009	0.0114	0.0426	0.0436
0.8	0.0102	0.0125	0.0425	0.0840	0.0734	0.0341
0.9	0.0803	0.0041	0.1197	0.1051	0.0688	0.0452

4.3. Hyperparameters

4.3.1. Hyperparameter Tuning

Figure 4.9 visualizes the hyperparameter sweep results across different combinations of key hyperparameters: seed, diversity type (TV Distance and Correlation), and diversity weight. The sweep aims to identify the best configuration for maximizing the one-year out-of-sample validation period episode reward mean (on the right). The plot highlights how variations in these parameters impact the performance objective. It is hard to pinpoint the optimal settings that yield the highest mean reward, although

they offer insights into the influence of each parameter on the overall model performance. The hyperparameters used in the model can be found in Appendix A

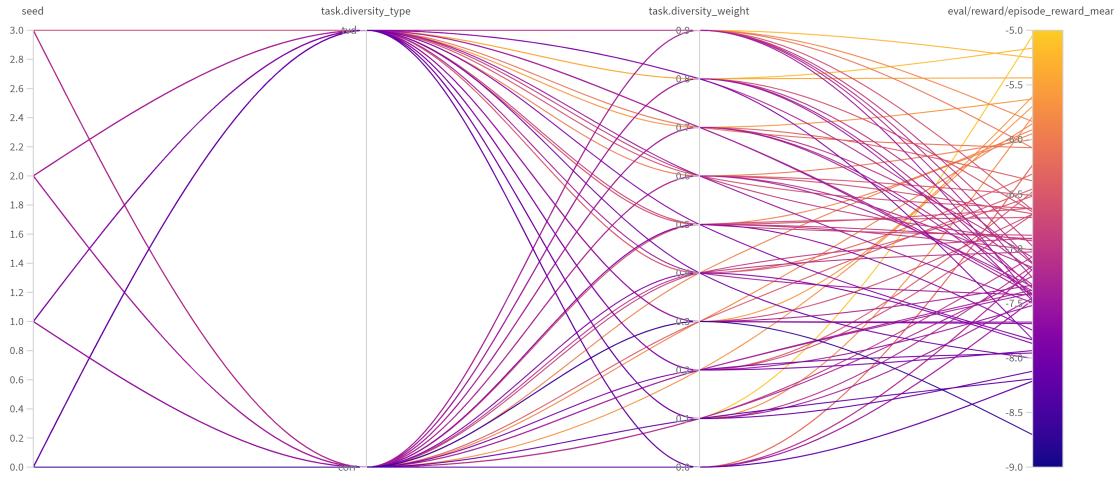


Figure 4.9: Hyperparameter Sweep Results via Weights and Biases

4.3.2. Importance of the Random Seed

The hyperparameter sweep conducted in this study underscored the importance of the random seed in DRL experiments. The hyperparameter with the highest impact on the validation reward was the seed, with a correlation of 0.75. The correlation between the diversity weight and validation reward was 0.20, followed by the type of diversity penalty, 0.14. Colas et al. [15] emphasized how RL algorithms are particularly susceptible to variations of random seeds, given the inherently stochastic nature of both the environment and the learning process. Policy exploration is driven by the seed. Slight changes in the random seed can lead an agent to explore vastly different parts of the state-action space, ultimately altering the optimization path and resulting in different learned policies [15, 29]. The random initialization of the neural network itself can lead to diverging results across different random seeds [51, 61]

5 Discussion

Reinforcement Learning has demonstrated remarkable success in various simulated environments [53, 72], where its capabilities have been extensively validated through vast experiments. However, these idealized environments for real-world applications present additional challenges. Dulac-Arnold et al. [19] outlined challenges associated with applying RL in practical scenarios. In this chapter, I address the challenges and limitations outlined by [19], as well as additional challenges encountered in applying multi-agent RL and other relevant issues observed during the research.

Limited State and Action Spaces
In finance, cross-sectional analysis, which involves examining stocks relative to one another, is essential for making informed investment decisions [6, 25]. However, implementing this type of analysis in RL is challenging due to the exponentially growing state and action spaces as the number of assets increases. This complexity can significantly impede the convergence of RL algorithms, making it difficult to effectively model and solve the problem [64]. I worked with 11 assets, similar to the number of assets used in other RL applications within portfolio management [33, 37, 43, 84]. Consequently, the feasibility of RL models for broad cross-sectional analysis is limited, impacting their ability to construct portfolios or strategies across a wide array of assets.

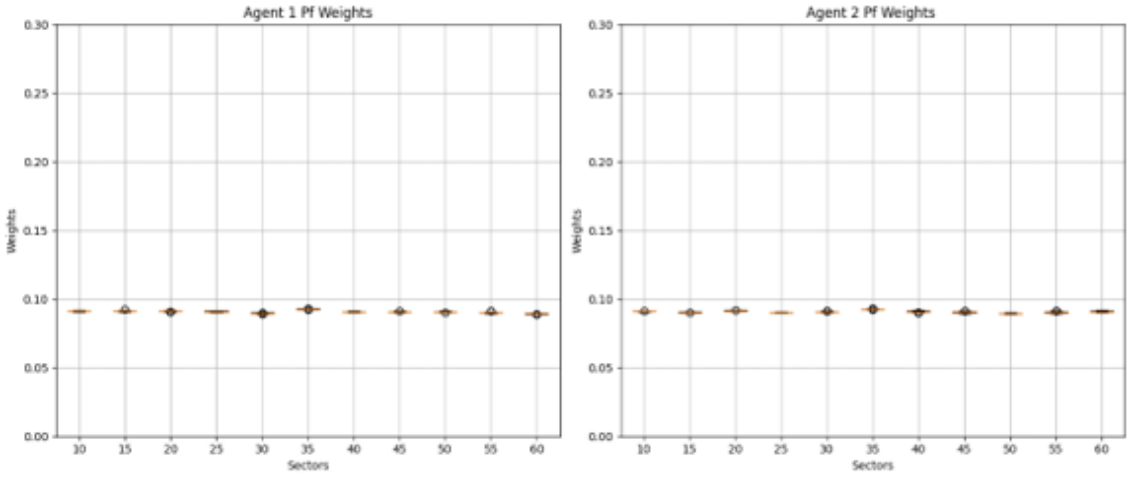
From my experience, practical applications of RL in portfolio management might be more feasible when focused on specific tasks such as sector timing. Utilizing the RL-generated policy as sector timing signals could serve as a valuable tool in portfolio management by indicating which sectors to overweight or underweight at each time step. Given the current limitations and complexity of modeling large-scale cross-sectional analysis in financial markets, this targeted application appears to be a more manageable and practical use of RL's capabilities.

Partial Observability and Non-Stationarity
Financial markets are inherently complex and often partially observable, meaning that not all relevant information about the market or the assets is available to the decision-making algorithm. In this work, I focused solely on returns, which simplified the problem by reducing the dimensionality of the data. However, real-world scenarios involve a broader range of factors influencing asset prices. To address this, expanding the dataset to include additional factors, as suggested by [22, 77], could provide more information about the returns.

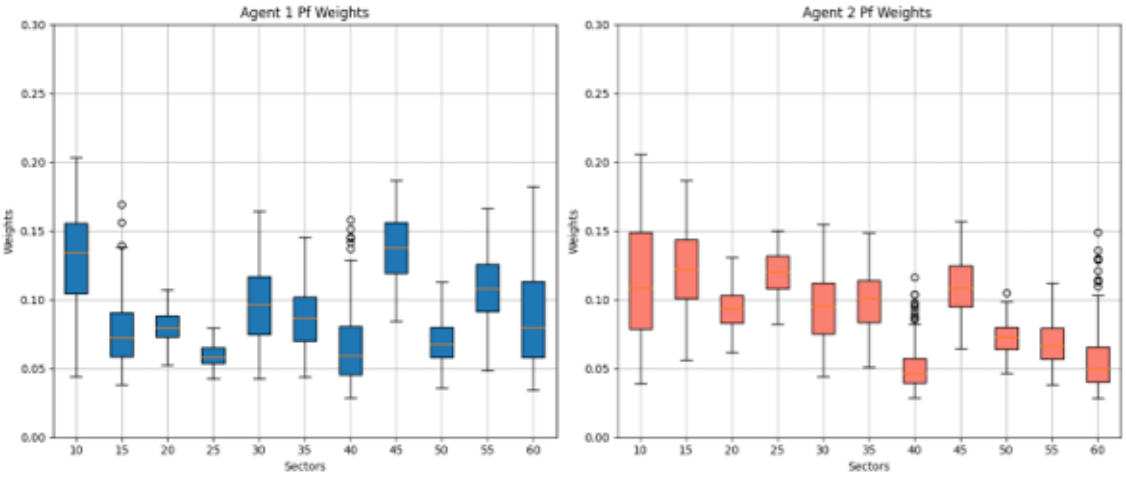
Financial markets are also characterized by non-stationarity, meaning their statistical properties change over time. This dynamic nature makes it challenging for RL models, which assume stationarity; however, the optimal action for a given observed state can change over time as the environment evolves, making it difficult to maintain consistent decision-making strategies.

Reward Function Design
While the goal in RL is to optimize a global reward, determining the appropriate reward structure can be challenging. Simulated environments can have a clear relationship between actions and rewards, whereas real-world applications often lack such straightforward signals. It is crucial to design the reward function to align with the intended objectives in the environment [19].

Since RL algorithms such as DQN and PPO aim to maximize the sum of rewards (arithmetic rewards), using logarithmic portfolio returns, proposed by Jiang et al. [33] as the reward is often adapted in extended studies [43, 84, 87]. Summing logarithmic returns directly corresponds to cumulative returns in the non-logarithmic scale, avoiding the multiplicative operations required for geometric returns. This method aligns with one of the optimization objectives in portfolio management: maximizing cumulative returns.



(a) Reward: Log return

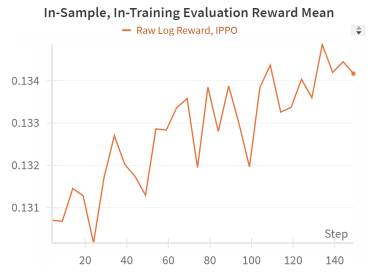


(b) Reward: Differential Sharpe ratio

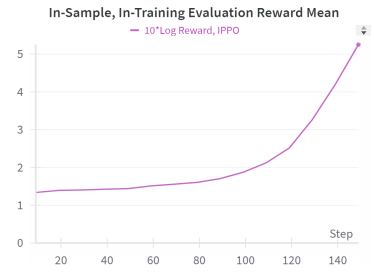
Figure 5.1: Action (PF weight) distributions of Agents trained with different types of rewards as environment reward.

One possible explanation for the large policy difference between the log return reward and DSR reward scenarios is that the hyperparameters of PPO used in this study, adapted from [73], were specifically optimized for environments with DSR as a reward. As hyperparameters in RL are problem-specific [20, 32, 34], this mismatch could result in suboptimal action distribution. Another possible explanation is that the small scale of logarithmic returns might not provide sufficiently strong gradients for stochastic gradient descent (SGD) and neural networks to learn effectively, leading the model to converge to the simplest solution—an equal-weight portfolio. For example, a daily portfolio return of 1% translates to approximately $\log(1 + 0.01) = 0.004$. Considering that reward values are typically recommended [32, 34] to be within the ranges of $[-1, 1]$ or $[-10, 10]$, such small magnitudes may hinder effective learning. To further investigate this potential issue, I trained two additional models using scaled rewards: (b) $10\times$ (log return) and (c) $100\times$ (log return). All other hyperparameters, including the number of timesteps, batch size, and learning rate, were held constant across the three runs. Figures in 5.2 present the evaluation reward curves (upper row) and PPO loss metrics (lower row). The

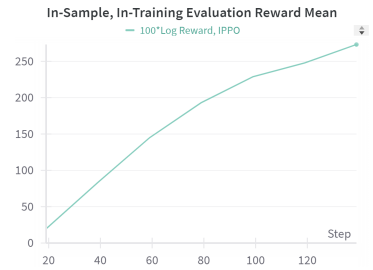
models exhibited noticeably different behaviors solely due to differences in reward scaling, highlighting the importance of careful reward design and problem-specific hyperparameter tuning. The observation aligns with the finding of Engstrom et al. [21], which emphasized the significant effect of reward scaling on RL algorithm performance.



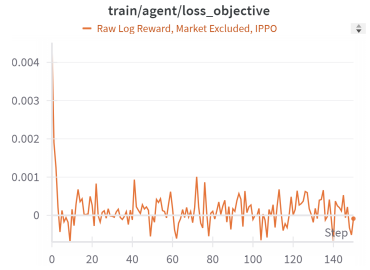
(a) Log return as RL reward



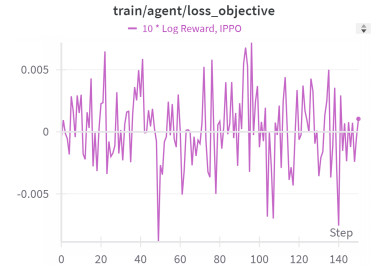
(b) $10 *$ Log return as RL reward



(c) $100 *$ Log return as RL reward



(d) PPO Loss Curve, Log return as RL reward



(e) Loss curve, RL reward = $10 *$ Log return

(f) Loss curve, RL reward = $100 *$ Log return

Figure 5.2: Upper Row: Evaluation Curve, Lower Row: Loss Curve

5.0.1. Importance of Model Hyperparameters

Many hyperparameters in RL algorithms are fine-tuned for environments with well-scaled reward structures[20, 44]. However, real-world environments often do not naturally provide rewards that fall within this ideal range, requiring additional scaling techniques to ensure stable learning dynamics[19].

Furthermore, in standard reinforcement learning environments where the stationarity assumption holds, and there is a clear relationship between actions and environmental changes, hyperparameter tuning, including different seeds, tends to be less problematic. In such settings, a comprehensive hyperparameter sweep can be performed to identify the optimal parameters, and these parameters will generally remain effective as the task environment is consistent [20]. However, in portfolio management applications, the market dynamics can shift dramatically over time. Consequently, hyperparameters that performed well in 2010 may not be optimal in 2024. This variability underscores the necessity for routine hyperparameter sweeps in RL applications within finance. Although this process is computationally intensive, it is crucial for maintaining the effectiveness of the models as market conditions evolve.

Importance of Random Seeds

The results are presented in the Section 4.2.1 as aggregated means and standard deviations across 8 agents for each λ value. Neglecting random seed variability and relying solely on results from a single seed could have led to different conclusions compared to those drawn from the aggregated averages across multiple seeds. For example, Figure 5.3 shows the portfolio returns from agents trained with seed 2. If conclusions were drawn solely from this plot, one might have inferred that $\lambda = 0.3$ and $\lambda = 0.4$ yield the best performance, as these settings show the highest portfolio returns. Analyzing the results across all four seeds revealed a positive correlation between λ and portfolio returns.

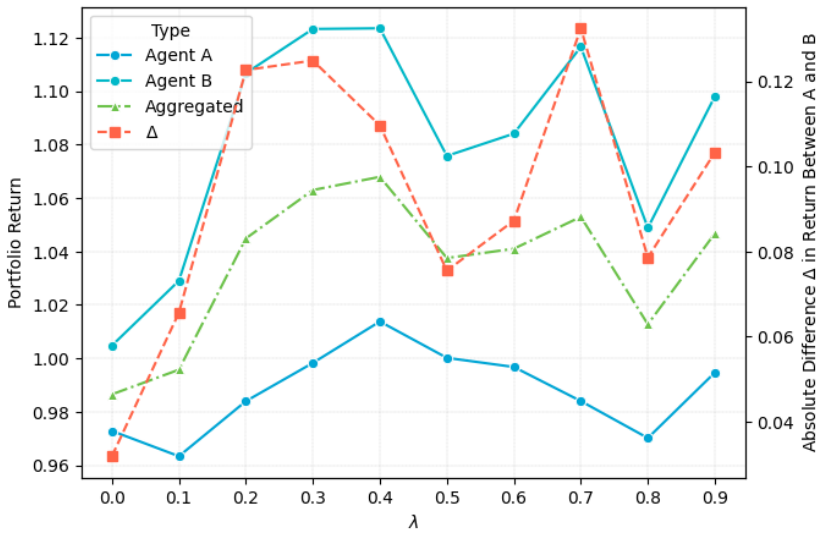
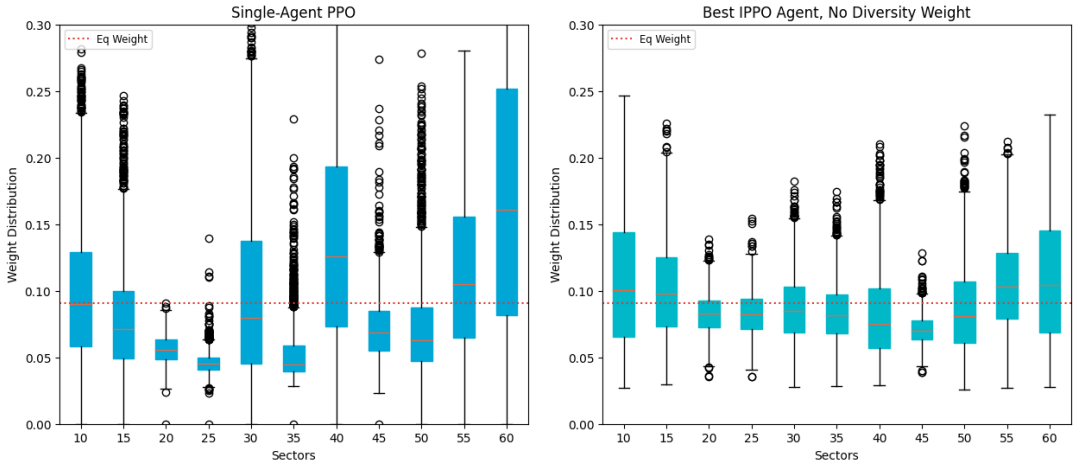


Figure 5.3: OOS (2015-2023) Portfolio Return vs. λ by Agents, Aggregated and Δ between agents. Seed = 2, DP = corr

Limitations on 1:1 comparison with single-agent PPO due to adaptation difference
In attempting a 1:1 comparison between the multi-agent and single-agent PPO frameworks, divergence in the resulting policies, expressed as weight distributions per sector in a portfolio, was observed, as shown in Figure 5.4. For agents that were trained under the same number of timesteps, learning rate, entropy coefficient, and other coefficients, The single-agent model utilized Stable Baselines 3 [62], while the multi-agent RL agents used IPPO implementation of BenchMARL, which again is based on a PPO implementation based on TorchRL. A key issue was the inability to directly translate hyperparameters from Stable Baselines 3 to BenchMARL, contributing to policy discrepancies. Recognizing the critical role of hyperparameters, I contributed to the project by providing a hyperparameter tuning framework to address these adaptation differences and reduce the observed policy gaps.



(a) Single PPO agent in Stable Baselines3 with fine-tuned hyperparameters

(b) IPPO agent in BenchMARL, with roughly tuned hyperparameters

Figure 5.4: Policies (Weight Distributions) of two agents trained with different implementations of PPO

6

Conclusion

The primary goal of this project was to explore the application of multi-agent reinforcement learning in portfolio management, with a particular focus on encouraging diversity among agents. Encouraging portfolio diversity is effective in reducing risk by lowering the correlation between assets, as demonstrated in academic research Markowitz [49] (1952) and supported by real-world practices used by human portfolio managers [74].

Motivated by these considerations, the study aimed to address two main research objectives: **(1) investigating a robust single-agent reinforcement learning portfolio management strategy**, and **(2) strengthening this strategy using multi-agent reinforcement learning**. The overall conclusion is that robust portfolio management relies on careful hyperparameter tuning and the use of rolling window training to mitigate overfitting. In the context of MARL, introducing both correlation and total variation distance as diversity measures resulted in improved portfolio performance metrics.

The first question, *What is the proper training framework to develop a single-agent portfolio management strategy that the model outperforms the market during the out-of-sample period, specifically in terms of cumulative return and Sharpe ratio?* is answered in Section 4.1 of the report. By replicating the results of Sood et al. [73], it was observed that hyperparameters and random seed significantly influence model performance. The result showed that the rolling window training approach with [73] hyperparameters showed the highest Sharpe ratio and portfolio return during 9-year out-of-sample periods. For long-term evaluation, models trained using a rolling window approach demonstrated superior performance to those without windowed training.

The first sub-question regarding MARL, *Can MARL models deliver higher risk-adjusted returns than a single-agent model?*, was addressed in Section 4.2.1, which indicates that the MARL approach did not achieve higher risk-adjusted returns. For the second sub-question, *Will encouraging diversity between the actions deliver higher risk-adjusted returns?*, Section 4.2.1 showed that promoting diversity during training, specifically correlation and total variation distance, resulted in more diverse actions during the testing phase. Introducing these diversity measures improved both portfolio returns and the Sharpe ratio compared to multi-agent models without a diversity penalty. Additionally, the Sharpe ratios of the MARL models were competitive with those of single-agent models that had been carefully fine-tuned.

The key takeaways from this work emphasize several critical aspects necessary in using (multi-agent) reinforcement learning in portfolio management. First, the results underscored the importance of thorough robustness testing, particularly regarding the hyperparameters, random seeds, and using a testing period longer than the typical 80:20 data split. I analyzed the policies trained on 8-year information on a 9-year out-of-sample period and showed that rolling window-trained policies are better at generalization over the longer time horizon. The performance difference between PPO agents and IPPO agents that were based on two different implementations of the original algorithm, discussed in Section 5.0.1, highlights the importance of problem and algorithm-specific hyperparameters and multiple seeds testing. The results underscored the importance of employing robust training methodologies and problem-specific hyperparameters to achieve consistent, long-term performance. Without such fine-tuning, models are prone to overfitting or suboptimal behavior, particularly in complex financial environments.

This work represents a novel contribution to the field by being the first to explore the application of IPPO with homogeneous portfolios and continuous action spaces. To my knowledge, it also marks the first investigation of different diversity measures—such as correlation and total variation distance—in portfolio management. These explorations provide valuable insights into how diversity among agents can be encouraged to enhance portfolio performance. The discussion in Chapter 5 highlighted and emphasized the significance of hyperparameter fine-tuning and several other limitations and challenges in applying (MA)RL to real-world scenarios and portfolio management. This research underscores the complexities of applying reinforcement learning to portfolio management and highlights the potential for strengthening the strategy via encouraging diversity among the agents. These insights also pave the way for future work, which will be discussed in the following chapter.

7

Future Works

While this study has shown the potential of (MA)RL models in portfolio management applications, particularly through encouraging agent diversity, it has also exposed several areas where enhancements can be made. Building on these findings, this chapter outlines key directions for future research that can address these limitations and expand the applicability of RL in finance.

Improving Data Representation to the Model
I used flattened past 60-day returns to represent historical information. However, financial markets often exhibit complex temporal dependencies that simple return windows cannot fully capture. Future research could explore advanced sequential models such as Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks, which are better suited to capture long-term dependencies in time series data. Additionally, off-policy algorithms like the soft actor-critic (SAC) algorithm could be introduced to improve the learning process by leveraging experience replay and learning from past experiences more effectively.

Expanding State Space
While this study focused on price returns as the primary input, incorporating additional factors could improve decision-making. Factors that capture specific asset characteristics, such as value, momentum, and size, as identified in Fama and French [22] and [77], can improve the RL model's understanding of asset behavior and contribute to a more robust portfolio construction. Additionally, future work could explore different rebalancing frequencies, such as taking a new action every week or month instead of every day, to provide more realistic portfolio management strategies in terms of trading cost and market impact, which are assumed to be ignored in the course of this study.

Hyperparameter Optimization Strategies
The study highlighted the critical role of hyperparameters, yet only a limited grid search was conducted due to computational constraints. Future research should explore advanced hyperparameter optimization techniques with higher computational power, such as random search and Bayesian optimization, to find the best combination of problem-specific hyperparameters. These methods can better navigate the hyperparameter space and identify settings that generalize well across market conditions.

Deeper Exploration on Diversity in Multi-Agent Models
The current MARL setup relied primarily on initialization differences and diversity penalties to encourage diverse agent behavior. However, there are several avenues for enhancing diversity and specialization among agents:

- **Structural Diversity:** Introducing agents with different network architectures as in [40] or different hyperparameters can promote diversified strategies driven by structural differences, reflecting the heterogeneous nature of real-world portfolio managers.

- **Task-Specialized Agents:** Instead of agents controlling the homogeneous set of investment instruments, future work could explore task-specialized agents, such as sector-specialized agents that are trained on certain industry sectors that control the portfolio consisting of individual stocks. These specialized agents could then further collaborate using shared state spaces or rewards, which is expected to allow them to benefit from non-independent MARL algorithms.
- **Advanced Diversity Techniques:** Incorporating recent advances in diversity-driven learning, such as network diversity methods [8] or diversity-promoting regularization techniques [59], could further improve risk-adjusted returns by ensuring a more diversified set of policies.
- **Number of Agents:** The current experiment consisted of only two agents. Expanding to more agents could benefit from further diversification similar to population-based training, leading to richer strategic interactions and improved model performance.

These suggestions are expected to address the current limitations, focusing on the robustness of the RL model and the way to introduce diversity in the system. RL's applicability to real-world finance will continue to grow as these challenges are tackled, paving the way for more reliable and adaptive portfolio strategies.

These proposals aim to enhance the robustness, diversity, and ultimately the performance of portfolios, addressing key limitations identified in the current single- and multi-agent reinforcement learning models in this research, covered in the discussions in Chapter 5. Through explorations into these areas, the applicability of reinforcement learning in portfolio management contexts could be incrementally strengthened, potentially leading to more innovative and stable portfolio strategies.

Bibliography

AbdelKawy, R., Abdelmoez, W. M., & Shoukry, A. (2021). A synchronous deep reinforcement learning model for automated multi-stock trading. *Progress in Artificial Intelligence*, *10*(1), 83–97. https://doi.org/10.1007/s13748-020-00225-z

Aboussalah, A. M., & Lee, C.-G. (2020). Continuous control with Stacked Deep Dynamic Recurrent Reinforcement Learning for portfolio optimization. *Expert Systems with Applications*, *140*, 112891. https://doi.org/10.1016/j.eswa.2019.112891

Almahdi, S., & Yang, S. Y. (2017). An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, *87*, 267–279. https://doi.org/10.1016/j.eswa.2017.06.023

Andrychowicz, M., Raichuk, A., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., & Bachem, O. (2021). WHAT MATTERS FOR ON-POLICY DEEP ACTOR-CRITIC METHODS? A LARGE-SCALE STUDY.

Ashwin Rao & Tikhon Jelvis. (2022, October). *Foundations of Reinforcement Learning with Applications in Finance* [MAG ID: 4306822038 S2ID: 5f21db42c3cc43fd2e735e265452b365d52d8145]. https://doi.org/10.1201/9781003229193

Bali, T. G. (n.d.). Empirical Asset Pricing.

Bettini, M., Prorok, A., & Moens, V. (n.d.). BenchMARL: Benchmarking Multi-Agent Reinforcement Learning.

Bettini, M., Shankar, A., & Prorok, A. (2023, May). System Neural Diversity: Measuring Behavioral Heterogeneity in Multi-Agent Learning [arXiv:2305.02128 [cs]]. Retrieved May 23, 2024, from http://arxiv.org/abs/2305.02128

Bou, A., Bettini, M., Dittert, S., Kumar, V., Sodhani, S., Yang, X., Fabritiis, G. D., & Moens, V. (2023). TorchRL: A data-driven decision-making library for PyTorch [_eprint: 2306.00577].

Busoniu, L., Babuska, R., & De Schutter, B. (2008). A Comprehensive Survey of Multiagent Reinforcement Learning [Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)]. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(2), 156–172. https://doi.org/10.1109/TSMCC.2007.913919

Canese, L., Cardarilli, G. C., Di Nunzio, L., Fazzolari, R., Giardino, D., Re, M., & Spanò, S. (2021). Multi-Agent Reinforcement Learning: A Review of Challenges and Applications [Number: 11 Publisher: Multidisciplinary Digital Publishing Institute]. *Applied Sciences*, *11*(11), 4948. https://doi.org/10.3390/app11114948

Chen, L., Pelger, M., & Zhu, J. (2023). Deep Learning in Asset Pricing [Publisher: INFORMS]. *Management Science*. https://doi.org/10.1287/mnsc.2023.4695

Chen, M.-Y., Chen, C.-T., & Huang, S.-H. (2023). Knowledge distillation for portfolio management using multi-agent reinforcement learning. *Advanced Engineering Informatics*, *57*, 102096. https://doi.org/10.1016/j.aei.2023.102096

Chen, Y.-Y., Chen, C.-T., Sang, C.-Y., Yang, Y.-C., & Huang, S.-H. (2021). Adversarial Attacks Against Reinforcement Learning-Based Portfolio Management Strategy. *IEEE Access*, *PP*, 1–1. https://doi.org/10.1109/ACCESS.2021.3068768

Colas, C., Sigaud, O., & Oudeyer, P.-Y. (2018, July). How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments [arXiv:1806.08295 [cs, stat]]. https://doi.org/10.48550/arXiv.1806.08295

DeMiguel, V., Garlappi, L., Nogales, F. J., & Uppal, R. (2009). A Generalized Approach to Portfolio Optimization: Improving Performance by Constraining Portfolio Norms [Publisher: INFORMS]. *Management Science*, *55*(5), 798–812. https://doi.org/10.1287/mnsc.1080.0986

de Witt, C. S., Gupta, T., Makoviichuk, D., Makoviychuk, V., Torr, P. H. S., Sun, M., & Whiteson, S. (2020, November). Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? [arXiv:2011.09533 [cs]]. https://doi.org/10.48550/arXiv.2011.09533

Dittrich, M.-A., & Fohlmeister, S. (2020). Cooperative multi-agent system for production control using reinforcement learning. *CIRP Annals*, *69*(1), 389–392. https://doi.org/10.1016/j.cirp.2020.04.005

Dulac-Arnold, G., Mankowitz, D., & Hester, T. (2019, April). Challenges of Real-World Reinforcement Learning [arXiv:1904.12901 [cs, stat]]. https://doi.org/10.48550/arXiv.1904.12901

Eimer, T., Lindauer, M., & Raileanu, R. (2023, June). Hyperparameters in Reinforcement Learning and How To Tune Them [arXiv:2306.01324 [cs]]. Retrieved July 4, 2024, from http://arxiv.org/abs/2306.01324

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., & Madry, A. (2020, May). Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO [arXiv:2005.12729 [cs, stat]]. Retrieved January 26, 2024, from http://arxiv.org/abs/2005.12729

Fama, E. F., & French, K. R. (1992). The Cross-Section of Expected Stock Returns [Publisher: [American Finance Association, Wiley]]. *The Journal of Finance*, *47*(2), 427–465. https://doi.org/10.2307/2329112

Gao, Z., Gao, Y., Hu, Y., Jiang, Z., & Su, J. (2020). Application of Deep Q-Network in Portfolio Management [arXiv:2003.06365 [cs, q-fin, stat]]. https://doi.org/10.48550/arXiv.2003.06365

Goodfellow, I., Benigo, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, *33*(5), 2223–2273. https://doi.org/10.1093/rfs/hhaa009

Gu, S., Kelly, B. T., & Xiu, D. (2019, September). Autoencoder Asset Pricing Models. https://doi.org/10.2139/ssrn.3335536

Gupta, J. K., Egorov, M., & Kochenderfer, M. (2017). Cooperative Multi-agent Control Using Deep Reinforcement Learning [Series Title: Lecture Notes in Computer Science]. *Autonomous Agents and Multiagent Systems*, *10642*, 66–83. https://doi.org/10.1007/978-3-319-71682-4_5

Heaton, J. B., Polson, N. G., & Witte, J. H. (2018, January). Deep Learning in Finance [arXiv:1602.06561 [cs]]. Retrieved January 30, 2024, from http://arxiv.org/abs/1602.06561

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning That Matters [Number: 1]. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1). https://doi.org/10.1609/aaai.v32i1.11694

Hu, S., Zhong, Y., Gao, M., Wang, W., Dong, H., Liang, X., Li, Z., Chang, X., & Yang, Y. (2023). MARLlib: A Scalable and Efficient Multi-agent Reinforcement Learning Library. *Journal of Machine Learning Research*, *24*(315), 1–23. Retrieved January 8, 2024, from http://jmlr.org/papers/v24/23-0378.html

Huang, N. E., Wu, M.-L., Qu, W., Long, S. R., & Shen, S. S. P. (2003). Applications of Hilbert–Huang transform to non□stationary financial time series analysis. *Applied Stochastic Models in Business and Industry*, *19*(3), 245–268. https://doi.org/10.1002/asmb.501

Huang, S., Dossa, R. F. J., Raffin, A., Kanervisto, A., & Wang, W. (2022). The 37 Implementation Details of Proximal Policy Optimization. *ICLR Blog Track*. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/

Jiang, Z., Xu, D., & Liang, J. (2017, July). A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem [arXiv:1706.10059 [cs, q-fin]]. https://doi.org/10.48550/arXiv.1706.10059

Jones, A. (n.d.). Debugging Reinforcement Learning Systems. Retrieved August 18, 2024, from https://andyljones.com/posts/rl-debugging.html

Jost, L. (2006). Entropy and Diversity [Publisher: [Nordic Society Oikos, Wiley]]. *Oikos*, *113*(2), 363–375. Retrieved February 14, 2024, from https://www.jstor.org/stable/40234813

Kirchner, U., & Zunckel, C. (2011, February). Measuring Portfolio Diversification [arXiv:1102.4722 [q-fin]]. https://doi.org/10.48550/arXiv.1102.4722

Koratamaddi, P., Wadhwani, K., Gupta, M., & Sanjeevi, S. G. (2021). Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation. *Engineering Science and Technology, an International Journal*, *24*(4), 848–859. https://doi.org/10.1016/j.jestch.2021.01.007

Kumar, N. (2022). An Army of Faceless Suits Is Taking Over the $4 Trillion Hedge Fund World. *Bloomberg.com*. Retrieved January 12, 2024, from https://www.bloomberg.com/news/features/2022-01-30/top-hedge-funds-citadel-millennium-shift-4-trillion-sector-from-rock-stars

Lee, J. W., Park, J., O, J., Lee, J., & Hong, E. (2007). A Multiagent Approach to Q-Learning for Daily Stock Trading. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *37*(6), 864–877. https://doi.org/10.1109/TSMCA.2007.904825

Lee, J., Raehyun Kim, Kim, R., Yi, S. W., & Kang, J. (2020). MAPS: Multi-Agent reinforcement learning-based Portfolio management System. [ARXIV_ID: 2007.05402 MAG ID: 3104176526 S2ID: 07ee4b23dce68111bf8c538d2bbb1f8f1feac639]. *International Joint Conference on Artificial Intelligence*, *5*, 4520–4526. https://doi.org/10.24963/ijcai.2020/623

Li, C., Wang, T., Wu, C., Zhao, Q., Yang, J., & Zhang, C. (2021, November). Celebrating Diversity in Shared Multi-Agent Reinforcement Learning [arXiv:2106.02195 [cs]]. https://doi.org/10.48550/arXiv.2106.02195

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Gonzalez, J., Goldberg, K., & Stoica, I. (n.d.). Ray RLlib: A Composable and Scalable Reinforcement Learning Library.

Liang, Z., Chen, H., Zhu, J., Jiang, K., & Li, Y. (2018, November). Adversarial Deep Reinforcement Learning in Portfolio Management [arXiv:1808.09940 [cs, q-fin, stat]]. https://doi.org/10.48550/arXiv.1808.09940

Liessner, R., Schmitt, J., Dietermann, A., & Bäker, B. (2019). Hyperparameter Optimization for Deep Reinforcement Learning in Vehicle Energy Management: *Proceedings of the 11th International Conference on Agents and Artificial Intelligence*, 134–144. https://doi.org/10.5220/0007364701340144

Lintner, J. (1965). Security Prices, Risk, and Maximal Gains from Diversification [_eprint: https://onlinelibrary.wiley.com/doi/pd 6261.1965.tb02930.x]. *The Journal of Finance*, *20*(4), 587–615. https://doi.org/10.1111/j.1540-6261.1965.tb02930.x

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning, 157–163. https://doi.org/10.1016/B978-1-55860-335-6.50027-1

Liu, X.-Y., Yang, H., Gao, J., & Wang, C. D. (2021). FinRL: Deep Reinforcement Learning Framework to Automate Trading in Quantitative Finance [arXiv:2111.09395 [cs, q-fin]]. *Proceedings of the Second ACM International Conference on AI in Finance*, 1–9. https://doi.org/10.1145/3490354.3494366

Liu, Y., Liu, Q., Zhao, H., Pan, Z., & Liu, C. (2020). Adaptive Quantitative Trading: An Imitative Deep Reinforcement Learning Approach [Number: 02]. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(02), 2128–2135. https://doi.org/10.1609/aaai.v34i02.5587

Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance*, *7*(1), 77. https://doi.org/10.2307/2975974

McKee, K. R., Leibo, J. Z., Beattie, C., & Everett, R. (2022). Quantifying the effects of environment and population diversity in multi-agent reinforcement learning [arXiv:2102.08370 [cs]]. *Autonomous Agents and Multi-Agent Systems*, *36*(1), 21. https://doi.org/10.1007/s10458-022-09548-8

Mehrer, J., Spoerer, C. J., Kriegeskorte, N., & Kietzmann, T. C. (2020). Individual differences among deep neural network models [Publisher: Nature Publishing Group]. *Nature Communications*, *11*(1), 5725. https://doi.org/10.1038/s41467-020-19632-w

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013, December). Playing Atari with Deep Reinforcement Learning [arXiv:1312.5602 [cs]]. https://doi.org/10.48550/arXiv.1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning [MAG ID: 2145339207]. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, *17*(5-6), 441–470. https://doi.org/10.1002/(SICI)1099-131X(1998090)17:5/6<441::AID-FOR707>3.0.CO;2-#

Moos, J., Hansel, K., Abdulsamad, H., Stark, S., Clever, D., & Peters, J. (2022). Robust Reinforcement Learning: A Review of Foundations and Recent Advances [Number: 1 Publisher: Multidisciplinary Digital Publishing Institute]. *Machine Learning and Knowledge Extraction*, *4*(1), 276–315. https://doi.org/10.3390/make4010013

Mossin, J. (1966). Equilibrium in a Capital Asset Market [Publisher: [Wiley, Econometric Society]]. *Econometrica*, *34*(4), 768–783. https://doi.org/10.2307/1910098

Neuneier, R. (1997). Enhancing Q-Learning for Optimal Asset Allocation. *Advances in Neural Information Processing Systems*, *10*. Retrieved January 30, 2024, from https://proceedings.neurips.cc/paper_files/paper/1997/hash/970af30e481057c48f87e101b61e6994-Abstract.html

OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H. P. d. O., Raiman, J., Salimans, T., … Zhang, S. (2019, December). Dota 2 with Large Scale Deep Reinforcement Learning [arXiv:1912.06680 [cs, stat]]. Retrieved February 4, 2024, from http://arxiv.org/abs/1912.06680

Parker-Holder, J., Pacchiano, A., Choromanski, K., & Roberts, S. (2020, October). Effective Diversity in Population Based Reinforcement Learning [arXiv:2002.00632 [cs, stat]]. https://doi.org/10.48550/arXiv.2002.00632

Pendharkar, P. C., & Cusatis, P. (2018). Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, *103*, 1–13. https://doi.org/10.1016/j.eswa.2018.02.032

Picard, D. (2023, May). Torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision [arXiv:2109.08203 [cs]]. https://doi.org/10.48550/arXiv.2109.08203

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, *22*(1), 268:12348–268:12355.

Rapach, D., Strauss, J., & Zhou, G. (2012, May). International Stock Return Predictability: What is the Role of the United States? https://doi.org/10.2139/ssrn.1508484

Richard S. Sutton & Andrew G. Barto. (2018). *Reinforcement Learning, Second Edition : An Introduction* (Vol. Second edition). Bradford Books.

Sato, Y. (2019, May). Model-Free Reinforcement Learning for Financial Portfolios: A Brief Survey [arXiv:1904.04973 [cs, q-fin, stat]]. https://doi.org/10.48550/arXiv.1904.04973

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2017, April). Trust Region Policy Optimization [arXiv:1502.05477 [cs]]. https://doi.org/10.48550/arXiv.1502.05477

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2018, October). High-Dimensional Continuous Control Using Generalized Advantage Estimation [arXiv:1506.02438 [cs]]. https://doi.org/10.48550/arXiv.1506.02438

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017, August). Proximal Policy Optimization Algorithms [arXiv:1707.06347 [cs]]. https://doi.org/10.48550/arXiv.1707.06347

Sharpe, W. F. (1964). Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk [_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1964.tb02865.x]. *The Journal of Finance*, *19*(3), 425–442. https://doi.org/10.1111/j.1540-6261.1964.tb02865.x

Sharpe, W. F. (1966). Mutual Fund Performance [Publisher: University of Chicago Press]. *The Journal of Business*, *39*(1), 119–138. Retrieved January 30, 2024, from https://www.jstor.org/stable/2351741

Silva, F. L. D., Taylor, M. E., & Costa, A. H. R. (2018). Autonomously Reusing Knowledge in Multiagent Reinforcement Learning. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 5487–5493. https://doi.org/10.24963/ijcai.2018/774

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, I–387–I–395.

Sood, S., Papasotiriou, K., Vaiciulis, M., & Balch, T. (2023). Deep Reinforcement Learning for Optimal Portfolio Allocation: A Comparative Study with Mean-Variance Optimization, 21.

Stanley, M. (2023, July). How Multi-Manager Platforms Find Strength in Numbers. Retrieved January 12, 2024, from https://www.morganstanley.com/im/en-nl/institutional-investor/insights/articles/how-multi-manager-platforms-find-strength-in-numbers.html

Sun, S., Qin, M., Wang, X., & An, B. (2022). PRUDEX-Compass: Towards Systematic Evaluation of Reinforcement Learning in Financial Markets. *Transactions on Machine Learning Research*. Retrieved December 7, 2023, from https://openreview.net/forum?id=JjbsIYOuNi

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems*, *12*. Retrieved February 2, 2024, from https://proceedings.neurips.cc/paper_files/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html

Swade, A., Hanauer, M. X., Lohre, H., & Blitz, D. (2023, October). Factor Zoo (.zip). https://doi.org/10.2139/ssrn.4605976

Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., & Vicente, R. (2015, November). Multiagent Cooperation and Competition with Deep Reinforcement Learning [arXiv:1511.08779 [cs, q-bio]]. Retrieved January 31, 2024, from http://arxiv.org/abs/1511.08779

Terry, J. K., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sullivan, R., Santos, L., Perez, R., Horsch, C., Dieffendahl, C., Williams, N. L., Lokesh, Y., & Ravi, P. (2021, October). PettingZoo: Gym for Multi-Agent Reinforcement Learning [arXiv:2009.14471 [cs, stat]]. https://doi.org/10.48550/arXiv.2009.14471

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., & Younis, O. G. (2023, March). Gymnasium. https://doi.org/10.5281/zenodo.8127026

Vijh, M., Chandola, D., Tikkiwal, V. A., & Kumar, A. (2020). Stock Closing Price Prediction using Machine Learning Techniques. *Procedia Computer Science*, *167*, 599–606. https://doi.org/10.1016/j.procs.2020.03.326

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., … Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning [Number: 7782 Publisher: Nature Publishing Group]. *Nature*, *575*(7782), 350–354. https://doi.org/10.1038/s41586-019-1724-z

Wang, H., & Zhou, X. Y. (2019, May). Continuous-Time Mean-Variance Portfolio Selection: A Reinforcement Learning Framework [arXiv:1904.11392 [cs, math, q-fin]]. https://doi.org/10.48550/arXiv.1904.11392

Xiong, Z., Liu, X.-Y., Shan Zhong, Zhong, S., Hongyang Yang, Yang, H., & Walid, A. (2018). Practical Deep Reinforcement Learning Approach for Stock Trading [MAG ID: 2901174038]. *arXiv: Learning*.

Yang, H., Liu, X.-Y., Zhong, S., & Walid, A. (2020). Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3690996

Yang, Y., & Wang, J. (2021, March). An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective [arXiv:2011.00583 [cs]]. https://doi.org/10.48550/arXiv.2011.00583

Ye, Y., Pei, H., Wang, B., Chen, P.-Y., Zhu, Y., Xiao, J., & Li, B. (2020). Reinforcement-Learning Based Portfolio Management with Augmented Asset Movement Prediction States. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(01), 1112–1119. https://doi.org/10.1609/aaai.v34i01.5462

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., & Wu, Y. (2022). The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *Advances in Neural Information Processing Systems*, *35*, 24611–24624. Retrieved December 14, 2023, from https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract-Datasets_and_Benchmarks.html

Zhang, K., Yang, Z., Liu, H., Zhang, T., & Basar, T. (2018). Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents [ISSN: 2640-3498]. *Proceedings of the 35th International Conference on Machine Learning*, 5872–5881. Retrieved January 4, 2024, from https://proceedings.mlr.press/v80/zhang18n.html

A

IPPO Hyperparameters

Hyperparameters used for the BenchMARL IPPO model are shown in Table A.1. Unmentioned hyperparameters are the set as the default hyperparameters.

Table A.1: IPPO hyperparameters used in experiments

Category	Hyperparameters	Values Used
Optimizer(SGD)	Learning Rate	0.0006
	Minibatch Size	64
	Minibatch number of iterations	2
Nerual network	Number of layers	2
	Size of layers	64
	Random seeds	0,1,2,3
PPO	Entropy coefficient	0.035
	Batch size	8080
	n_timesteps	1454400
Diversity	Type of diversity penalty	Correlation, TV Distance
	Weight of the penalty	discussed in 4

B

Results

Table B.1: Portfolio statistics of IPPO agents, trained with four random seeds $[0, 1, 2, 3]$ and 10 different Correlation diversity penalty weights λ.

Seed		0				1				2				3			
Cor	**Agent**	**Return**	**Sharpe**	**MDD**	**Turnover**	**Return**	**Sharpe**	**MDD**	**Turnover**	**Return**	**Sharpe**	**MDD**	**Turnover**	**Return**	**Sharpe**	**MDD**	**Turnover**
0	A	1.0036	0.5217	-0.1295	0.3598	1.0064	0.5269	-0.0869	0.3519	0.9729	0.5150	-0.1041	0.3287	0.9833	0.5186	-0.1236	0.3436
0	B	0.9600	0.5140	-0.1150	0.3487	1.0292	0.5324	-0.1194	0.3429	1.0048	0.5238	-0.1370	0.3560	0.9823	0.5139	-0.1453	0.3409
0.1	A	0.9839	0.5209	-0.1173	0.3491	1.0818	0.5520	-0.1022	0.3386	0.9634	0.5117	-0.1070	0.3426	1.0325	0.5344	-0.1314	0.3422
0.1	B	0.9744	0.5173	-0.1344	0.3498	1.0242	0.5294	-0.1120	0.3368	1.0290	0.5326	-0.1333	0.3503	1.0211	0.5263	-0.1427	0.3360
0.2	A	1.0108	0.5277	-0.1150	0.3458	1.0412	0.5374	-0.1188	0.3355	0.9840	0.5176	-0.1140	0.3299	1.0367	0.5344	-0.1427	0.3403
0.2	B	0.9892	0.5216	-0.1313	0.3355	1.0369	0.5338	-0.1090	0.3280	1.1067	0.5593	-0.1056	0.3430	1.0469	0.5368	-0.1003	0.3396
0.3	A	1.0632	0.5457	-0.0871	0.3334	1.0590	0.5437	-0.1142	0.3296	0.9983	0.5224	-0.1212	0.3309	1.0286	0.5304	-0.1303	0.3333
0.3	B	1.0167	0.5300	-0.1317	0.3347	1.0017	0.5213	-0.1238	0.3251	1.1232	0.5646	-0.0935	0.3369	1.0161	0.5252	-0.1474	0.3385
0.4	A	1.0283	0.5365	-0.0791	0.3373	1.0556	0.5466	-0.0837	0.3287	1.0138	0.5239	-0.1251	0.3156	1.0386	0.5327	-0.1390	0.3255
0.4	B	0.9778	0.5191	-0.1229	0.3232	1.0619	0.5418	-0.1033	0.3270	1.1235	0.5649	-0.1005	0.3347	0.9961	0.5211	-0.1125	0.3221
0.5	A	1.0408	0.5390	-0.0800	0.3221	1.0816	0.5543	-0.0810	0.3225	1.0002	0.5227	-0.1348	0.3159	1.0633	0.5432	-0.1255	0.3240
0.5	B	1.0013	0.5258	-0.1210	0.3212	1.0274	0.5310	-0.1179	0.3118	1.0757	0.5511	-0.0889	0.3202	0.9793	0.5150	-0.1382	0.3141
0.6	A	1.0080	0.5259	-0.1154	0.3214	1.0512	0.5461	-0.0774	0.3157	0.9968	0.5219	-0.1141	0.3159	1.0809	0.5454	-0.1421	0.3223
0.6	B	0.9673	0.5173	-0.1305	0.3158	0.9528	0.5065	-0.1351	0.3103	1.0841	0.5554	-0.0790	0.3150	1.0006	0.5228	-0.1332	0.3100
0.7	A	1.0681	0.5470	-0.1093	0.3033	1.0871	0.5550	-0.0904	0.3130	0.9840	0.5159	-0.1249	0.3124	1.0158	0.5264	-0.1401	0.3171
0.7	B	1.0205	0.5331	-0.1138	0.3098	1.0084	0.5251	-0.1282	0.2991	1.1165	0.5624	-0.1024	0.3001	1.0117	0.5248	-0.1208	0.3023
0.8	A	1.0749	0.5469	-0.1102	0.2995	1.0975	0.5607	-0.0751	0.3124	0.9702	0.5138	-0.1242	0.2955	1.1107	0.5557	-0.1319	0.3114
0.8	B	0.9850	0.5204	-0.1283	0.2996	1.0048	0.5245	-0.1326	0.2836	1.0488	0.5471	-0.0705	0.2970	0.9909	0.5198	-0.1259	0.2979
0.9	A	1.0977	0.5581	-0.0960	0.2902	1.0894	0.5557	-0.0768	0.3005	0.9947	0.5227	-0.1074	0.2939	1.0928	0.5519	-0.1214	0.2986
0.9	B	0.9995	0.5267	-0.1202	0.2937	1.0292	0.5346	-0.1092	0.2849	1.0980	0.5589	-0.0854	0.2868	0.9878	0.5187	-0.1221	0.2929

Table B.2: Portfolio statistics of IPPO agents, trained with four random seeds $[0, 1, 2, 3]$ and 10 different TV Disatnce diversity penalty weights λ.

Seed		0				1				2				3				Mean	Std
Cor	Agent	Return	Sharpe	MDD	Turnover	Return	Sharpe	MDD	Turnover	Return	Sharpe	MDD	Turnover	Return	Sharpe	MDD	Turnover	Returns	Returns
0	A	1.0036	0.5217	-0.1295	0.3598	1.0064	0.5269	-0.0869	0.3519	0.9729	0.5150	-0.1041	0.3287	0.9833	0.5186	-0.1236	0.3436	0.9915	0.0162
0	B	0.9600	0.5140	-0.1150	0.3487	1.0292	0.5324	-0.1194	0.3429	1.0048	0.5238	-0.1370	0.3560	0.9823	0.5139	-0.1453	0.3409	0.9941	0.0297
0.1	A	1.0143	0.5295	-0.1011	0.3530	1.0443	0.5374	-0.1104	0.3490	0.9227	0.4975	-0.1224	0.3439	1.0325	0.5291	-0.1168	0.3465	1.0154	0.0529
0.1	B	1.0162	0.5302	-0.1220	0.3461	1.0143	0.5248	-0.1185	0.3453	1.0723	0.5471	-0.1164	0.3609	1.0211	0.5346	-0.1295	0.3456	1.0122	0.0254
0.2	A	0.9989	0.5242	-0.1143	0.3620	1.0567	0.5429	-0.0952	0.3581	0.9774	0.5147	-0.1296	0.3443	1.0367	0.5351	-0.1282	0.3543	1.0182	0.0264
0.2	B	1.0379	0.5368	-0.1205	0.3556	1.0118	0.5254	-0.1159	0.3472	1.0442	0.5398	-0.1000	0.3606	1.0469	0.5178	-0.1490	0.3454	1.0449	0.0483
0.3	A	1.0707	0.5467	-0.1057	0.3555	1.0989	0.5558	-0.0931	0.3567	0.9644	0.5122	-0.1188	0.3454	1.0286	0.5359	-0.1124	0.3518	1.0373	0.0302
0.3	B	1.0013	0.5263	-0.1307	0.3545	1.0216	0.5312	-0.0881	0.3441	1.1219	0.5635	-0.0933	0.3576	1.0161	0.5283	-0.1431	0.3470	1.0394	0.0563
0.4	A	1.0269	0.5318	-0.0919	0.3599	1.0405	0.5376	-0.1115	0.3486	1.0083	0.5257	-0.1205	0.3429	1.0386	0.5302	-0.1347	0.3448	1.0341	0.0176
0.4	B	0.9844	0.5213	-0.1268	0.3548	1.0730	0.5444	-0.1036	0.3430	1.1193	0.5640	-0.0913	0.3504	0.9961	0.5353	-0.1173	0.3393	1.0398	0.0664
0.5	A	1.0293	0.5337	-0.1037	0.3638	1.0857	0.5536	-0.0959	0.3545	1.0212	0.5303	-0.1174	0.3428	1.0633	0.5317	-0.1361	0.3488	1.0465	0.0351
0.5	B	1.0180	0.5345	-0.0954	0.3525	1.0268	0.5327	-0.0964	0.3397	1.0931	0.5533	-0.1087	0.3513	0.9793	0.5223	-0.1440	0.3385	1.0209	0.0415
0.6	A	1.0229	0.5296	-0.1234	0.3512	1.1044	0.5560	-0.0983	0.3606	1.0123	0.5272	-0.1256	0.3449	1.0809	0.5144	-0.1534	0.3505	1.0342	0.0390
0.6	B	1.0304	0.5355	-0.1245	0.3506	1.0125	0.5286	-0.0991	0.3381	1.0818	0.5513	-0.1063	0.3465	1.0006	0.5295	-0.1433	0.3367	1.0012	0.0587
0.7	A	1.0690	0.5434	-0.1219	0.3515	1.0636	0.5466	-0.0884	0.3512	1.0433	0.5353	-0.1281	0.3450	1.0158	0.5258	-0.1067	0.3420	1.0388	0.0473
0.7	B	1.0086	0.5282	-0.1302	0.3522	1.0535	0.5379	-0.1108	0.3331	1.1392	0.5675	-0.1052	0.3436	1.0117	0.5205	-0.1366	0.3432	1.0393	0.0517
0.8	A	1.0523	0.5385	-0.1198	0.3503	1.0953	0.5567	-0.0803	0.3535	1.0149	0.5269	-0.1279	0.3388	1.1107	0.5280	-0.1230	0.3352	1.0633	0.0638
0.8	B	0.9931	0.5242	-0.1292	0.3470	1.0201	0.5304	-0.0878	0.3262	1.0545	0.5419	-0.1200	0.3494	0.9909	0.5036	-0.1549	0.3386	1.0074	0.0288
0.9	A	1.0418	0.5368	-0.1253	0.3477	1.0711	0.5475	-0.0856	0.3541	1.0121	0.5277	-0.1234	0.3317	1.0928	0.5280	-0.1230	0.3352	1.0686	0.0494
0.9	B	1.0168	0.5307	-0.1336	0.3415	1.0364	0.5368	-0.0882	0.3382	1.1225	0.5652	-0.0917	0.3437	0.9878	0.5036	-0.1549	0.3386	1.0286	0.0494