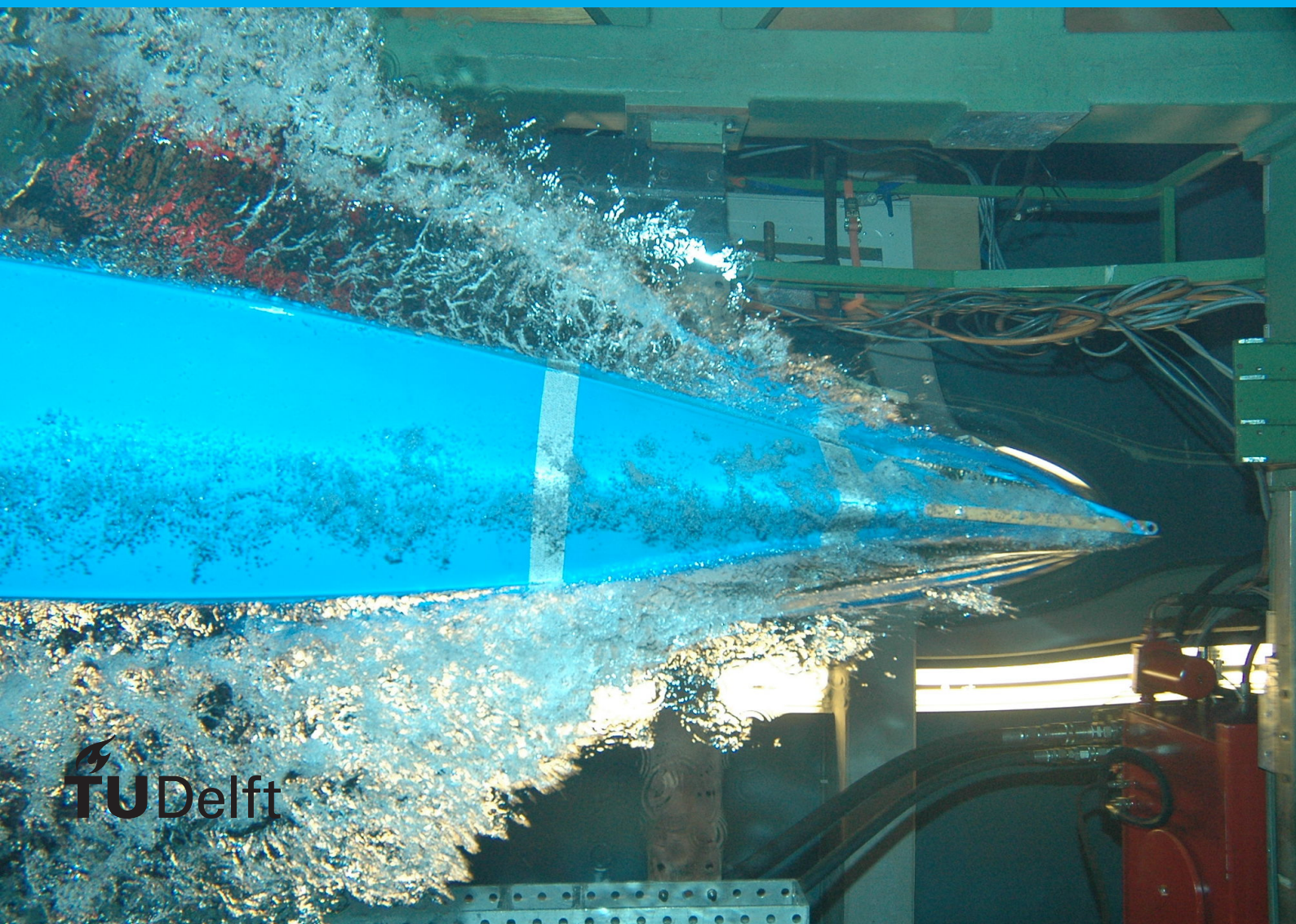
Risk-sensitive Reinforcement Learning for Portfolio Allocation

Aranya Sinha



Risk-sensitive Reinforcement Learning for Portfolio Allocation

by

Aranya Sinha

to obtain the degree of Master of Science, Computer Science
at the Delft University of Technology,
to be defended publicly on Wednesday September 18, 2024 at 9:00 AM.

Student number:	5774683	
Project duration:	December 1, 2023 – September 18, 2024	
Thesis committee:	Chair: Dr. Frans Oliehoek	Associate Professor
	Core member 2: Dr. Luciano Cavalcante Siebert	Assistant Professor (Greenlist)
	Core Member 3: Dr. Antonis Papapantoleon	Full Professor
	Member 4: Dr. Mustafa Mert Çelikok	Post-doctoral Researcher
	External Member: Rob Huisman	Company Supervisor (Robeco)

TU Delft

Abstract

This study explores the application of risk-sensitive Reinforcement Learning (RL) in portfolio optimization, aiming to integrate asset pricing and portfolio construction into a unified, end-to-end RL framework. While RL has shown promise in various domains, its traditional risk-neutral approach is unsuitable for financial contexts where risk sensitivity is crucial. This research focuses on risk-sensitive RL methods that incorporate different risk measures to manage uncertainty and volatility in financial markets better. The project extends existing RL techniques by adapting the cross-sectional approach to risk-sensitive settings and introducing new variants like PPO-CVaR and PPO-Expectile for portfolio management. A comparative study of these methods is conducted to assess their performance with real market data and simulated environments. The research addresses key questions related to how different risk measures impact learned portfolio strategies, the influence of risk appetite on decision-making, and the performance gap between simulated and real market data. The findings aim to provide insights for practitioners looking to implement risk-sensitive RL in financial asset management.

Contents

1 Introduction

Reinforcement Learning (RL) is a framework for addressing sequential decision-making problems where an agent learns to interact with an environment to maximize cumulative rewards over time. This process involves the agent receiving feedback from its actions through rewards, which guide it to learn the optimal behavior, also known as the optimal policy. Deep Reinforcement Learning (Deep RL) extends traditional RL by leveraging deep neural networks to handle high-dimensional input spaces and actions, allowing for learning more complex policies directly from data. This approach has led to significant advancements in various fields, including game-playing, robotics, and autonomous driving. Recent notable use-cases of Deep RL include the development of AlphaGo by DeepMind [1], which surpassed human experts in the game of Go, OpenAI's Dota 2 bot, which demonstrated advanced planning and real-time decision-making in a complex multiplayer game environment [2], and fine-tuning large language models such as ChatGPT to align with human behaviour[3].

Portfolio optimization is a critical task in financial management. A portfolio consists of assets (e.g., stocks, bonds, currency)[4]. The portfolio optimizer distributes a given cash budget between these assets, essentially changing their proportions (also called *weights*) in the portfolio. The objective of the optimization is to maximize the portfolio's return while considering the risks. This is often formulated as either achieving maximum expected return within a specified level of risk or minimizing risk for a specified level of expected return.

In this project, the focus is on an important variant of portfolio optimization, called *multi-period portfolio optimization*. Unlike the single-period variant, the objective here is to reallocate the assets at multiple time steps, adjusting their weights dynamically. This way, the portfolio optimizer is more adaptive and can account for changes in market dynamics. Traditional approaches to multi-period portfolio optimization often rely on stochastic processes and control theory to model financial markets and derive optimal strategies. For instance, Mean-Variance Optimization (MVO) is a widely used technique that balances the trade-off between risk and return. These models are valued for their simplicity and explanatory power, which makes them suitable for matching regulatory requirements and being accountable to clients. However, their most significant downside is that they can lead to sub-optimal strategies and financial losses due to their simplified assumptions about market behavior.

The recent surge in data availability and advancements in computational power have paved the way for data-driven algorithms to address decision-making problems in finance more effectively. Reinforcement Learning (RL) is one such well-suited group of algorithms due to its ability to learn optimal policies through interaction with the environment rather than relying on predefined models of market dynamics. This shift is particularly advantageous when dealing with the complex nature of financial markets, where traditional models can be inadequate. Model-free RL algorithms, which many modern Deep RL algorithms fall under, bypass the need for explicit system dynamics modeling and instead focus on learning from the agent's interactions with the market. This method can uncover adaptive and robust strategies, potentially leading to better performance in portfolio management tasks.

This project investigates the application of reinforcement learning (RL) to portfolio optimization. However, standard RL is risk-neutral, which is not always suitable for portfolio optimization problems where risk sensitivity is an important parameter. Therefore, we focus on a subset of RL methods called *risk-sensitive RL*. We aim to combine the traditionally separate processes of asset pricing and portfolio

construction into a unified, end-to-end, risk-sensitive RL framework. Asset pricing can be thought of as identifying which factors contribute to the changes in the market prices of an asset and how. In that sense, this can be seen as equivalent to feature engineering. Portfolio construction then uses the features by predicting price changes and constructing portfolios accordingly, which is prediction and decision-making. Performing these tasks has been possible in many RL environments through end-to-end representation learning and policy optimization. However, this is particularly challenging in financial markets due to their uncertain nature. In addition, most RL tasks have an agreed-upon reward function, which the agent should optimize. In portfolio optimization, a good reward function is an open question, making reward design a challenging task in itself.

Managing uncertainty—whether originating from the model or the environment—is crucial across many applications in RL. Unlike the risk-neutral approaches, risk-sensitive RL considers dealing with uncertainty in a broader spectrum than expectation, such as learning risk-averse policies that minimize risk measures like conditional value at risk instead of expected value. These approaches often alter the agent's objective, such as optimizing to minimize the risk instead of maximizing returns. This approach is useful, for example, in financial asset management, where we must consider the risk associated with investments, or in robotics, where navigating uncertain environments safely is essential. Unfortunately, implementing risk-sensitive RL is often non-trivial in practice, as the recursive Bellman operator often does not hold. Handling this issue requires significant modifications to standard reinforcement learning algorithms or developing surrogate objectives.

This project performs a comparative study of risk-sensitive RL methods on the portfolio optimization problem. While various applications of risk-sensitive RL to portfolio optimization exist, to our knowledge, a comparative study has not been conducted before. In addition, we also extend the comparative study to other risk-sensitive RL methods previously not used for portfolio optimization or finance in general, such as Jiang, Liu, Ma, *et al.* [5] and Marzban, Delage, and Li [6]. Since each risk-sensitive RL method uses a different approach to achieve risk sensitivity, it is important to have a comparative study of their performances and learned portfolio strategies in ortoss their strengths and weaknesses in a financial context. The results of this project serve as a starting point for practitioners looking into applying risk-sensitive RL to their portfolio management problems.

The project starts with Aboussalah, Xu, and Lee [7] as the (risk-neutral) RL baseline, referred to here as the cross-sectional (CS) approach. We chose this work because the CS feature extractor and reward function are grounded in financial principles. Implementing the baseline is followed by implementing the different risk-sensitive RL methods on top of the CS approach. The general methodology for this is to extend the objective function of the CS approach from optimizing expected returns to optimizing the expected value of risk measures. The project also benchmarks against traditional non-RL portfolio optimization methods like Mean-Variance Optimization (MVO) and Equal-Weights-Buy (EQB). Different RL approaches to portfolio optimization model the task with different reward functions. Therefore, we use financial metrics such as portfolio return, Sharpe ratio, and maximum drawdown to ensure a fair comparison. This comparison is also more realistic since these metrics are, in the end, what portfolio managers care about.

A significant challenge in this project is evaluating the methods effectively, particularly whether to use market data or simulators. Risk-neutral methods like the baseline CS approach are typically evaluated with market data[7]–[9]. However, Deep RL algorithms are often sample inefficient, and many existing risk-sensitive RL methods demonstrate performance through simulators [6], [10]. For many methods, this is because estimating the risk requires multiple samples from the environment [11]. The discrepancy between how risk-neutral and risk-sensitive approaches are evaluated can present a danger to practitioners. Good risk performance in simulation does not necessarily imply good performance in real market conditions. To have an effective comparison in this project, the initial comparison will utilize simulations, followed by exploring how these methods perform with real market data. The discrepancy between market data and simulation conditions is expected to affect performance. While simulations offer unlimited episodes, market data represents a continuous, singular episode. The results suggest a notable Sim2Real gap, highlighting the need for caution among practitioners when transitioning from simulated to real-world applications like portfolio optimization.

1.1. Contributions

This study investigates various risk-sensitive RL methods and compares them to explore the impact of different risk measures in a data-driven Deep Reinforcement Learning (RL) framework for portfolio optimization. While prior work, such as Pacheco Aznar [12], focused on risk measures within distributional RL for portfolio optimization, this research works with the cross-sectional approach[7] and evaluates dynamic time-consistent risk measures[10] and the risk-adjusted Bellman operator [5]. The main contributions are as follows:

- The cross-sectional approach [7] is extended to risk-sensitive RL for portfolio optimization
- A generic actor-critic algorithm that uses CVaR (CE-AC)[10] is modified to a novel variant of PPO that is risk-sensitive, which we call PPO-CVaR.
- Two risk-sensitive RL methods that use the same risk measure, DDPG-Expectile[6] and PPO-Expectile[5], are applied to the portfolio optimization problem for the first time. Their performances are evaluated and compared.
- The performances of two risk-sensitive RL algorithms with different risk measures, PPO-CVaR and PPO-Expectile, are tested and compared in both real market data and simulations.

1.2. Research Question

The main research question addressed in this study is: *How do the risk-sensitive variants of a cross-sectional approach to portfolio optimization using Deep Reinforcement Learning perform with market data and simulated prices?* The main question is investigated through three sub-questions. Since risk-sensitivity can be achieved with respect to different risk measures, a natural first step is to examine how this choice affects results. Each risk-sensitive RL approach has a parameter that controls the risk appetite of the decision-maker. These parameters and their interplay with different risk measures are other important factors investigated in the second sub-question. Finally, the majority of the risk-sensitive RL approaches evaluate the risk performance in simulated environments since accurate risk estimation can require many samples. Whether simulation results extend to real portfolio optimization tasks or not is an important question, which is the third sub-question below.

- **RQ1:** What is the impact of employing different risk measures on portfolio strategies learned by the RL agent?
- **RQ2:** How does adjusting the parameters of risk measures influence the risk appetite of the learned portfolio strategies?
- **RQ3:** What is the performance gap between applying these methods to simulated problems and applying them to real market data?

Different risk measures may disagree on how the risk of certain events are quantified. Therefore, one would expect that the portfolio investment strategy learned from different measures will also be distinct. A reasonable hypothesis for **(RQ1)** is then given by **H1** below. Each risk measure comes with a parameter that controls the risk sensitivity of the RL algorithm (e.g., conservative, risk-seeking, risk-neutral). A reasonable hypothesis for **(RQ2)** is then given by **H2**, where a similar relationship is observed in the portfolio strategies. Finally, since the real market data represents a single episode, the risk estimates are unlikely to be as accurate as simulations with thousands of episodes. Thus for **(RQ3)**, the hypothesis **H3** is a reasonable expectation.

- **H1:** Different risk measures result in distinct portfolio optimization strategies.
- **H2:** Adjusting risk measure parameters alters the risk appetite, leading to more conservative policies or strategies with lower risk and returns.
- **H3:** Risk-sensitive methods performing well in simulations does not imply the ability of similar learning performance in real market data.

1.3. Outline

This report is organized into six chapters, along with an appendix. Following the introduction, Chapter 2 delves into the background knowledge necessary for understanding the project. Chapter 3 provides a comprehensive overview of the relevant literature on risk-sensitive reinforcement learning (RL) and portfolio optimization. The methodology employed in the project, including both risk-neutral and risk-sensitive RL algorithms, is detailed in Chapter 4. Chapter 5 outlines the experiments conducted, including their design and the results obtained. Finally, Chapter 6 presents the discussion of the findings and the conclusions drawn from the study.

2

Background

This chapter covers the relevant background knowledge for the project. In Section 2.1, we introduce reinforcement learning and the relevant Deep RL algorithms used in the project. Then, section 2.2 introduces the portfolio optimization framework, and section 2.3 introduces the framework for applying RL to portfolio optimization. Section 2.4 covers risk measures, their properties, and the two risk measures used in this project: CVaR and Expectile. Section 2.5 introduces risk-sensitive RL and the relevant background concepts of dynamic time-consistent risk and elicitable functions, followed by the expectile Bellman operator. Elicitable functions are the theoretical basis for the PPO-CVaR and DDPG-Expectile methods and the expectile Bellman operator for PPO-Expectile. Chapter 4 covers the specifics of the RL algorithms that use these risk measures: PPO-CVaR, PPO-DDPG, and PPO-Expectile. Finally, Section 2.6 introduces the relevant deep learning concepts of this project.

2.1. Reinforcement Learning

The Reinforcement Learning (RL) framework is modeled by a standard Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$ [13]. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state space and the action space, respectively. The transition probability function is $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, which describes the probability of transitioning from one state to another given a specific action. The reward function is denoted as $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the discount factor $\gamma \in (0, 1)$ represents the importance of future rewards.

In the RL framework, an agent interacts with an environment. At each discrete time step, the agent perceives the current state of the environment and executes an action accordingly. Subsequently, the agent receives a reward signal from the environment, indicating the desirability of the current state. The agent's primary objective is to maximize its cumulative reward, denoted as the return (G_t), which is the discounted sum of rewards:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is defined as the conditional probability of taking action a in state s, denoted by $\pi(a \mid s)$. It is a mapping that dictates the agent's actions given a specific state and can be either deterministic or stochastic. A deterministic policy is a special case where $\pi : \mathcal{S} \to \mathcal{A}$.

The state value function $V^\pi : \mathcal{S} \to \mathbb{R}$ under a policy π is defined as the expected return starting from state s and following policy π:

$$V^\pi(s) = \mathbb{E}^\pi \left[G_t \mid S_t = s \right]$$

The state-action value function (Q-value function) $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ under a policy π is defined as the expected return starting from state s, taking action a, and thereafter following policy π:

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

The advantage function $A(s_t, a_t)$ represents the relative benefit of taking action a_t in state s_t compared to the average value of the state, and is defined as:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

with $Q(s_t, a_t)$ and $V(s_t)$ being the Q value function and the value function, respectively.

Deep RL implements the value and policy functions as neural networks. This allows the agent to approximate these functions for large state and/or action spaces and to learn from observed experiences. Policy-based methods, such as policy gradient methods, aim to optimize the policy directly by adjusting the parameters of the policy network in the direction that maximizes the expected return. Value-based methods, on the other hand, focus on learning the value function (V or Q), with the policy subsequently derived from it using methods such as the ϵ-greedy approach.

Actor-critic algorithms combine elements of both approaches, employing a dual framework where a critic network learns the value function while an actor-network learns the policy function. The critic evaluates the action taken by the actor and provides feedback to improve the policy. Additionally, RL algorithms can be categorized based on whether they operate on-policy or off-policy and whether they learn stochastic or deterministic policies, leading to a diverse range of algorithms. The standard actor-critic Deep RL algorithms used in this project are the on-policy and stochastic Proximal Policy Optimization (PPO) [14] and the off-policy and deterministic Deep Deterministic Policy Gradient (DDPG) [15].

2.1.1. PPO

Proximal Policy Optimization (PPO)[14] is an RL algorithm based on trust region methods, particularly inspired by Trust Region Policy Optimization (TRPO)[16]. PPO aims to maintain the stability of policy updates by optimizing a clipped surrogate objective function, which prevents large deviations from the previous policy. The primary objective function in PPO is defined as:

$$L(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

In this equation, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ represents the probability ratio between the new policy π_θ and the old policy $\pi_{\theta_{\text{old}}}$. $\hat{\mathbb{E}}_t$ is the empirical average on a finite batch of samples and $\hat{A}_t$ the estimate of the advantage at time t. The clipping function $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ constrains the policy updates to ensure that they remain within a specified range, where ϵ is a hyperparameter that controls the extent of clipping.

2.1.2. DDPG

Deep Deterministic Policy Gradient (DDPG)[15] is an actor-critic algorithm tailored for continuous action spaces. It employs two neural networks: a critic network that estimates the Q-value function $Q^\psi(s, a)$ and an actor-network that represents the (deterministic) policy $\mu^\theta(s)$, which outputs a specific action given a state. The critic network is updated by minimizing the Bellman error, defined as:

$$L(\psi) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim D} \left[\left(r_t + \gamma Q^\psi(s_{t+1}, \mu^\theta(s_{t+1})) - Q^\psi(s_t, a_t) \right)^2 \right]$$

Here, r_t is the reward obtained after executing action a_t in state s_t, γ is the discount factor, and D is the experience replay buffer. The actor-network is updated using the deterministic policy gradient, which optimizes the expected return by taking the gradient of the critic network's output with respect to the actor's parameters:

$$\nabla_\theta J \approx \mathbb{E}_{s_t \sim D} \left[\nabla_a Q^\psi(s, a)|_{a=\mu^\theta(s)} \nabla_\theta \mu^\theta(s) \right]$$

This gradient adjusts the actor's parameters in the direction that maximizes the Q-value estimated by the critic network. Additionally, DDPG uses techniques such as target networks and soft updates to stabilize training and reduce the variance of the updates.

2.2. Portfolio Optimization

As outlined in chapter 1, this project concerns the multi-period portfolio optimization problem. The formulation of the problem in the context of this project is as follows:

Consider a financial market comprising of n risky assets. An investor begins with an initial wealth of x_0 and aims to reallocate this wealth among the n assets at each time step t to achieve some goal, such as maximizing the return. At each time step, random rates of return for the assets are given by

$\mathbf{r_t} = (r_t^1, ..., r_t^n)$ and the amount invested in asset i by $a_t^i (i = 1, ..., n)$. An investment strategy is defined as a sequence of portfolio weights $\{\mathbf{a}_t\}_{t=0}^{T-1}$, and the goal is to obtain the optimal investment strategy for the goal.

A well-established approach to solving this problem is Mean-Variance Optimization (MVO), as described in Li and Ng [4]. MVO seeks to maximize the expected return, quantified by the mean, while constraining the portfolio risk, measured by the variance. This approach leads to the following optimization problem:

$$\max_{\{\mathbf{a_t}\}_{t=0}^{T-1}} \quad \mathbb{E}[x_T] - \phi Var(x_T), \tag{2.1}$$

subject to, (2.2)

$$x_{t+1} = \sum_{i=1}^{n} a_t^i r_t^i, \quad t = 0, 1, ..., T-1 \tag{2.3}$$

Here, ϕ is a parameter that adjusts the trade-off between risk and return. As demonstrated in Li and Ng [4], this optimization problem can be reformulated as a Linear-Quadratic (LQ) problem, for which an analytical solution is available.

2.3. Reinforcement Learning for Portfolio Optimization

Portfolio Optimization (PO) is traditionally approached as a mathematical optimization problem aimed at maximizing returns or minimizing risks given a set of constraints. Framing PO as an RL problem introduces a dynamic, adaptive paradigm in which an agent interacts with a financial market over multiple periods. The multi-period portfolio optimization problem [4] can be reformulated as a discrete-time Markov Decision Process (MDP), where the system being controlled is a portfolio consisting of multiple assets, and the actions are the portfolio weights. In this RL formulation, the agent observes the current state of the market, typically represented by price information of assets in the portfolio, receives some reward (such as the one-step profit or loss), and decides on new portfolio weights based on this information. In portfolio optimization, neither the future returns nor the state transition probabilities are known, making model-free RL a solid choice for this application.

The **state** representation in this setup comprises the historical price information concatenated with the previous actions taken by the agent. The exact details depend on the RL method and will be explained in chapters 4 and 5.

The **action** taken by the agent at each time step t are continuous actions that correspond to new portfolio weights, denoted as $\mathbf{a}_t = (a_{t,1}, ..., a_{t,n})$, where n is the number of assets in the portfolio.

The **reward** function varies based on the specific financial objectives and risk preferences. A common approach is to use a variant of the log rate-of-return.

In standard RL environments, the desired behavior is usually well-defined, and there is an agreed-upon reward function that induces it. However, the same is not true for portfolio optimization. There is no objectively best reward function to determine the investment strategy, but instead, different strategies based on different approaches or goals such as MVO[4], risk parity[17], and low volatility[18]. This means there is no well-defined objective or reward function, and the reward design is one of the biggest challenges of using RL for portfolio optimization.

Similarly, this also means there is no objectively best metric to determine how well a portfolio has performed. While the cumulative reward is usually a good metric to indicate the performance of an agent when it comes to portfolio optimization, it is only used as a complementary metric to - for example - indicate if the agent is learning or not. Instead, we use standard finance metrics such as Sharpe Ratio, Final Portfolio Value, and Maximum DrawDown. These are described in detail in section 5.1.1.

2.4. Risk Measures

When it comes to financial decision-making under uncertainty, a static risk measure is a fundamental tool used to assess the risk associated with random variables within a given probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Formally, it is represented as a mapping $\rho : \mathcal{Y} \to \mathbb{R}$, where $\mathcal{Y}$ denotes the set of relevant random variables, and $\rho(Y)$ quantifies the level of risk attributed to $Y \in \mathcal{Y}$ [10].

Over the years, certain properties of risk measures have been identified in financial literature that make them suitable for working with financial applications[19]. One of the fundamental types of risk measures, monetary risk measures, should satisfy the following two properties:

- **Monotonicity:** If $X < Y$ (where X and Y are random variables), then $\rho(X) < \rho(Y)$. This property signifies that greater costs are associated with higher levels of risk.
- **Translation invariance:** $\rho(Y+m) = \rho(Y)+m$ for any constant m. Translation invariance implies that the absolute level of risk is unaffected by shifts in the deterministic components of costs.

Furthermore, a monetary risk measure is **convex** if it also satisfies:

- **Convexity**: For any $\lambda \in [0,1], \rho(\lambda X + (1-\lambda)Y) \leq \lambda\rho(X) + (1-\lambda)\rho(Y)$,

and it is **coherent** if it is convex and

- **Positive homogeneous**: For any $\alpha \geq 0, \rho(\alpha Y) = \alpha\rho(Y)$.

These properties are foundational as they establish clear guidelines for evaluating risk consistently across different scenarios. In financial environments, they offer strong theoretical foundations and useful applications for risk management and optimization[19].

The risk measures used for this project are CVaR and Expectile. They are chosen because they are considered suitable monetary risk measures due to their theoretical properties, such as being convex and coherent [20], [21]. They are also elicitable (or conditionally elicitable) functions, which significantly helps when it comes to estimating them in a Deep RL setting. Elicitable functions are explained further in section 2.5.1.

VaR and CVaR: Value at Risk (VaR) is a widely used risk measure that quantifies the maximum possible loss at a specified confidence level α. For a random variable X representing the loss, the VaR at level α is defined as the quantile of the distribution:

$$\text{VaR}_\alpha(X) = \inf\{x \in \mathbb{R} \mid F_X(x) \geq \alpha\},$$

where $F_X(x) = \mathbb{P}(X \leq x)$ is the cumulative distribution function (CDF) of X.

In contrast, Conditional Value at Risk (CVaR) measures the expected loss given that the loss exceeds the VaR threshold. CVaR is particularly sensitive to the tail behavior of the distribution and is considered a better risk measure than VaR. It is defined as:

$$\text{CVaR}_\alpha(X) = \mathbb{E}[X \mid X \geq \text{VaR}_\alpha(X)],$$

CVaR thus captures the expected loss in the worst $1-\alpha$ fraction of cases, being more sensitive to the tail behavior of the distribution. This is explained further in Figure 2.1[1].

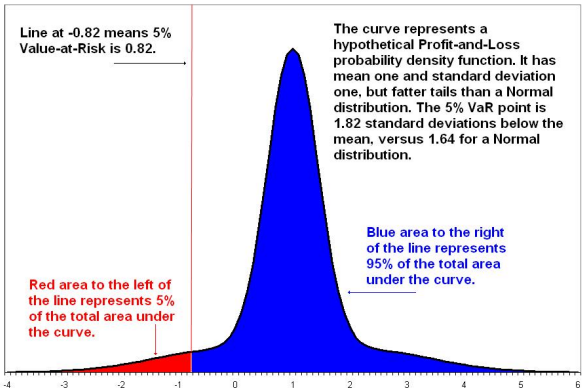


Figure 2.1: VaR and CVaR

[1]AaCBrown, Public domain, via Wikimedia Commons

Expectile: Expectile is an elicitable and coherent risk measure and an alternative to VaR and CVaR. However, it is less intuitive to interpret compared to VaR and CVaR[21]. Unlike quantiles, expectiles are defined as the solution to an optimization problem that involves minimizing an asymmetric loss function. For a given $\tau \in (0, 1)$, the τ-expectile $e_\tau(X)$ is defined as:

$$e_\tau(X) = \arg\min_{x \in \mathbb{R}} \tau \mathbb{E}[[X - x]_+^2] + (1 - \tau)\mathbb{E}[[X - x]_-^2]$$

where $[\cdot]_+ = \max(\cdot, 0)$ and $[\cdot]_- = \min(\cdot, 0)$. Expectiles can be interpreted as an asymmetric generalization of the mean.

2.5. Risk-sensitive RL

While section 2.4 introduced static risk measures, in a decision-making problem such as portfolio optimization where the market dynamics can change with time, accounting for risk at each time step is considered a better approach [11], [22].

Thus, in this section we expand to the dynamic setting. Following Ruszczyński [22], let $\mathcal{T} := \{0, \dots, T\}$ denote the sequence of periods. A **dynamic risk measure** is a sequence of risk measures $\{\rho_{t,T}\}_{t \in \mathcal{T}}$, where each $\rho_{t,T}$ satisfies the monotonicity property. These mappings $\rho_{t,T}$ represent the acceptable charges at time t instead of future costs.

An essential characteristic of dynamic risk measures is **time-consistency**, which asserts that for any $t \in T$ if $\rho_{t+1}(X) \geq \rho_{t+1}(Y)$, then $\rho_t(X) \geq \rho_t(Y)$. This property enables the formulation of recursive risk measures [23], defined as:

$$\rho_{t,T}(Y_t, \dots, Y_T) = Y_t + \rho_t\left(Y_{t+1} + \rho_{t+1}\left(Y_{t+2} + \cdots + \rho_{T-2}\left(Y_{T-1} + \rho_{T-1}(Y_T)\right) \dots\right)\right)$$

The recursive nature of dynamic risk measures means they can be used easily in an MDP framework.

In RL, the objective is to maximize the cumulative rewards. This can be equivalently stated as minimizing the cumulative cost. In risk-sensitive RL, the optimization objective changes from minimizing cumulative costs to minimizing the cumulative risk of the costs:

$$\min_\pi \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t c\left(s_t, a_t, s_{t+1}\right)\right]$$

to

$$\min_\pi \rho\left(\sum_{t=0}^{T-1} \gamma^t c\left(s_t, a_t, s_{t+1}\right)\right)$$

Expanding on this, the formulation of the value function changes from:

$$V_t(s; \theta) := \mathbb{E}\left[c_t^\theta + \mathbb{E}\left[c_{t+1}^\theta + \cdots + \mathbb{E}\left[c_T^\theta \mid S_{t+1}\right]\right] \mid S_t = s\right]$$

to

$$V_t(s; \theta) := \rho_t\left(c_t^\theta + \rho_{t+1}\left(c_{t+1}^\theta + \cdots + \rho_T\left(c_T^\theta\right)\right) \mid s_t = s\right)$$

Here, $V_t(s; \theta)$ denotes the value function incorporating the risk measure ρ applied sequentially to future costs $c_t^\theta, c_{t+1}^\theta, \dots, c_T^\theta$, conditioned on the state $s_t = s$. The value function changes from the cumulative cost starting at state s to the cumulative risk of cost starting at state s.

Static risk measures, typically evaluated at the end of each episode, lack time consistency. Thus, the dynamic setting, where risk measures are computed at each time step, is the primary focus of this project. Static measures can only assess immediate risk, leading to time-inconsistent solutions. With dynamic time-consistent risk measures, a recursive formulation akin to Bellman equations becomes feasible [11]. Standard exponential discounting (often denoted by γ) also exemplifies a time-consistent discounting function [24].

However, estimating the value function at each state (i.e., the dynamic risk measure) poses challenges. Previous approaches involved sampling transitions at each state, which is computationally

inefficient and impractical with market data due to the inability to sample new episodes [11]. Recent advancements leverage elicitable functions, which are minimized by scoring functions (analogous to loss functions) [10]. For instance, mean squared error serves as a scoring function for the mean, similar to how a neural network approximates the value function by minimizing the mean squared error between predicted and actual returns.

2.5.1. Elicitable functions

Elicitable functions are the first method used for estimating risk measures. This section provides the background theory for it.

Consider a random variable Y with support on $\mathbb{Y}$ and a mapping $M(Y)$, where $M : \mathbb{Y} \to \mathbb{A}$. The objective is to obtain an approximation $\mathfrak{a} \in \mathbb{A}$ of $M(Y)$. This approximation is evaluated using a scoring function $S : \mathbb{A} \times \mathbb{Y} \to \mathbb{R}$, akin to a loss function for $M(Y)$. When observing a realization $y \in \mathbb{Y}$, the current point forecast $\mathfrak{a} \in \mathbb{A}$ is penalized by $S(\mathfrak{a}, y)$. In essence, the scoring function S is designed so that forecasting the true value of $M(Y)$ (i.e. forecasting the realization correctly) minimizes the expected score. A mapping M is called elicitable if such a scoring function S exists. Intuitively, a function is called elicitable if there exists a loss function we can minimize, where the minimum loss is achieved only at predicting this function exactly.

To draw parallels to the risk-sensitive settings, expectiles serve as elicitable risk measures, with corresponding scoring functions minimizing the expectile of costs [25]. CVaR (Conditional Value at Risk), however, is conditionally elicitable on VaR (Value at Risk) [10]. This conditionality implies that while VaR is elicitable, CVaR requires separate approximation via two neural networks—one for VaR and another for CVaR given VaR.

2.5.2. Expectile Bellman operator

The expectile Bellman operator is the second method we use to estimate risk measures. We first begin with the Bellman operator, which can be described as applying one step of the (temporal difference) TD-error. However, while the TD-error is applied by looking at a single sample, the Bellman operator is an expectation over all states and actions.

$$(TV)(s) \coloneqq \max_{a}[V(s) + E_s[\delta(s,a)]]$$

Jiang, Liu, Ma, *et al.* [5] extends the Bellman operator by applying expectile mechanics to define the expectile Bellman operator:

$$T_\tau^\pi V(s) \coloneqq V(s) + 2\alpha \mathbb{E}_\alpha \mathbb{E}_{s'}[\tau[\delta]_+ + (1-\tau)[\delta]_-]$$

where α is the step size set to $\frac{1}{2\max\{\tau, 1-\tau\}}$, δ is the one-step TD error: $r(s,a,s') + \gamma V(s') - V(s)$, $[\cdot]_+ = \max(\cdot, 0)$ and $[\cdot]_- = \min(\cdot, 0)$. Note the similarities with the expectile definition in 2.4.

Jiang, Liu, Ma, *et al.* [5] show that the Expectile Bellman operator can be interpreted as an interpolation between the best-case and worst-case Bellman operations. They show that $\tau = 0$ and $\tau = 1$ reduce to the worst-case and best-case, respectively, and $\tau = 0.5$ reduces the operator to the standard Bellman operator.

τ is used as the parameter for risk preference, with $tau = 0$ for risk-averse and $tau = 1$ for risk-seeking learning.

2.6. Deep Learning

This section covers the deep learning concepts relevant to this project. This includes common ANN architectures such as CNN, 1-D CNN, flatten layer, dense layer, and softmax layer[26]:

- **1-Dimensional Convolutional Neural Network (1-D CNN)** A 1-D CNN is a variant of the CNN that operates on one-dimensional data, such as time series or sequential data. The network slides a filter (kernel) over the input data to capture features across the sequence. The 1-D convolution operation can be expressed as:

$$y_t = \sum_{i=1}^{m} x_{t+i-1} \cdot k_i$$

where x is the input vector, k is the kernel of length m.

- **Flatten Layer** The Flatten layer reshapes any multi-dimensional input (such as the output from a convolutional layer) into a one-dimensional vector. This transformation is commonly used to connect convolutional layers to dense layers.
- **Dense Layer** The Dense layer, also known as a fully connected layer, performs a linear transformation on the input data using a weight matrix and applying an activation function. The output of a Dense layer is computed as:

$$y = \sigma(Wx + b)$$

where W is the weight matrix, x is the input vector, b is the bias vector, and σ is the activation function applied element-wise to the linear transformation.

- **Softmax Layer** The Softmax layer is commonly used as the final neural network layer to normalize input values into a probability distribution. However, in our case, it will be used to normalize portfolio weights predicted by the policy network to ensure that the outputs sum to one, making them interpretable as the weights being allocated to the different assets. The softmax function for a vector input z is defined as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

3

Literature Review

This chapter presents an overview of the existing literature on reinforcement learning and its application to portfolio optimization, with a particular emphasis on risk-sensitive RL methods. The chapter begins by exploring portfolio optimization and the traditional methods commonly employed to solve this problem. It then transitions into a review of current research on using Deep RL for portfolio optimization. The focus then switches to risk-sensitive RL, discussing previous work that deals with risk measures in RL. We then highlight methods particularly relevant to this work, focusing on dynamic risk and the expectile Bellman operator. This is followed by a discussion on risk-sensitive distributional RL and robust RL, along with its connections to risk-sensitive methods. Finally, we present a table summarizing different risk measures considered for this project, their usefulness for portfolio optimization, and the implementation methods for applying them to RL.

3.1. Portfolio Optimization

The cornerstone of portfolio optimization lies in the Markowitz model, also known as the mean-variance model, which aims to maximize expected return while managing risk, typically measured by variance [27]. Initially formulated for single-period optimization, subsequent research has extended its application to multi-period scenarios [4].

In addition to mean-variance optimization, alternative methodologies include the Kelly Criterion, which prioritizes wealth growth by maximizing the expected logarithm of terminal wealth [28], [29], and Risk Parity, which adjusts asset allocations to equalize their contributions to portfolio risk [17].

Recent advancements in portfolio optimization introduce approaches such as mean-CVaR optimization, where Conditional Value at Risk (CVaR) replaces variance as the risk measure [20], and semi-variance optimization, which minimizes downside risk by focusing on semi-variance [30]. These methodologies, centered on various risk measures, align with ongoing efforts to integrate different risk frameworks into reinforcement learning formulations for portfolio optimization.

This project aims to contribute to the evolution of portfolio optimization by leveraging RL techniques that can adaptively incorporate diverse risk measures, thereby enhancing decision-making under uncertain financial environments.

3.2. Deep RL for Portfolio Optimization

Deep RL for portfolio optimization has received recent attention due to its potential to improve traditional approaches by being adaptive and data-driven. The recent survey conducted by Hambly, Xu, and Yang [31] on RL and its current applications in finance explores various Reinforcement Learning (RL) methods applied to portfolio optimization. It investigates several approaches, including value-based techniques utilizing Deep Q-Network (DQN) [13], [32], where the actions are buy-hold-sell signals instead of portfolio weights. Policy gradient algorithms such as Deterministic Policy Gradient (DPG), Deep Deterministic Policy Gradient (DDPG), and Proximal Policy Optimization (PPO) are also examined [14], [15], [33] [7], [34]–[36].

Some works that apply Deep RL to portfolio optimization and have similar goals to the baseline work we chose for this project[7] are Jiang, Xu, and Liang [8] and Sood, Papasotiriou, Vaiciulis, *et*

al. [9]. Jiang, Xu, and Liang [34] propose a framework for RL-based portfolio optimization employing one-dimensional convolutions and tensors to represent price histories as states, which aligns with our chosen baseline work. Similarly, Sood, Papasotiriou, Vaiciulis, *et al.* [9] conducts a comparative analysis between Deep Reinforcement Learning (DRL) and traditional Mean-Variance Optimization (MVO). This study employs a sliding-window style training and state space representations similar to Aboussalah, Xu, and Lee [7] but with notable differences, such as using sector returns instead of individual stock returns and adopting the Sharpe ratio as the reward function.

Unlike previous work, Aboussalah, Xu, and Lee [7], utilized as the baseline method in this study, employs a cross-sectional approach to Deep Reinforcement Learning (RL) for Portfolio Optimization. ”Cross-sectional” refers to the use of input features covering both related stocks at the same time point (cross-sectional) and historical time-series (time-series) of individual stocks. This means that the paper uses both the cross-sectional and the time-series approach for extracting features. Various policy gradient RL algorithms are evaluated in this work: Deterministic Policy Gradients(DPG)[37], Vanilla Policy Gradients[38], DDPG[15], and PPO[14]. These algorithms (and their variants) learn distinct ’strategies’: some risky, some conservative (by the degree of diversity).

Many other papers take different approaches from the above-mentioned methods to solve the portfolio optimization problem. In hierarchical portfolio management, Wang and Ku [39] introduces an approach where workers aim to maximize returns while maintaining risks below a predefined threshold, with managers intervening to minimize risk using metrics like Conditional Value at Risk (CVaR) [20]. MetaTrader [40] explores training multiple policies and using a meta-policy for policy selection. Zhang, Zhao, Wu, *et al.* [41] employs time-series and cross-sectional networks similar to Aboussalah, Xu, and Lee [7], utilizing RNNs for time-series data and convolutions for cross-sectional data. Furthermore, Jiang and Wang [42] integrates Twin Delayed Deep Deterministic Policy Gradient (TD3) with Long and Short-Term Risk control (LSTR), reallocating assets to risk-free options based on short-term risk levels while assessing long-term risk using historical probabilities over a specified period. Betancourt and Chen [43] applies Deep RL techniques in portfolio optimization scenarios with a dynamically changing number of assets.

3.3. Risk-sensitive Deep Reinforcement Learning

Extensive research has been conducted on risk measures and their application to RL. Previous studies have traditionally focused on minimizing risk over episodes or extending to dynamic risk measures, although outside the model-free RL framework [11]. One-dimensional measures over entire episodes, such as variance [44], CVaR [45], comonotonic measures [46], entropic measures [47], and spectral measures [48], have been explored, but are perceived as less engaging compared to dynamic counterparts. Additionally, various theoretical frameworks have been explored, including distribution-invariant risk measures, coherent risk measures, convex risk measures, and dynamic assessment indices[11]. Q-learning-based methodologies [49] and model-based dynamic risk measures [50] also contribute to the evolving landscape of risk-sensitive RL applications.

When it comes to this project, we investigate two distinct methodologies for integrating risk into reinforcement learning: dynamic time-consistent risk measures [22] and the expectile Bellman operator[5]. These approaches aim to enhance the robustness and stability of RL algorithms by addressing uncertainties and potential risks inherent in decision-making processes. The two approaches are further discussed below:

Dynamic risk measures, particularly those that are time-consistent, are well-suited for integration into the model-free RL framework due to its inherent time-consistency and recursive Bellman equations, as discussed in Ruszczyński [22]. The application of dynamic convex risk measures within RL, addressed by Coache and Jaimungal [11], presents computational challenges due to the necessity of sampling transitions at each visited state. Subsequent work by the same authors [10] alleviates these issues by confining convexity to spectral risk measures and utilizing conditional elicitability for CVaR (as introduced in section 2.5.1). Another approach by Marzban, Delage, and Li [6] explores elicitability properties in the context of expectile risk measures, primarily applied to option pricing and hedging problems. Unlike the on-policy and stochastic policy approaches commonly used by Coache, Jaimungal, and Cartea [10], this study employs an off-policy and deterministic policy algorithm. While not covered in this project, Pesenti [51] investigates the use of dynamic distortion measures.

The Expectile Bellman operator was first introduced in Jiang, Liu, Ma, *et al.* [5] and based on previ-

ous work in Ma, Yang, Hu, *et al.* [52]. The work focused on self-play(multi-agent RL where the agents learn by playing against themselves) but is also interested in risk-sensitive learning. It introduces the expectile Bellman operator, which is interesting for them as the risk parameter can be adjusted so that the agent can go from risk-averse to risk-seeking. It is not based on dynamic risk but should be time-consistent as it extends the Bellman update to account for risk sensitivity.

This project focuses on methods that leverage risk measures within reinforcement learning, emphasizing their integration into the model-free RL paradigm. Future research directions aim to refine these approaches by addressing computational efficiencies and expanding the applicability of dynamic risk measures in RL.

Risk-sensitive Distributional RL As mentioned previously, another approach for risk-sensitive RL is through distributional RL, which operates on the distribution of returns rather than a scalar expected return [53]. In this framework, the risk measure is a function that maps the distribution of the value function to a real number. Extending from Ma, Xia, Zhou, *et al.* [53], subsequent work further applies and expands this methodology to the portfolio optimization problem[12]. This approach entails formulating risk assessments for Value at Risk (VaR), Mean-variance, and distorted expectations [54] (which include Cumulative Probability Weighting (CPW) [55], Wang's risk measure [56], and Conditional Value at Risk (CVaR) [20]). Ma, Xia, Zhou, *et al.* [53] use the returns distribution to approximate the value function under various risk measures. This methodology finds application in diverse RL environments such as MuJoCo and Box2d within the OpenAI Gym [57], [58]. Out of these risk measures, CPW is observed to exhibit sensitivity issues at extreme wealth levels, thereby rendering it less suitable for portfolio optimization, as noted in recent research that extends this approach to portfolio management [12].

Robust RL Robust RL is a sub-topic of RL that, though not directly related to risk-sensitive RL, is still tangentially related and warrants future research. There exists a link between risk-sensitive reinforcement learning (RL) and robust RL paradigms, which can be interesting to explore in the future. Robustness is defined as the optimization against the worst-case scenario within a specified uncertainty set [11]. The work in Moos, Hansel, Abdulsamad, *et al.* [59] establishes that various risk measures can be viewed equivalently as methods in robust RL under specific uncertainty sets. Several studies delve into robustness with varying degrees of emphasis. For instance, Jaimungal, Pesenti, Wang, *et al.* [60] focuses on a robust version of the Rank Dependent Expected Utility (RDEU) risk measure, which is used in portfolio allocation. Notably, this approach differs from others by adopting an optimal pre-commitment strategy rather than adhering strictly to dynamically time-consistent risk measures. Alternatively, Queeney and Benosman [61] addresses model uncertainty within RL contexts, incorporating risk measures such as Conditional Value at Risk (CVaR) and Wang's risk measure. This study is significant as it aligns risk measures with robustness guarantees, emphasizing their equivalence under uncertainty. Moreover, Kanazawa, Wang, and Gupta [62] introduces a framework involving multiple actors and critics within RL, where risk is incorporated into the actor's loss function, specifically utilizing CVaR. Epistemic uncertainty, represented by the variance in the critic Q-value networks, is also considered.

Choosing risk measures Various risk measures and their suitability were also assessed. These are summarized in 3.1. We choose to work with CVaR and Expectile and follow the dynamic risk and Bellman operator methods, as discussed previously. We decided not to work with distributional RL in this project because of time constraints and because it has already been applied to portfolio optimization by another project[12].

Table 3.1: Summary of risk measures

Measure	Formula	Implementation	Usefulness
VaR	α-quantile	Distributional[12], [53]	Potential loss for given α, no indication of how bad loss above threshold
CVaR	$\frac{1}{1-\alpha}\int_{\alpha}^{1} VaR_u(Y)du$	Distributional[12], [53] and Dynamic[10]	Expected value of loss above threshold
Wang	$g(\tau) = \Phi(\Phi^{(}-1)(\tau)+\beta)$	Distributional[12], [53]	$g'(\tau)$ similar to CVaR, but smoothly decreasing
Expectile	$\arg\min_{\tau}(1-\tau)\mathbb{E}[(\tau-X)_{+}^{2}]+\tau\mathbb{E}[(\tau-X)_{-}^{2}]$	Dynamic[6] and Expectile Bellman[5]	asymmetric generalization of mean
Semi-variance	$\mathbb{E}[Z]-\beta\sqrt{\mathbb{V}[Z]_{-}}$	Perturbation Analysis[63]	Not useful, RL formulation solves single-period problem
CPW	$g(\tau)=\frac{\tau^{\beta}}{(\tau^{\beta}+(1-\tau)^{\beta})^{\frac{1}{\beta}}}$	Distributional[53]	Not useful, both sides sensitive
mean-variance	$\mathbb{E}[Z]-\beta\sqrt{\mathbb{V}[Z]}$	Distributional[12], [53]	Not useful, both sides sensitive

4

Methods for Risk-sensitive RL

This chapter describes the specific implementation details of the different algorithms (and their variants) used in this project. This mainly includes the risk-sensitive RL methods (PPO-CVaR, PPO-DDPG, and PPO-Expectile) covered in section 4.2. These build on top of the risk-neutral cross-sectional approach described in section 4.1. The explanations here include both existing risk-sensitive RL methods and any changes made to adapt them for portfolio optimization, as well as changes made to the algorithm for methods already applied to portfolio optimization. Finally, section 4.3 describes a method for introducing exploration noise when the agent is learning the policy. However, this work is orthogonal to the project's main focus, risk-sensitive RL, and the experiments and results can be found in the appendix (A.3).

It should be noted that the next chapter (5) will cover the experimental design and setup using the algorithms described here. This includes details such as the data used or the training setup followed. Thus, they are independent of whether they work with market data or a simulator. Apart from this, when it comes to evaluating the agents, while the goal of these agents might be to maximize cumulative reward (or minimize the cumulative risk of cost), that is not the metric by which their performance will be evaluated. The actual evaluation metrics will be financial metrics, described as a part of the experiment design in 5.1.

We first begin with section 4.1, which describes the baseline risk-neutral approach adapted for this project. This work introduces a novel cross-sectional and time-series feature extractor and a novel reward function, both based on financial concepts. The risk-sensitive methods described ahead will build on top of this method. The risk-neutral baseline is followed by section 4.2 with different risk variants that build on top of the baseline: PPO-CVaR, DDPG-Expectile, PPO-Expectile. It should be noted that DDPG-Expectile and PPO-Expectile were used for other applications (option pricing and heading and self-play, respectively) and are now applied to a portfolio optimization setting. PPO-CVaR, on the other hand, has already been used with portfolio optimization but has now been modified from being based on a generic actor-critic algorithm to being applied to PPO. We use the same CS feature extractor with 1D CNNs from the baseline work, while the reward function (alongside other details) will differ for the different methods. The use of the feature extractor in the project is also novel, as the risk-sensitive RL methods use standard ANN layers in their implementation.

4.1. Cross-sectional approach

Aboussalah, Xu, and Lee [7] introduce a data-driven Deep RL approach for portfolio optimization. The paper implements four actor-critic RL algorithms in the portfolio optimization setup as discussed in section 3.2. This work focuses on the DDPG and PPO RL algorithms. As discussed before, there is no objectively best reward function or ground truth for the state representation. Thus, the main contributions of this paper, the reward function, and the state representation are part of the methodology.

Derived from finance concepts, the feature extractor is based on the concept of time-series and cross-sectional asset pricing[64]. The TS feature extractor looks at the historical performance of a single stock, while the CS extractor looks at the relative performance of stocks at the current time step. The idea is that the agent learns to decide the portfolio weights based on this information. The reward function has the log return alongside three other terms: alpha, beta, and gamma, which will be

discussed further below.

4.1.1. State Representation

The state representation starts with the tensor of input data, which is fed into the feature extractor. The feature extractor outputs a feature vector acting like a latent state representation. This feature vector is, in turn, fed into the policy and value (or Q-value) neural networks used by the RL algorithms.

Input Tensor The state consists of five normalized pricing data points: high, low, open, close prices, and volume, over a historical window for multiple stocks. Additionally, the state includes the previous action, i.e., the portfolio weights. Including previous weights is crucial to mitigate high transaction costs and to ensure that the reward function accounts for the actual change in the portfolio, avoiding stochastic reward variations for the same state under different previous actions.

Feature Extractor Architecture The state tensor is fed into a feature extractor that produces a feature vector, subsequently input to the policy and value networks. The relevant deep learning components have been defined in 2.6. The feature extractor's architecture comprises:

- Time-series (TS): Two blocks of 1D CNNs, analyzing the time window of each stock separately.
- Cross-sectional (CS): Two blocks of 1D CNNs, analyzing all stocks together at the current timestep.
- Multiple combinations of the feature extractor are possible. These variants are as follows:
  - CNN-TS → Flatten → Dense → Concat action.
  - CNN-CS → Flatten → Dense → Concat action.
  - CNN-TS and CNN-CS → Flatten & Concat → Dense → Concat action.

4.1.2. Action Representation

The action space represents the portfolio weights assigned to each stock. These are continuous actions ranging from (0, 1). The environment performs a softmax[26] on the actions predicted by the policy network. This additional constraint forces the actions to sum up to 1.

4.1.3. Reward Function

The reward function is designed as follows:

$$\log\left(\frac{\text{price}_t}{\text{price}_{t-1}}\right) - \alpha \cdot \text{equal weights return} - \beta \cdot \text{covariance} - \gamma \cdot \text{max weights}$$

The log return (instead of the return) takes advantage of the additive property. Summing the log returns yields the total log return over a given period. This simplifies the optimization, converting the product over periods into a sum. This also fits nicely into the recursive Bellman equations, which compute the (RL) return as the cumulative sum of rewards.

The alpha term uses the log return of equal weights to introduce some idea of an agent trying to outperform a baseline. The beta term uses the covariance between the returns of the stocks to force the agent to reduce volatility. The gamma term limits the maximum weight allocated to any stock to encourage diversity.

4.2. Risk-sensitive Deep RL methods for Portfolio Optimization

This section describes the different risk-sensitive RL methods, which will build on the cross-sectional approach described above (4.1). The state representation, feature extractor, and actions remain the same in all the following methods. Each method introduces a risk measure in the optimization function so that the agent's goal no longer remains to maximize the reward function described in 4.1. These specific methods need to be used as optimizing for risk is non-trivial. We needed to account for how the Bellman operator breaks when minimizing a risk measure. PPO-CVaR and DDPG-Expectile present their algorithms in a cost-minimization framework (instead of the usual reward maximization)[6], [10], and thus, the reward function changes to the one-step loss (negative of one-step portfolio return). PPO-Expectile changes the Bellman operator, but there are no changes to the reward function.

4.2.1. PPO CVaR

Coache, Jaimungal, and Cartea [10] uses the concept of conditional elicitable CVaR (as explained in 2.5.1) to devise a risk-sensitive actor-critic algorithm. They also apply this to the portfolio optimization setting (among others) as an example. In this project, the generic actor-critic algorithm (referred to as CE-AC from now on) is extended to the PPO algorithm. We first show the methodology of Coache, Jaimungal, and Cartea [10], followed by the changes made to CE-AC to obtain what we call PPO-CVaR.

As discussed in 2.5, the objective of the RL agent changes from minimizing cost to minimizing the CVaR of cost. Similarly, the value function is an estimate of the CVaR of cost for a given state and time:

$$V_t(s;\theta) = CVaR_\alpha\Big(c_t^\theta + V_{t+1}(s_{t+1}^\theta;\theta) \mid s_t = s\Big)$$

However, CVaR cannot be estimated directly. It is possible to obtain multiple samples for each state and calculate an estimate from that[11]. Still, this method is computationally inefficient and not possible with market data where only a single episode is available. Thus, the conditionally elicitable property of CVaR is exploited. This means that a scoring function (similar to a loss function) exists for CVaR, which allows it to be estimated by employing VaR as an intermediary step. Two separate neural networks are utilized to estimate the VaR and CVaR. $H_{1,t}(s;\theta)$ functions as the estimator for VaR, while $H_{2,t}(s;\theta)$ serves for the intermediate value, ultimately contributing to the overall value (CVaR) estimation as $H_{1,t}(s;\theta)+H_{2,t}(s;\theta)$. This partitioning approach ensures computational stability, enforcing the condition CVaR > VaR. The scoring function employed is as follows:

$$S(\mathfrak{a}_1,\mathfrak{a}_2,y) = \log(\frac{\mathfrak{a}_2 + C}{y + C}) - \frac{\mathfrak{a}_2}{\mathfrak{a}_2 + C} + \frac{\mathfrak{a}_1(\mathbb{1}_{y\le\mathfrak{a}_1} - \alpha) + y\mathbb{1}_{y>\mathfrak{a}_1}}{(\mathfrak{a}_2 + C)(1-\alpha)} \qquad \text{(L1 loss)}$$

Here, $\mathfrak{a}_1$ represents VaR, $\mathfrak{a}_2$ denotes CVaR, and y denotes the target value. While theoretically, this means an ensemble of ANNs for each time step, practically, a single ANN can be used with time treated as a dimension within the state space. The explanation of the scoring function and how this specific formulation was chosen can be found in Coache, Jaimungal, and Cartea [10].

For the policy gradient, the following loss function is used:

$$\mathcal{L}^\theta = \frac{1}{1-\alpha}\left[\Big(c_t + V_{t+1}^\psi(s_{t+1};\theta) - H_{1,t}^{\psi 1}(s_t;\theta)\Big)_+\Big(\nabla_\theta log\pi^\theta(a|s_t)\Big)\right] \qquad \text{(L2 loss)}$$

Explained next are the changes made to incorporate the above-described algorithm of a generic actor-critic method for PPO. In PPO, the overall loss function is divided into the (clipped) policy gradient loss, the value estimation loss, and the entropy loss. No changes are made to the entropy loss, as that is simply the negative entropy of the policy. The value loss is adjusted according to the L1 loss described above, and the policy-gradient loss is adjusted according to the L2 loss:

1. Value loss
   (a) L1 loss in CE-AC has three terms: $\mathfrak{a}_1$, $\mathfrak{a}_2$, and y. a1 being VaR, a2 being CVaR (or value estimate), and y being the target. CE-AC calculates the target as the one-step target. Conversely, PPO calculates the MSE loss between the value predicted by the value network and the TD-lambda target from Generalised Advantage Estimation (GAE)[65]. Thus, for PPO-CVaR, taking L1 loss from CE-AC and the TD-lambda target from PPO, L1 loss with the TD-lambda target is used.
   (b) The value loss in PPO is calculated as the MSE loss between the value predicted by the value network and the TD-lambda target calculated using GAE. In CE-AC, the MSE loss between the predicted value and TD-lambda target is replaced with L1 loss. Here, are $H_{1,t}(s;\theta)$, value and target respectively.
2. Policy gradient loss
   (a) The standard policy gradient can be stated as: $\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)A_t\right]$ and the modified policy gradient in Coache, Jaimungal, and Cartea [10] that accounts for the change in advantage is:

$$\mathbb{E}\left[\frac{1}{1-\alpha}\Big(\nabla_\theta \log \pi_\theta(a \mid s_t)\Big)\big(\text{target} - H_{1,t}(s;\theta)\big)_+\right] \qquad (4.1)$$

(b) In trust region algorithms like PPO, the standard policy gradient is modified to work with policy ratios ($r_t(\theta)$): $\mathbb{E}\left[\nabla_\theta r_t(\theta) A_t\right]$.

(c) Thus, the goal is to replace the advantage according to the risk-sensitive policy gradient loss. The policy loss in equation 4.1 has the value part of the policy gradient as (target$-H_{1,t}(s;\theta)$), so in PPO-CVaR, the advantage (which in standard PPO is estimated by target $- V_t(s;\theta)$) is replaced by (target $- H_{1,t}(s;\theta)$).

4.2.2. DDPG-Expectile

The expectile, being coherent and elicitable, is considered a suitable risk measure, especially compared to the conditionally elicitable Conditional Value-at-Risk (CVaR). Compared to standard DDPG, DDPG-Expectile replaces the Mean Squared Error (MSE) loss between the target and predicted Q-values with the following expectile loss function:

$$l(y) := \left((1-\tau)\mathbf{1}_{\{y>0\}} + \tau \mathbf{1}_{\{y\leq 0\}}\right) y^2$$

where $y = Q_t - (c_t + Q_{t+1})$, with Q_t and Q_{t+1} denoting the current and next q-values respectively, and c_t representing the cost. Here, τ parameterizes the loss function, offering a generalized form of the MSE-loss.

To apply this framework to a Policy Optimization (PO) problem, the objective changes from maximizing the reward to minimizing the risk of the cost. Thus, the goal is to minimize the Q-function, and the reward function is substituted with the cost (negative reward), expressed as $-\frac{P_t - P_{t-1}}{P_{t-1}}$. Notably, the current time is incorporated into the state representation.

To address the practical issues encountered with extreme actions in Deep Deterministic Policy Gradient (DDPG), an alternative approach in the form of PPO-Expectile was adopted to work with the expectile measure. The results from the DDPG-Expectile experiments can be found in the Appendix (A.2).

4.2.3. PPO-Expectile

Instead of using dynamic risk or elicitable functions, PPO-Expectile works with the concept of Expectile Bellman Operator, introduced in section 2.5.2. This method was originally to train a risk-sensitive algorithm for self-play[5]. The main advantages of this algorithm are that it requires minimal changes to PPO, works independently of the data, and can be tuned to be both risk-averse and risk-sensitive.

The main change is made to the Bellman operator, generalizing it based on expectile to account for worst-case to best-case Bellman operator, depending on risk parameter tau:

$$T_\tau^\pi V(s) := V(s) + 2\alpha \mathbb{E}_{a,s'}\left[\tau[\delta]_+ + (1-\tau)[\delta]_-\right]$$

where δ is the 1-step TD error $\delta = r + \gamma V(s') - V(s)$

Given that PPO uses Generalized Advantage Estimation (GAE), the extension of GAE to incorporate expectile is an important step. However, this extension presented challenges, as the Bellman expectile operator is nonlinear. Typically, GAE is a weighted sum of various n-step Bellman operations, but due to the non-linearity introduced by the operations $[\cdot]_+$ and $[\cdot]_-$, there is no general formula for an n-step expectile Bellman operator. The formula is only feasible for specific cases, such as the n-step and (n-1)-step operations. A table must be maintained to handle these calculations. Empirical analysis indicated that this requirement introduces a complexity increase in the order of 1.x, which was acceptable.

To apply these concepts to the policy optimization (PO) problem, the setup was aligned with the baseline CS approach [7], maintaining the same feature extractor, state representations, actions, and reward structures. For simplicity, however, the reward was defined solely in terms of log return.

4.3. Exploration noise

In Proximal Policy Optimization (PPO), a common challenge is ensuring adequate exploration, particularly when different actions lead to similar or identical states. This issue is illustrated in scenarios where, regardless of the chosen portfolio weights, the portfolio returns at the next time step remain unaffected. The agent's action, $\mathbf{a} = (a_1, \ldots a_N)$, is an N-dimensional vector, with each action component corresponding to the proportion of an asset in the portfolio. Standard exploration in PPO uses univariate Gaussians (or multivariate Gaussian with diagonal covariance), where each component of $\mathbf{a}$ is treated

independently. Consequently, there may be dependencies between the action components that would not be learned, which is inappropriate for portfolio weights that are inherently interdependent[66].

A straightforward solution to this problem is to change the distribution from a multivariate Gaussian with diagonal covariance to one with full covariance. This adjustment allows for modeling correlations between action components, which is more suitable for portfolio management tasks. In contrast, other applications, such as robotics, encounter similar issues but do not appear to suffer significantly from using univariate Gaussians.

Lattice (LATent TIme-Correlated Exploration) proposes a more general approach [67]. It incorporates state and policy-dependent perturbations of the actions while modeling the policy as a multivariate Gaussian. This method aims to address the limitations of univariate Gaussian policies by better capturing the underlying correlations between action components, leading to improved exploration and policy performance in tasks where action dependencies are significant.

The topic of exploration with dependencies between action components appears to be an interesting problem for RL algorithms in general and for the portfolio optimization problem. However, since this was also tangential to risk-sensitive RL methods, this was not explored as extensively, and we have only conducted preliminary studies.

5

Experiments and Results

This chapter describes the experiments and their corresponding results for this work. Section 5.1 first describes the different experiments and how they link back to the research questions defined in section 1.2. It then covers the different evaluation metrics in section 5.1.1, which are used to quantify the performance/behavior of the agents in the experiments. Section 5.1.2 describes the two training setups: using market data and using a simulator. This covers the data characteristics such as the financial assets, the time period, and the train/test splits.

Three sections cover the experiments and their results. First is 5.2, which covers the reproduction of the baseline risk-neutral RL approach and results relating to the reward function design. Next is 5.3, which covers the risk-variant experiments of PPO-CVaR and PPO-Expectile in both the simulator and the market data setups. Finally, 5.4 covers the single sample simulation experiments with PPO-CVaR. This is done to evaluate the learning from a single sample of simulation data in a market data setup.

It should be noted that due to the issue of extreme actions faced with DDPG, the focus was shifted to PPO-Expectile. The experiments related to DDPG-Expectile are included in the appendix A.2.

5.1. Experiment Design

1. **Baseline reproduction** Since our methods are based on extending a risk-neutral cross-sectional approach [7] to risk-sensitive RL, we first replicate the results of this paper as a verification step. The results of the baseline using PPO are reproduced in the market setup. The RL method used here is the risk-neutral cross-sectional approach described in 4.1, and the training setup followed is described under 'Market Data' in 5.1.2.

2. **Risk-sensitive RL with simulations (RQ1 and RQ2)** The research questions RQ1 and RQ2 (section 1.2) were about measuring the impact of using different risk measures and their risk sensitivity parameters on portfolio strategies. To address these questions in a controlled environment, we first design an experiment where the PPO-CVaR and PPO-Expectile methods (section 4.2) are evaluated in a *market simulator*, as described in section 5.1.2. Simulations allow us to isolate the effects of risk measures and their parameters without worrying about whether we have enough data or not.

3. **Risk-sensitive RL with market data (RQ1, RQ2, and RQ3)** We would like to address RQ1 and RQ2 in a more realistic setting where we do not have the ability to reset the market conditions and generate as many samples as we need. In addition, the research question RQ3 inquires about how the performance is affected by having risk-sensitive RL methods trained under real market data instead of a simulator. We train and evaluate PPO-CVaR and PPO-Expectile to address all three questions using real market data. This experiment is identical to the previous one except for the training setup, which is now described under the *market data* in section 5.1.2.

4. **Risk-sensitive RL with a single sample from simulations (RQ3)** Comparing the results of experiments 2 and 3 provides a partial answer to RQ3. Since the simulated environments have the advantage of generating as many episodes as we like, we also compare simulated studies

with only one episode to the real market case. We use PPO-CVaR with the market data setup from experiment 3, but the market data is replaced by a single sample from the simulator instead of the real market data. The goal here is to see whether the advantage of a simulator comes from the ability to generate many samples or from the fact that the simulators used are much simpler than the real market dynamics. This experiment is detailed in section 5.4.

5.1.1. Evaluation criteria

The following metrics will be used to evaluate the experiments:

- **Sharpe Ratio:** The risk-adjusted return of an investment portfolio. It is calculated as the ratio of the portfolio's excess return (over the risk-free rate) to the standard deviation of the portfolio's excess return.

$$SR = \frac{R_p - R_f}{\sigma_p}$$

where R_p is the portfolio return, R_f is the risk-free rate, and σ_p the standard deviation of the portfolio return.

- **Portfolio Return:** A portfolio's overall return over a period.

$$PR = \frac{P_T - P_1}{P_1}$$

where P_T is the final portfolio value and P_1 the initial portfolio value.

- **Maximum Drawdown:** The maximum loss from a peak to a trough experienced by the portfolio during a specified period.

$$MDD = \frac{\text{Peak Value} - \text{Trough Value}}{\text{Peak Value}} \times 100\%$$

where the Peak Value is the highest portfolio value reached before a drawdown, and the Trough Value is the lowest portfolio value reached during the drawdown period

- **Terminal CVaR**: CVaR of the average return at the end of the episode, estimated over the evaluation episodes.

- The plot of actions learned by the agent will be qualitatively evaluated for any change in strategies across methods.

5.1.2. Training setup

The two training procedures for the experiments that work with either the market data or the simulated episodes are presented here:

Market Data This setup is used for all market data experiments. Following Aboussalah, Xu, and Lee [7], the setup employs daily market data sourced from Yahoo Finance for a selection of 10 stocks, namely American Tower Corp. (AMT), American Express Company (AXP), Boeing Company (BA), Chevron Corporation (CVX), Johnson & Johnson (JNJ), Coca-Cola Co (KO), McDonald's Corp. (MCD), Microsoft Corporation (MSFT), AT&T Inc. (T), and Walmart Inc (WMT). These stocks are chosen from diverse sectors of the S&P 500 index to ensure diversity. The input tensor has dimensions historical_window * number_features * number_stocks. This becomes 20 * 5 * 10. The input features are open-high-low-close prices and the volume. The combined train and test time-period covers approximately 2012 to 2015.

The experiments follow a sliding window approach based on Aboussalah, Xu, and Lee [7], which is also commonly used in financial applications as described in West [68] and depicted in Figure 5.1. There are 50 rounds, each with a 200-day training and 10-day testing phases. A 10-day sliding window is added between rounds to adapt the model to evolving market conditions. In each round, the dataset is shifted by ten days, discarding the outdated training data and including test data from the previous

round. This approach efficiently uses the limited data while ensuring the model is continuously tested out-of-sample during each testing phase. The trained model from an earlier round is reused and, in a way, fine-tuned each round to leverage the most recent market information. Each round of training lasts 10k steps. The test results are consolidated into a single continuous time series for all experiments to evaluate the model's performance over the entire test period. This training procedure is more realistic for financial applications and simulates how deep RL models would be deployed and used by portfolio managers in real time.

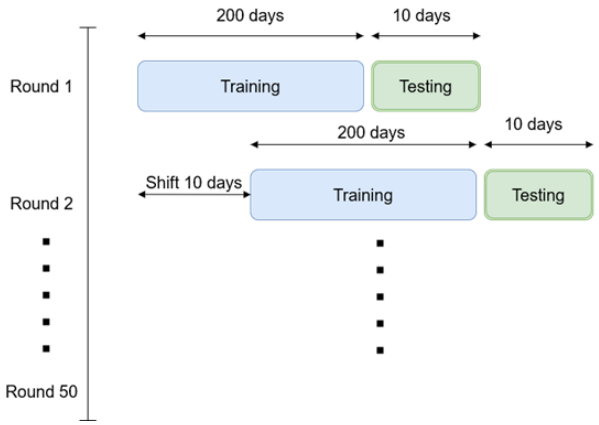


Figure 5.1: Sliding window training and testing

Market Simulator The training setup for the market simulator is taken from Coache, Jaimungal, and Cartea [10]. It is a model with three assets where the price behavior of the assets is simulated through a Geometric Brownian motion, defined as:

$$dS_t^{(i)} = \mu^{(i)} S_t^{(i)}\, dt + \sigma^{(i)} S_t^{(i)}\, dW^{(i)}$$

The simulation parameters are as follows:

- The drift coefficients (μ) are given by $\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix}$.
- The volatility coefficients(σ) are given by $\begin{bmatrix} 0.06 & 0.12 & 0.18 \end{bmatrix}$.
- The correlation matrix is given by:

$$\begin{bmatrix} 1.0 & 0.2 & -0.2 \\ 0.2 & 1.0 & -0.2 \\ -0.2 & -0.2 & 1.0 \end{bmatrix}.$$

The prices in this simulation have the same drift-to-sigma ratio (implying the same Sharpe Ratios) but with increasing returns (μ) and volatility (σ). Figure 5.2 shows the average simulation and $1-\sigma$ deviation over 100 episodes. Like the market setup, the training episode length is 200. However, the simulated price is now the only feature; the number of assets is 3, and the historical window is 36. The input tensor of historical_window * number_features * number_stocks is thus 36 * 1 * 3.

There is no sliding window; instead, we sample episodes to train and evaluate the agent, just like in standard RL training and evaluation. The training uses 1000 simulated episodes, and the evaluation is done over 800 simulated episodes. The training converges on the cumulative reward at around 200k steps. Thus, 1000 episodes (at 200 steps per episode) were chosen.

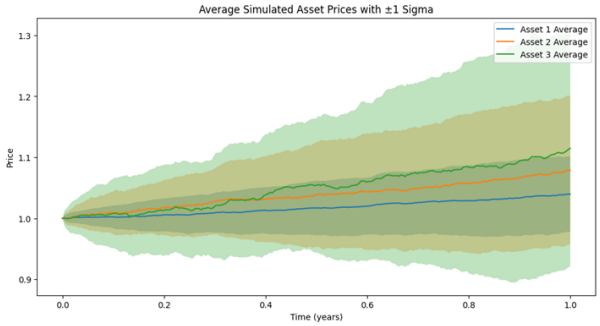


Figure 5.2: Average over 100 simulations

Continuing from the experiment design in 5.1, the presented experiment design is slightly adjusted for clarity when presenting the results. The first set of experiments concerning the baseline reproduction is under one section (5.2). However, experiments 2 and 3 aim to see how market data vs simulation affects the algorithms (PPO-CVaR and PPO-Expectile). So, we look at them per algorithm, and the second and third sets of experiments are combined under one section. The simulation and market data experiments for PPO-CVaR and PPO-Expectile are presented together in section 5.3. The final experiments following a single sample of simulations are presented in section 5.4.

5.2. Baseline Cross-sectional approach

5.2.1. PPO Reproduction

Figure 5.5 shows the portfolio return over the test period, with the reproduction to the left and the original plot from Aboussalah, Xu, and Lee [7] to the right. The reproduction plots the average and 1σ deviation over five seeds of portfolio value over five runs, and the original plot contains the portfolio value for multiple variants. It should be noted that the original paper's plot ends around 2014-10, while the reproduction continues till 2015-01. As we can see, these results are very close to the original paper but do not completely overlap, as the hyper-parameters or seeds were unavailable. The results of the PPO run, alongside the non-RL baselines of MVO and EQB, are shown in table 5.1. The performance of all three is very close, but the PPO does not outperform in terms of the Sharpe Ratio or portfolio return.

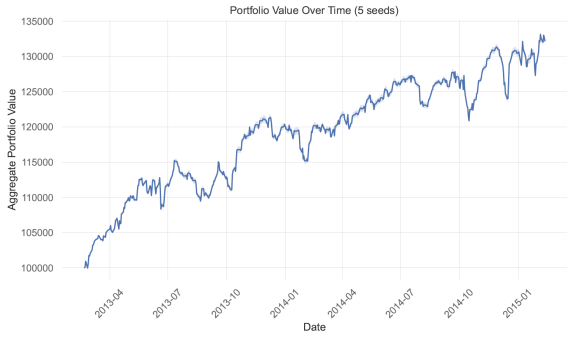


Figure 5.3: Average portfolio values (5 seeds)

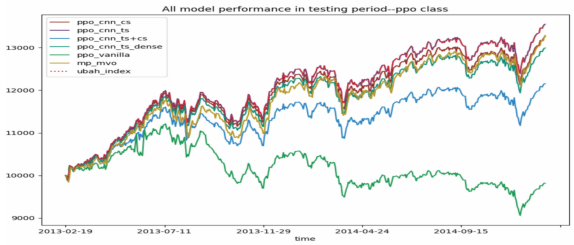


Figure 5.4: PPO portfolio performance, multiple-variants

Figure 5.5: PPO baseline portfolio performance: Reproduction (left) and Original (right)

Figure 5.6 shows the actions area plot. The shaded area corresponding to an asset indicates the relative amount allocated at that time step. This is close to equal weights (similar to paper). However, it is unclear what the strategy was or how it was learned. The following section describes the investigation of the reward function, which sheds some light on how the agent was learning.

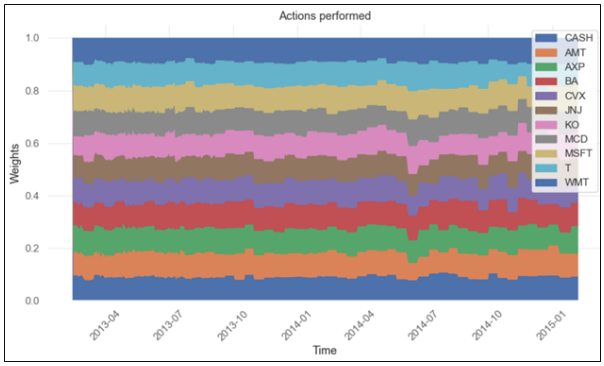


Figure 5.6: PPO Reproduction: Actions area plot

ID	Final Accumulative Portfolio	Maximum DrawDown	Sharpe Ratio
CS (average 5 runs)	1.32	-0.055	1.57
EQB	1.35	-0.056	1.72
MVO	1.36	-0.06	1.73

Table 5.1: Portfolio Performance Metrics

5.2.2. Reward function design

The alpha, beta, and gamma terms introduced by the baseline paper were found to impact the strategies learned by the agent significantly. This was mainly seen by the reward function design described below:

The experiment shown here worked with gamma=0. It should be noted that experiments with beta=0 and gamma=0 have very similar performance. Figure 5.7 shows one of the results from these experiments where the agent learns to focus on one stock. Figure 5.8 shows the portfolio return, outperforming the PPO reproduction from the previous section. However, looking at the action plot, it is clear that this happens because the agents learn to focus on one stock with higher returns, which is a very risky strategy. It is only by removing the gamma term from the reward function that we see this behavior. When the gamma is in the reward function, the agent learns to stay close to equal weights. These results indicate that it is the gamma term that forces the agent to stay close to equal weights, as having removed the term, the agent learns to focus on one stock with high returns.

Figure 5.7: Reward function design: Actions

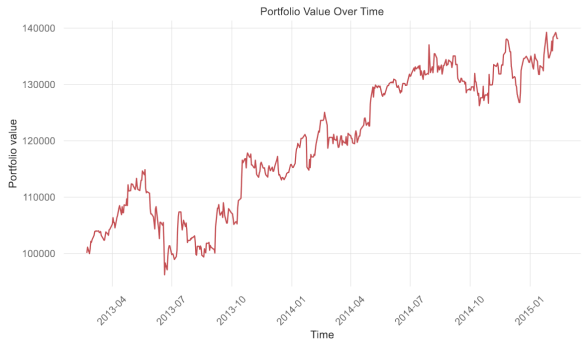


Figure 5.8: Reward function design: Portfolio return

The reward function highly depends on hyper-parameter tuning, significantly influencing the agent's performance. Specifically, the agent's performance is primarily driven by the gamma term within the reward function, which appears to bias the agent towards maintaining near-equal portfolio weights. While beneficial in reducing risk, this bias potentially hinders the agent's ability to explore a broader range of strategies.

The design of the reward function introduces an inductive bias towards policies that favor equal-weighted portfolios (EQB). While this bias aligns well with the current market conditions, it does not necessarily imply that the agent is learning an optimal or adaptable policy. Instead, it suggests that the reward function effectively guides the agent to adopt EQB as a favorable strategy, which may be close to the best achievable outcome given the current data. However, it should also be noted that both the MVO approach and the PPO algorithm appear to converge towards a similar policy, indicating that an equal-weighted strategy is indeed robust for the data provided.

We also tried some other changes to the reward function, but they all either learned close to all-in strategies or strategies that focused on a few stocks but performed much worse than equal weights. These experiments are detailed in the appendix (A.1).

5.3. Risk-sensitive RL with simulated data and market data

5.3.1. PPO-CVaR

Simulator Figure 5.9 plots the Profit&Loss and the terminal portfolio value over the average of 800 evaluations. This is calculated for the risk-neutral PPO (base) and three runs of risk-sensitive PPO-CVaR (0.05, 0.1 and 0.9). The outer band represents the worst/best performance, while the dotted plot is the 90% quantile.

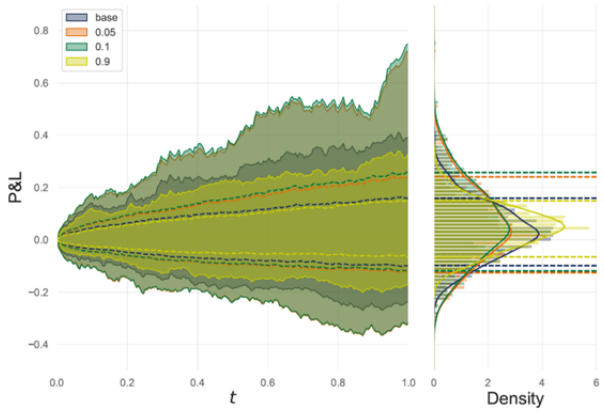


Figure 5.9: PPO-CVaR: Simulator evaluation

Figure 5.10 illustrates the average actions predicted by the agent over 800 evaluation episodes. The actions associated with alpha values of 0.05 and 0.1 exhibit similar patterns, with a heavy allocation towards asset 3. However, for an alpha value of 0.9, the agent's actions shift towards an equal-weighted

portfolio. As alpha increases, the allocation tends to favor lower volatility assets (assets 1 and 2), while the allocation to the higher volatility asset three decreases. This suggests that higher alpha values drive the agent towards a more balanced portfolio distribution, reducing the exposure to high-risk assets.

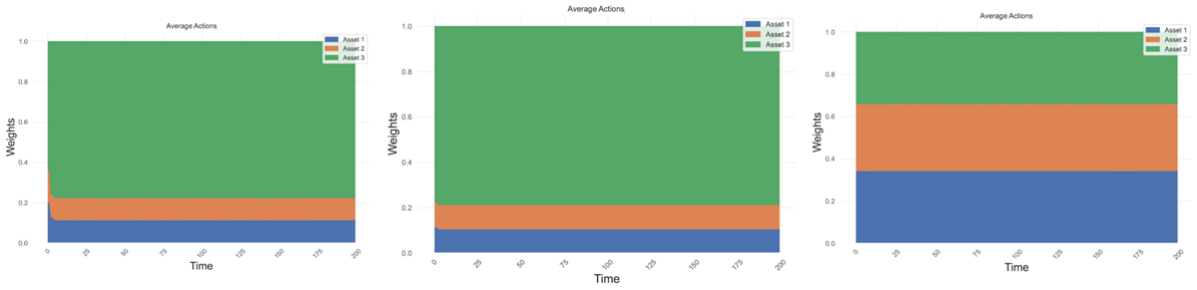


Figure 5.10: Average actions: PPO-CVaR

Figure 5.11 shows the terminal CVaR values at different alphas for the four runs (base, alpha(0.05, 0.1, 0.9)). Here, CVaR at alpha level 5% means the CVaR of the return of worst 5% runs. We see that alpha=0.9 (most risk-sensitive) always has the best CVaR value at the different alpha levels. This makes sense as the CVaR(alpha=0.9) is supposed to be (and learns) the most risk-sensitive strategy.

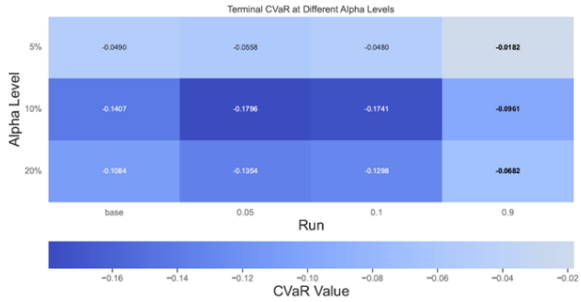


Figure 5.11: Terminal CVaR: PPO-CVaR

We obtain reasonable results with the simulator setup. Some additional work was required: normalizing the rewards and observations and a hyper-parameter sweep. The following parameters are found to be the most important: ent_coef, gae_lambda, and lr. However, it was noted that the explained variance stays around 0. A possible explanation is that the agent is learning action regardless of state, and the simulator is simple enough for this to happen. This can be addressed in future work by working with more scenarios (by varying the model coefficients) or by working with more complex price simulators.

Market Data We now switch to a market data setup with ten assets and a sliding window setup, as described in 5.1. The portfolio performance under the 2-year test period is presented in table 5.2. The risk-sensitive variants have similar values but perform worse on average than the baseline PPO. The action area plots for the test period can be seen in 5.15. There are also no clear patterns with the changing α parameter. We notice some patterns, such as $\alpha = 0.9$ picking 'T,' which is a low volatility asset. However, from these experiments, it is unclear why this was done. Also, other assets apart from 'T' with volatility were not chosen.

ID	Final Accumulative Portfolio	Maximum DrawDown	Sharpe Ratio
ppo-base	1.39028	-0.061539038	1.713114246
PPO-CVaR(alpha=0.9)	1.297663828	-0.082591202	1.213504521
PPO-CVaR(alpha=0.1)	1.276795156	-0.070177493	1.240114107
PPO-CVaR(alpha=0.05)	1.244600625	-0.071948831	1.142339542

Table 5.2: PPO-CVaR: Portfolio Performance

Figure 5.12: Actions area: alpha=0.05

Figure 5.13: Actions area: alpha=0.1

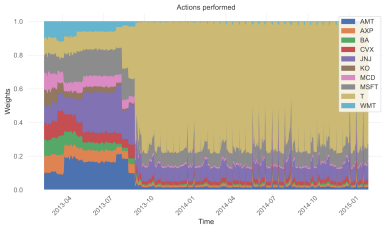


Figure 5.14: Actions area: alpha=0.9

Figure 5.15: Actions area for PPO-CVaR with market data. alpha=0.05, 0.1, 0.9

5.3.2. PPO-Expectile

Simulator PPO-Expectile follows the same experimental setup as with PPO-CVaR. Figure 5.16 again plots the Profit&Loss and the terminal portfolio value over the average of 800 evaluations. Here, the four variants are base (risk-neutral), $\tau = 0.1$ (risk-averse), $\tau = 0.5$ (risk-neutral with PPO-Expectile), and $\tau = 0.9$ (risk-seeking). We also see similar results, with the strategy changing with the risk parameter. The risk-neutral and risk-averse variants learn the same strategies and thus overlap on the plot. The base variant has a smaller spread than them, while the risk-seeking variant has the largest spread.

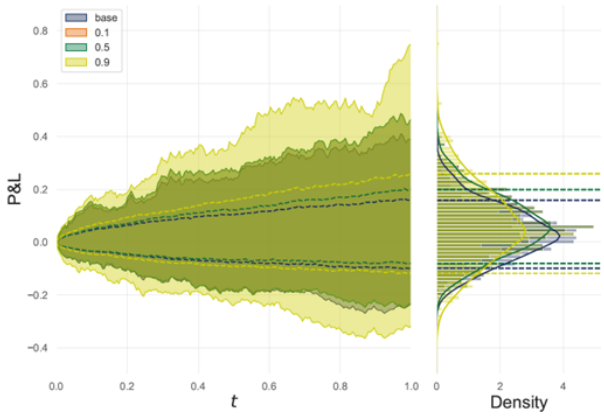


Figure 5.16: PPO-Expectile: Simulator evaluation

Figure 5.17 shows the average actions predicted by the agent over 800 evaluation episodes. As the agent moves towards risk-seeking ($\tau = 0.9$) behavior, it learns to allocate more towards asset 3, which is more volatile (but also has a higher return). It should be noted that the strategies learned are different from those learned by PPO-CVaR.

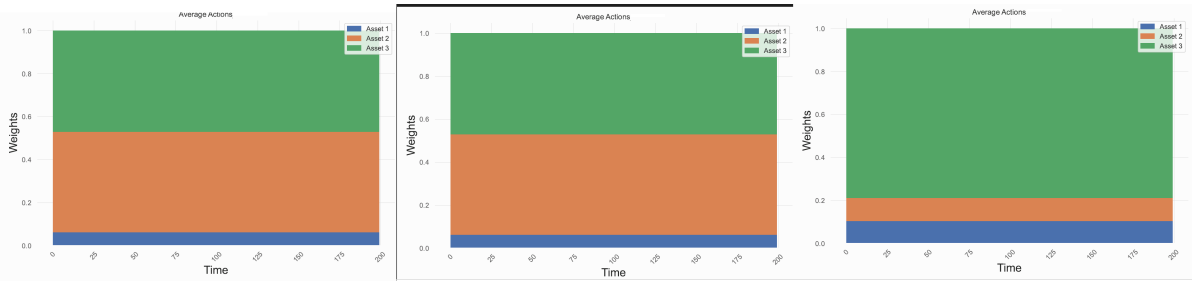


Figure 5.17: Average actions: PPO-Expectile

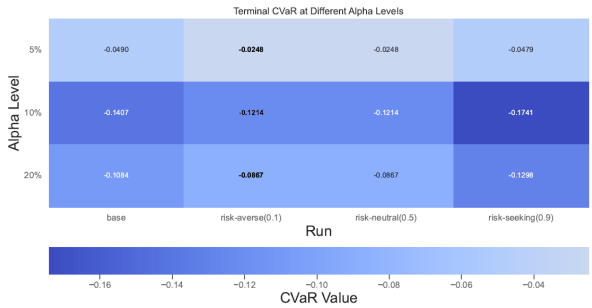


Figure 5.18: Terminal CVaR: PPO-Expectile

Market data We now switch again to a market data setup with ten assets and a sliding window setup, as described in 5.1. The portfolio performance under the 2-year test period is presented in table 5.3. The risk-sensitive variants have similar values, but some variants perform better on some values, such as Maximum Drawdown for $\tau = 0.5$. The action area plots for the test period can be seen in 5.22. There are also no clear patterns with the changing α parameter, with only minor changes in portfolio weights noticed with the changing τ parameter.

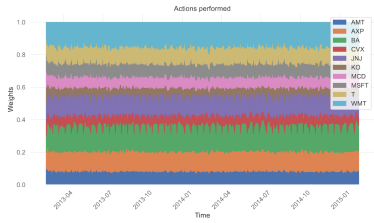


Figure 5.19: Actions area: tau=0.1

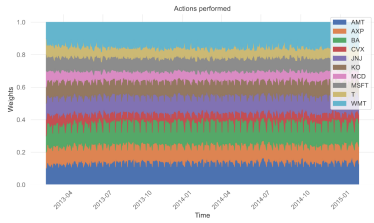


Figure 5.20: Actions area: tau=0.5

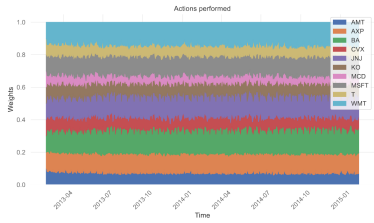


Figure 5.21: Actions area: tau=0.9

Figure 5.22: Actions area for PPO-Expectile with market data. tau=0.05, 0.1, 0.9

ID	Final Accumulative Portfolio	Maximum DrawDown	Sharpe Ratio
ppo-base	1.39028	-0.061539038	1.713114246
ppo-ex(tau=0.9)	1.3604025	-0.067382852	1.548774185
ppo-ex(tau=0.5)	1.368088438	-0.05957631	1.594194108
ppo-ex(tau=0.1)	1.363764375	-0.064055094	1.576983875

Table 5.3: Portfolio Performance Metrics

5.4. Single sample simulator

For this study, we focused on one risk management method: PPO-CVaR. PPO-CVaR and PPO-Expectile yielded similar outcomes, including both in the single-sample experiment, so the analysis

would have unnecessarily been complicated. Therefore, we used a setup similar to the sliding window approach, dividing the simulated data into 50 rounds of episodes. Here, each round utilized a single episode, and in each round, the model was trained for 10,000 steps, analogous to training for 10,000 steps in the market setup. However, this particular seed led to poor performance, as shown in Figure 5.23, which may explain the differences in the agents' behavior compared to the simulator-based evaluation.

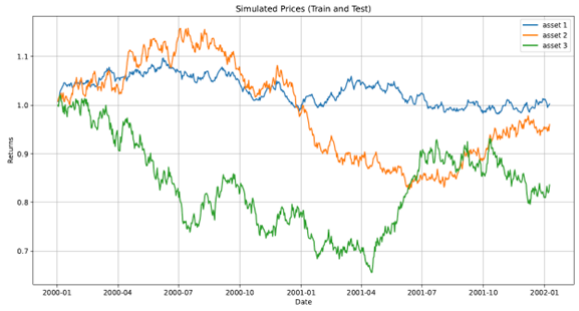


Figure 5.23: Single sample of simulated prices

The test results are illustrated in Figure 5.27. As risk sensitivity increased, the model learned to prioritize asset 1, a finding consistent with previous results obtained from the simulator-based experiments.

Figure 5.24: Actions area: alpha=0.05

Figure 5.25: Actions area: alpha=0.1

Figure 5.26: Actions area: alpha=0.9

Figure 5.27: Actions area for PPO-CVaR with a single sample of simulated prices. alpha=0.05, 0.1, 0.9

6

Discussion and Conclusions

6.1. Discussion

6.1.1. Discussion baseline

This project started by reproducing the results from the cross-sectional (CS) approach[7], which served as the baseline for the development on top of which the risk-sensitive methods in portfolio optimization were implemented. While the original results were successfully replicated, several challenges and limitations emerged that generally apply to using reinforcement learning (RL) for portfolio optimization.

The design of the reward function in RL is not straightforward in the context of financial markets where there is no universally accepted investment strategy. However, even a reward design that follows a certain investment strategy is not necessarily enough. For example, it is not necessarily true that incorporating a reward function based on Differential Sharpe Ratio[9] will lead to learning a better Sharpe Ratio. For instance, while Aboussalah, Xu, and Lee [7] introduced the alpha parameter to encourage the agent to outperform an equal-weighted portfolio, our experiments indicated that this term had a negligible impact on the agent's performance.

It is insufficient to evaluate the success of a portfolio solely based on metrics such as the final return. The agent's behavior throughout the investment period is equally critical. High returns and Sharpe Ratios may be observed, yet the agent might achieve these metrics by adopting highly risky strategies, such as going all-in on a single stock. This behavior underlines the necessity of reporting the portfolio weights, a practice that some studies fail to follow (Sood, Papasotiriou, Vaiciulis, *et al.* [9] and Jiang, Xu, and Liang [8]). Although Aboussalah, Xu, and Lee [7] does report portfolio weights, there are concerns that the reported performance is driven by the agent's tendency to remain close to equal weights, which is discussed further below.

Several issues were also identified with the results of the baseline study. The performance of the baseline model appears to be highly sensitive to the hyperparameters in the reward function. In particular, the gamma term in the baseline reward function, which has a higher scale than the log return, seems to dominate the reward, leading the agent to gravitate towards an equal-weighted portfolio rather than truly learning from the environment. Secondly, the paper makes the claim that the cross-sectional feature is ineffective with PPO due to random sampling from the buffer. However, this claim is not clearly substantiated and appears inconsistent with the results presented. Finally, there are concerns regarding the quality of the market data used. The two-year test period may not be sufficient, as it does not contain a full market cycle, the prices remain relatively stable, and daily returns are considered noisy in general[69]. All of these issues potentially undermine the robustness of the results.

6.1.2. Discussion risk-sensitive methods

Simulations In response to Research Question 1 (RQ1), our findings demonstrate a noticeable change in behavior between the risk-neutral baseline method and the risk-sensitive methods. Specifically, we observe different strategies employed by the PPO-CVaR and PPO-Expectile agents. Addressing RQ2, varying the risk parameter leads to different strategies. This is evident with the parameter α for PPO-CVaR and τ for PPO-Expectile. The behavior of the agents aligns with the changes in these risk parameters. For instance, when $\tau = 0.9$, the agent exhibits risk-seeking behavior, leading to the

widest P&L band and the highest allocation of weights to the more volatile asset. Additionally, our results show that the agent is indeed capable of minimizing risk when CVaR is used as a metric. It is important to clarify that the agent's objective is to minimize dynamic risk throughout the episode rather than focusing solely on static risk at the end of the episode. Therefore, while the CVaR metric is useful, it is unreliable in isolation. Notably, the baseline PPO does not achieve a better terminal CVaR than PPO-CVaR. Finally, approximately 1,000 simulated episodes are required, even in a simple case involving three assets. This underscores the necessity of careful consideration of market data, as significant amounts of data are needed to achieve reliable results even in relatively straightforward scenarios.

Market data When addressing RQ1 regarding the behavior of PPO-CVaR and PPO-Expectile with market data compared to risk-neutral methods, we see that these methods appear to be learning different strategies. However, this divergence in strategy does not translate into a significant difference in performance. Metrics such as portfolio return and maximum drawdown (max-DD) and the policies learned by the agents show no clear improvement or degradation. This suggests that while the risk-sensitive methods might be altering the approach to decision-making, they do not consistently outperform or under-perform risk-neutral strategies in this context. Regarding RQ2, a behavior change is observed with varying risk parameters, particularly for PPO-CVaR. The most risk-sensitive variant, with $\alpha = 0.9$, demonstrates a preference for investing in a low-volatility asset. This behavior indicates a potential shift in strategy aligned with the increased risk aversion. However, the underlying reasons for this asset selection remain unclear, especially given the complexity of the market data setup. There are other assets with similar risk profiles where this pattern does not occur, complicating the interpretation of this finding. For PPO-Expectile, no consistent patterns are observed in the agent's behavior, whether in terms of performance or policy learned, across different risk parameters. As a result, the results do not allow for a definitive answer to either RQ1 or RQ2 within the market data setup. The variations in strategy and performance are present but do not provide a clear direction or consistent trend, highlighting the challenges of interpreting agent behavior in complex financial environments.

Single episode simulation In the market setup using a single episode sampled from the simulator, we observe that the agent can learn reasonable policies similar to those learned in the simulator setup. As the agent's risk sensitivity increases, it increasingly focuses on the least volatile asset, demonstrating a clear shift in strategy in response to increased risk aversion. However, the specific strategy learned by the agent in this market setup differs from that observed in the simulator environment. This divergence in strategy appears logical when considering the price movements within the sample episode, suggesting that the agent is adapting to the unique characteristics of the market data in this particular scenario.

6.2. Conclusions and future work

6.2.1. Conclusions

This study explored the application of risk-sensitive Reinforcement Learning (RL) methods to portfolio optimization, focusing on understanding the behavior and performance of these methods compared to traditional risk-neutral approaches. A baseline cross-sectional (CS) approach was successfully implemented, replicating prior results. However, challenges emerged, particularly in designing effective reward functions and addressing the agent's behavior throughout the investment period. While high returns and favorable metrics were observed, agents occasionally adopted risky strategies, such as concentrating investments in single assets. This highlighted the importance of assessing portfolio weights and not just relying on performance metrics like Sharpe Ratio or final returns.

Investigating risk-sensitive methods revealed distinct portfolio strategies compared to the risk-neutral baseline, answering the first research question (RQ1). Both PPO-CVaR and PPO-Expectile exhibited different behaviors, with the risk parameters (α for PPO-CVaR and τ for PPO-Expectile) influencing the agent's risk appetite, addressing RQ2. For instance, with higher risk parameters, the PPO-CVaR adopted more conservative strategies, preferring lower-volatility assets. However, while these methods altered the decision-making process, they did not consistently outperform or underperform the baseline regarding key performance metrics like portfolio return and maximum drawdown.

The study also noted significant differences between simulated environments and real market data, with approximately 1,000 episodes required for reliable simulation results. In real market data, strategies learned by risk-sensitive methods were sometimes influenced by the volatility of specific assets but did not lead to consistently better outcomes. This supports the hypothesis (H3) that methods performing well in simulations may not necessarily translate to real-world success due to the unique characteristics and constraints of real financial data.

In conclusion, risk-sensitive RL methods provide distinct strategies for portfolio optimization, particularly under varying risk measures. However, translating these strategies into consistent real-world performance remains a challenge. The findings suggest that while risk-sensitive methods hold potential, careful consideration of market data, reward function design, and long-term evaluation is crucial for effective deployment in financial markets.

6.2.2. Future work

Future research stemming from this work can be pursued in several directions. Some possible directions discussed here are working with more complex simulator models to sample pricing data from, improving the state representation through better feature extraction and richer data, using a market/economic simulator to study the impact of ”black swan” events, and exploring how other Deep RL algorithms perform for portfolio optimization.

First, the simulator experiments conducted in this project employed relatively simple setups. Future investigations could benefit from exploring a broader range of simulation scenarios and incorporating more complex models beyond Geometric Brownian Motion.

Another potential avenue for improvement lies in the state space representation, which was not the primary focus of this project. In RL, the state space encapsulates all the information the agent uses to make decisions, and the adequacy of this representation is crucial for optimal performance. The current study utilized a one-dimensional convolutional neural network (1-D CNN) with returns, but this may not be sufficient for capturing the full complexity of the environment. Future work should explore more sophisticated data inputs and feature extraction methods. The price tensor used in this study is flexible and can be augmented with additional data types, such as financial ratios like Price-to-Earnings or Price-to-Book, as well as fundamental data like revenue and earnings[70]. Additionally, incorporating textual financial data, as demonstrated in recent studies, could enrich the state space and potentially improve decision-making[71], [72]. Regarding neural network architectures, there is also room to experiment with models better suited for time series and cross-sectional data, such as recurrent neural networks (RNNs) for time-series analysis and self-attention for cross-sectional analysis. It can also be interesting to look at transformer-based architectures, which have shown promise in portfolio optimization[73].

An important use case of risk-sensitive RL is how the agent adapts to bad economic or market conditions. Thus, developing a sophisticated market and economic simulator could provide a more realistic and challenging environment for training RL agents. For instance, simulating rare but impactful ”black swan” events could test the robustness of the algorithms under extreme conditions. Simple random events with a low probability of occurrence may not be sufficient to mimic the complexity of real-world markets, and more nuanced approaches to simulating such events should be explored.

Further work investigating different exploration methods and applying them to portfolio optimization. One interesting point is that lattice for stochastic policies is not applicable. For deterministic policies, we can use pink noise, an extension of the white noise (standard Gaussian) usually sampled from algorithms like TD3[74].

Lastly, this study focused on the Proximal Policy Optimization (PPO) algorithm, but there is potential value in investigating other deep RL algorithms. The deterministic policies used in algorithms like Deep Deterministic Policy Gradient (DDPG) and Twin Delayed Deep Deterministic Policy Gradient (TD3) encountered issues with extreme actions, suggesting that stochastic policies might offer a more reliable alternative. Future research could explore the performance of Soft Actor-Critic (SAC) as an alternative to PPO, given its reputation for being easier to work with and less dependent on hyperparameter optimization. These investigations could yield valuable insights into the relative strengths and weaknesses of different RL approaches in financial market applications.

A

Extra results

A.1. CS Reward function design

This section presents additional results obtained while working on reward function design. Three different approaches were explored: scaling the beta and gamma terms, modifying the gamma term to represent the difference from previous or benchmark weights, and employing the differential Sharpe ratio as the reward function.

A.1.1. Reward Scaling and tuning

The initial focus was on hyperparameter tuning and scaling of the beta and gamma terms. When scaling was applied, performance deteriorated as the agent concentrated on a single stock. Despite tuning efforts, no noticeable improvement was observed. According to the recommended reward range for PPO, the reward function should lie within (-1, 1) [75]. To adhere to this guideline, the reward function's alpha, beta, and gamma terms were scaled accordingly. Specifically, the beta term (volatility) was scaled up by a factor of 100, while the gamma term (maximum weights) was scaled down by a factor of 100, bringing them into the same range as the log return (reward and alpha term). The impact of this scaling can seen in the actions plot in A.1 and the portfolio return in A.2.

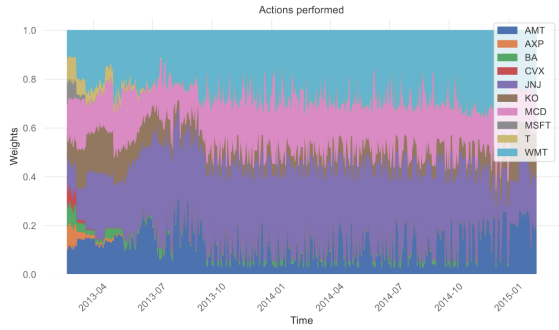


Figure A.1: Actions plot

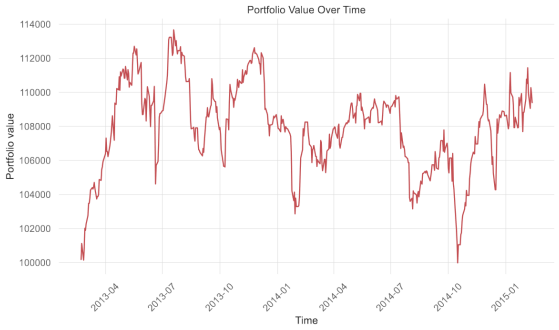


Figure A.2: Portfolio returns

Figure A.3: Reward scaling and tuning

The action plots show how the agent deviates from the equal weights behavior, learning to focus on a few stocks. However, from the portfolio return, it can be seen that the agent performance is worse than equal weights. After scaling the beta and gamma terms to the same scale as the log return, the run yielded worse performance, with the agent focusing primarily on a few stocks.

A.1.2. Change in gamma term

This experiment assessed the impact of replacing the gamma term with alternative formulations that would encourage diversification. Two approaches were tested: replacing the maximum weights term

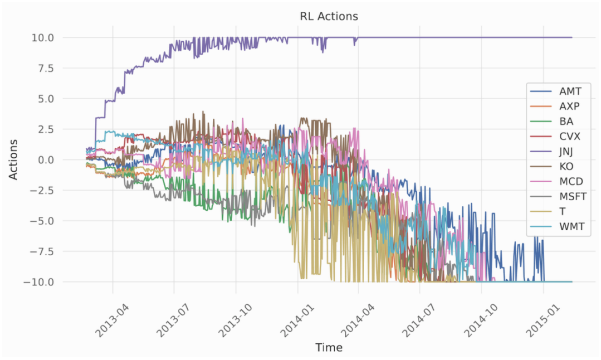


Figure A.4: Actions plot. Difference Previous Weights (DPR)

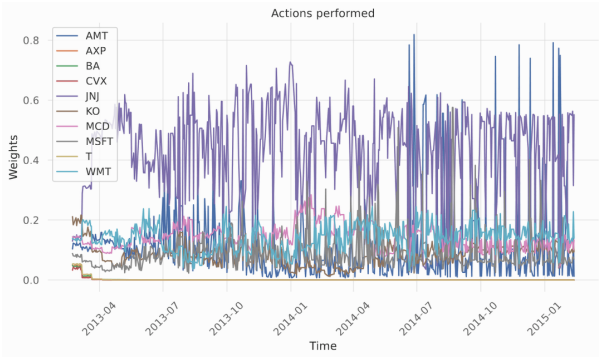


Figure A.5: Actions plot. Difference Equal Weights (DEQ)

with the difference from the previous or benchmark weights (denoted as DPR and DEQ). The results are presented in A.4 and A.5. However, in both cases, the agent continued to focus on one stock, and no significant improvement in performance was observed.

A.1.3. Differential SR as the reward

Following the approach outlined by Sood, Papasotiriou, Vaiciulis, *et al.* [9], an alternative reward function was tested based on the differential Sharpe ratio compared to the reward function used by Coache, Jaimungal, and Cartea [10]. The differential Sharpe Ratio essentially functions as a moving average of the Sharpe Ratio. More details can be found in Sood, Papasotiriou, Vaiciulis, *et al.* [9]. The results from changing to the Differential SR are presented in figure A.6. Similar to the previous experiments discussed in this section, the agent again learns to focus on a single stock.

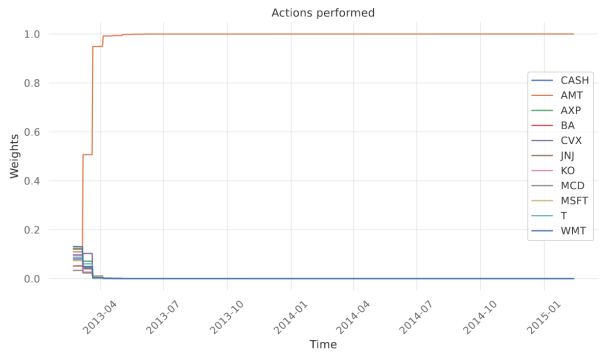


Figure A.6: Portfolio return, differential Sharpe Ratio

A.2. DDPG-Expectile

Initially, the plan was to utilize the DDPG-Expectile algorithm to explore a different risk measure, expectile, while adhering to elicitable functions. However, during implementation, an issue with extreme actions was encountered. Several attempts were made to mitigate this problem, including gradient clipping, increasing the interval between gradient updates, raising the exploration noise variance, reducing the learning rate, and replacing DDPG with TD3. Despite these efforts, the actions predicted by the agent consistently approached the minimum and maximum bounds of the action space, defined as [-10, 10]. These actions resembled discrete buy-sell signals rather than appropriate weights for stock allocation within a portfolio. The predicted actions converged to extreme values within approximately ten gradient update steps.

The issue of extreme actions is a known problem, previously seen in the baseline work when working with DDPG and DPG, as well as other works, including Amrouni [76]. This challenge appears to stem from the difficulty of learning deterministic policies, and to date, no solution seems to have been identified. A potential workaround involves transitioning to a discrete action space that aligns more closely with buy-sell signals.

Further experiments were not conducted after identifying the extreme actions issue, and the focus shifted to the PPO-Expectile algorithm. Thus, only results from the market data setup in section 5.1.2 are available, presented here. The relative performance of the various approaches during the test period is shown in Figure A.7, where the expectile method outperformed both the DDPG-base and EQB. However, no consistent patterns emerged regarding performance across the risk parameter tau. Figures A.8 and A.9 illustrate the extreme actions issue. The left panel shows a histogram of the extreme actions observed over time, with most falling at the extremes. The right panel depicts the un-normalized actions predicted by the policy network, where extreme action trends persist throughout the test period.

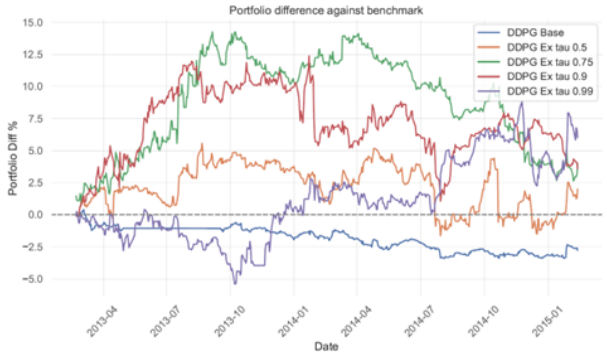


Figure A.7: DDPG-Expectile: Portfolio Returns

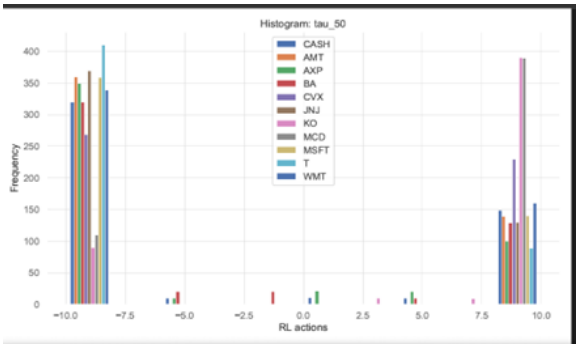


Figure A.8: Histogram of actions

Figure A.9: Un-normalized RL actions

Figure A.10: DDPG-Expectile: Extreme actions for tau=0.5

A.3. Exploration Noise

Figure A.11 plots the baseline CS PPO portfolio returns against the Lattice exploration noise. The plot shows an average of 5 runs with 1σ deviations. Figure A.12 shows the actions learned by the lattice PPO method, which are very different from the close-to-equal-weights actions learned by the base PPO. While the performance of the two methods is similar, the actions learned are very different. This shows that EQB is not the only possible strategy with a good performance for the market data setup and that exploration is currently heavily under-utilized for portfolio optimization.

Figure A.11: Exploration noise: Portfolio Returns

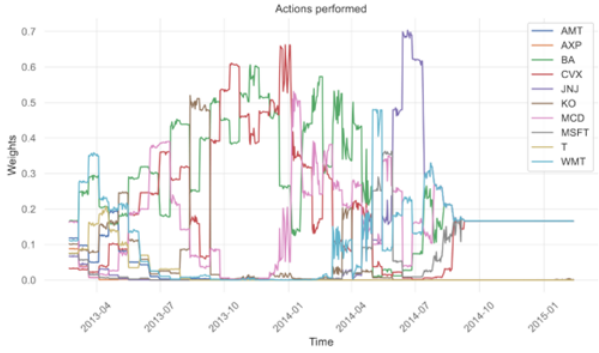


Figure A.12: Exploration noise: Actions

A.4. Implementation

- **Frameworks:** Stable-Baselines3[1], PyTorch[2]
- **Libraries:** FinRL[3], Portfolio Optimization Environment[77]
- **Experiment tracking:** Weights and Biases[4]
- **Setup used:** Surface Laptop 5 (Intel Core i7-1255U, 16GB)

[1] https://github.com/DLR-RM/stable-baselines3
[2] https://pytorch.org/
[3] https://github.com/AI4Finance-Foundation/FinRL
[4] https://wandb.ai/

Bibliography

[1] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017, Publisher: Nature Publishing Group.

[2] OpenAI, : C. Berner, *et al.*, *Dota 2 with Large Scale Deep Reinforcement Learning*, Version Number: 1, 2019. DOI: 10.48550/ARXIV.1912.06680. [Online]. Available: https://arxiv.org/abs/1912.06680 (visited on 08/27/2024).

[3] OpenAI, J. Achiam, S. Adler, *et al.*, *GPT-4 Technical Report*, Version Number: 6, 2023. DOI: 10.48550/ARXIV.2303.08774. [Online]. Available: https://arxiv.org/abs/2303.08774 (visited on 08/27/2024).

[4] D. Li and W.-L. Ng, "Optimal dynamic portfolio selection: Multiperiod mean-variance formulation," *Mathematical finance*, vol. 10, no. 3, pp. 387–406, 2000.

[5] Y. Jiang, Q. Liu, X. Ma, *et al.*, "Learning Diverse Risk Preferences in Population-Based Self-Play," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 11, pp. 12 910–12 918, Mar. 2024, ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v38i11.29188. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29188 (visited on 05/24/2024).

[6] S. Marzban, E. Delage, and J. Y.-M. Li, "Deep reinforcement learning for option pricing and hedging under dynamic expectile risk measures," en, *Quantitative Finance*, vol. 23, no. 10, pp. 1411–1430, Oct. 2023, ISSN: 1469-7688, 1469-7696. DOI: 10.1080/14697688.2023.2244531. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/14697688.2023.2244531 (visited on 02/26/2024).

[7] A. M. Aboussalah, Z. Xu, and C.-G. Lee, "What is the value of the cross-sectional approach to deep reinforcement learning?" *SSRN Electronic Journal*, 2020. DOI: 10.2139/ssrn.3748130. [Online]. Available: https://ssrn.com/abstract=3748130.

[8] Z. Jiang, D. Xu, and J. Liang, "A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem," *Research Papers in Economics*, 2017.

[9] S. Sood, K. Papasotiriou, M. Vaiciulis, and T. Balch, "Deep Reinforcement Learning for Optimal Portfolio Allocation: A Comparative Study with Mean-Variance Optimization," *FinPlan 2023*, p. 21, 2023. [Online]. Available: https://icaps23.icaps-conference.org/papers/finplan/FinPlan23_paper_4.pdf.

[10] A. Coache, S. Jaimungal, and Á. Cartea, "Conditionally elicitable dynamic risk measures for deep reinforcement learning," *SSRN Electronic Journal*, 2022. DOI: 10.2139/ssrn.4149461. [Online]. Available: https://ssrn.com/abstract=4149461.

[11] A. Coache and S. Jaimungal, "Reinforcement learning with dynamic convex risk measures," *Mathematical Finance*, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/mafi.12388.

[12] D. Pacheco Aznar, "Portfolio management: A deep distributional rl approach," *Available at SSRN*, 2023.

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

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

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust Region Policy Optimization,” in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 1889–1897. [Online]. Available: https://proceedings.mlr.press/v37/schulman15.html.

[17] R. C. Merton, “On estimating the expected return on the market: An exploratory investigation,” *Journal of financial economics*, vol. 8, no. 4, pp. 323–361, 1980.

[18] T.-M. Chow, J. C. Hsu, L.-L. Kuo, and F. Li, “A study of low-volatility portfolio construction methods,” *The Journal of Portfolio Management*, vol. 40, no. 4, pp. 89–105, 2014, Publisher: Institutional Investor Journals Umbrella.

[19] H. Föllmer, A. Schied, and H. Föllmer, *Stochastic Finance : An Introduction in Discrete Time*. Berlin/Boston, GERMANY: Walter de Gruyter GmbH, 2011, ISBN: 978-3-11-021805-3. [Online]. Available: http://ebookcentral.proquest.com/lib/delft/detail.action?docID=772998.

[20] R. T. Rockafellar and S. Uryasev, “Optimization of conditional value-at-risk,” *The Journal of Risk*, vol. 2, no. 3, pp. 21–41, 2000. DOI: 10.21314/jor.2000.038.

[21] F. Bellini and E. Di Bernardino, “Risk management with expectiles,” en, *The European Journal of Finance*, vol. 23, no. 6, pp. 487–506, May 2017, ISSN: 1351-847X, 1466-4364. DOI: 10.1080/1351847X.2015.1052150. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/1351847X.2015.1052150 (visited on 04/19/2024).

[22] A. Ruszczyński, “Risk-averse dynamic programming for markov decision processes,” *Mathematical programming*, vol. 125, pp. 235–261, 2010. [Online]. Available: link.springer.com/article/10.1007/s10107-010-0393-3.

[23] B. Acciaio and I. Penner, “Dynamic risk measures,” *Advanced mathematical methods for finance*, pp. 1–34, 2011. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-18412-3_1.

[24] W. Fedus, C. Gelada, Y. Bengio, M. G. Bellemare, and H. Larochelle, “Hyperbolic discounting and learning over multiple horizons,” *arXiv preprint arXiv:1902.06865*, 2019.

[25] S. Marzban, E. Delage, and J. Y.-M. Li, “Deep reinforcement learning for option pricing and hedging under dynamic expectile risk measures,” *Quantitative Finance*, vol. 23, no. 10, pp. 1411–1430, 2023.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[27] H. Markowitz, “Portfolio selection,” *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952, ISSN: 00221082, 15406261. [Online]. Available: http://www.jstor.org/stable/2975974 (visited on 02/09/2024).

[28] J. L. Kelly, “A new interpretation of information rate,” *the bell system technical journal*, vol. 35, no. 4, pp. 917–926, 1956.

[29] E. O. Thorp, “The kelly criterion in blackjack sports betting, and the stock market,” in *Handbook of asset and liability management*, Elsevier, 2008, pp. 385–428.

[30] A. S. Alcántar, “Semi-variance optimization for the components of the dow jones industrial average index,” *Contaduría y administración*, vol. 68, no. 4, pp. 1–17, 2023.

[31] B. Hambly, R. Xu, and H. Yang, “Recent advances in reinforcement learning in finance,” *Mathematical Finance*, vol. 33, no. 3, pp. 437–503, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/mafi.12382.

[32] H. Park, M. K. Sim, and D. G. Choi, “An intelligent financial portfolio trading strategy using deep q-learning,” *Expert Systems with Applications*, vol. 158, p. 113 573, 2020.

[33] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *International conference on machine learning*, Pmlr, 2014, pp. 387–395.

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

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

[36] X.-Y. Liu, Z. Xiong, S. Zhong, H. Yang, and A. Walid, “Practical deep reinforcement learning approach for stock trading,” *arXiv preprint arXiv:1811.07522*, 2018.

[37] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic Policy Gradient Algorithms,” in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, Issue: 1, vol. 32, Bejing, China: PMLR, Jun. 2014, pp. 387–395. [Online]. Available: https://proceedings.mlr.press/v32/silver14.html.

[38] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction* (Adaptive computation and machine learning series), Second edition. Cambridge, Massachusetts: The MIT Press, 2018, ISBN: 978-0-262-03924-6.

[39] M. Wang and H. Ku, “Risk-sensitive policies for portfolio management,” *Expert Systems with Applications*, vol. 198, p. 116 807, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422002640.

[40] H. Niu, S. Li, and J. Li, “Metatrader: An reinforcement learning approach integrating diverse policies for portfolio optimization,” in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 1573–1583. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3511808.3557363.

[41] Y. Zhang, P. Zhao, Q. Wu, B. Li, J. Huang, and M. Tan, “Cost-sensitive portfolio selection via deep reinforcement learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 236–248, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9031418.

[42] C. Jiang and J. Wang, “A portfolio model with risk control policy based on deep reinforcement learning,” *Mathematics*, vol. 11, no. 1, p. 19, 2022. [Online]. Available: https://www.mdpi.com/2227-7390/11/1/19.

[43] C. Betancourt and W.-H. Chen, “Deep reinforcement learning for portfolio management of markets with a dynamic number of assets,” *Expert Systems with Applications*, vol. 164, p. 114 002, 2021.

[44] P. La and M. Ghavamzadeh, “Actor-critic algorithms for risk-sensitive mdps,” *Advances in neural information processing systems*, vol. 26, 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/hash/eb163727917cbba1eea208541a643e74-Abstract.html.

[45] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, “Risk-constrained reinforcement learning with percentile risk criteria,” *Journal of Machine Learning Research*, vol. 18, no. 167, pp. 1–51, 2018. [Online]. Available: https://www.jmlr.org/papers/v18/15-636.html.

[46] M. Petrik and D. Subramanian, “An approximate solution method for large risk-averse markov decision processes,” *arXiv preprint arXiv:1210.4901*, 2012.

[47] D. Nass, B. Belousov, and J. Peters, “Entropic risk measure in policy search,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 1101–1106. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8967699.

[48] N. Bäuerle and A. Glauner, “Minimizing spectral risk measures applied to markov decision processes,” *Mathematical Methods of Operations Research*, vol. 94, pp. 35–69, 2021.

[49] W. R. Clements, B. V. Delft, B.-M. Robaglia, R. B. Slaoui, and S. Toth, *Estimating risk and uncertainty in deep reinforcement learning*, 2020. arXiv: 1905.09638 [cs.LG].

[50] H. Liang and Z.-q. Luo, *Regret bounds for risk-sensitive reinforcement learning with lipschitz dynamic risk measures*, 2023. arXiv: 2306.02399 [cs.LG].

[51] S. Pesenti, “Risk budgeting allocation for dynamic risk measures,” in *2023 Fall Eastern Sectional Meeting*, AMS.

[52] X. Ma, Y. Yang, H. Hu, *et al.*, “Offline reinforcement learning with value-based episodic memory,” *arXiv preprint arXiv:2110.09796*, 2021.

[53] X. Ma, L. Xia, Z. Zhou, J. Yang, and Q. Zhao, *Dsac: Distributional soft actor critic for risk-sensitive reinforcement learning*, 2020. arXiv: 2004.14547 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2004.14547.

[54] A. Balbás, J. Garrido, and S. Mayoral, "Properties of distortion risk measures," *Methodology and Computing in Applied Probability*, vol. 11, no. 3, pp. 385–399, 2009.

[55] A. Tversky and D. Kahneman, "Advances in prospect theory: Cumulative representation of uncertainty," *Journal of Risk and uncertainty*, vol. 5, pp. 297–323, 1992.

[56] S. S. Wang, "A class of distortion operators for pricing financial and insurance risks," *The Journal of Risk and Insurance*, vol. 67, no. 1, pp. 15–36, 2000, ISSN: 00224367, 15396975. [Online]. Available: http://www.jstor.org/stable/253675 (visited on 10/25/2023).

[57] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2012, pp. 5026–5033.

[58] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[59] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, "Robust reinforcement learning: A review of foundations and recent advances," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 276–315, 2022, ISSN: 2504-4990. DOI: 10.3390/make4010013. [Online]. Available: https://www.mdpi.com/2504-4990/4/1/13.

[60] S. Jaimungal, S. M. Pesenti, Y. S. Wang, and H. Tatsat, "Robust risk-aware reinforcement learning," *SIAM Journal on Financial Mathematics*, vol. 13, no. 1, pp. 213–226, 2022. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/21M144640X.

[61] J. Queeney and M. Benosman, "Risk-averse model uncertainty for distributionally robust safe reinforcement learning," *arXiv preprint arXiv:2301.12593*, 2023.

[62] T. Kanazawa, H. Wang, and C. Gupta, "Distributional actor-critic ensemble for uncertainty-aware continuous control," in *2022 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2022, pp. 1–10. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9892771.

[63] X. Ma, S. Ma, L. Xia, and Q. Zhao, "Mean-semivariance policy optimization via risk-averse reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 75, pp. 569–595, 2022. [Online]. Available: https://dl.acm.org/doi/pdf/10.1613/jair.1.13833.

[64] D. W. Andrews, "Cross-section regression with common shocks," *Econometrica*, vol. 73, no. 5, pp. 1551–1585, 2005, Publisher: Wiley Online Library.

[65] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, *High-Dimensional Continuous Control Using Generalized Advantage Estimation*, Version Number: 6, 2015. DOI: 10.48550/ARXIV.1506.02438. [Online]. Available: https://arxiv.org/abs/1506.02438 (visited on 07/05/2024).

[66] S. Huang, R. F. J. Dossa, A. Raffin, A. Kanervisto, and W. Wang, "The 37 implementation details of proximal policy optimization," *The ICLR Blog Track 2023*, 2022.

[67] A. S. Chiappa, A. Marin Vargas, A. Huang, and A. Mathis, "Latent exploration for Reinforcement Learning," in *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36, Curran Associates, Inc., 2023, pp. 56 508–56 530. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/b0ca717599b7ba84d5e4f4c8b1ef6657-Paper-Conference.pdf.

[68] K. D. West, "Chapter 3 Forecast Evaluation," en, in *Handbook of Economic Forecasting*, vol. 1, Elsevier, 2006, pp. 99–134, ISBN: 978-0-444-51395-3. DOI: 10.1016/S1574-0706(05)01003-7. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1574070605010037 (visited on 09/11/2024).

[69] J. Brogaard, T. H. Nguyen, T. J. Putnins, and E. Wu, “What Moves Stock Prices? The Roles of News, Noise, and Information,” en, *The Review of Financial Studies*, vol. 35, no. 9, I. Goldstein, Ed., pp. 4341–4386, Aug. 2022, ISSN: 0893-9454, 1465-7368. DOI: 10.1093/rfs/hhab137. [Online]. Available: https://academic.oup.com/rfs/article/35/9/4341/6493385 (visited on 09/11/2024).

[70] X. Ren, Z. Jiang, and J. Su, “The use of features to enhance the capability of deep reinforcement learning for investment portfolio management,” in *2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*, IEEE, 2021, pp. 44–50.

[71] Z. Wang, B. Huang, S. Tu, K. Zhang, and L. Xu, “Deeptrader: A deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 643–650.

[72] Y. Ye, H. Pei, B. Wang, *et al.*, “Reinforcement-learning based portfolio management with augmented asset movement prediction states,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 1112–1119.

[73] T. W. Kim and M. Khushi, “Portfolio optimization with 2d relative-attentional gated transformer,” in *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, IEEE, 2020, pp. 1–6.

[74] O. Eberhard, J. Hollenstein, C. Pinneri, and G. Martius, “Pink Noise Is All You Need: Colored Noise Exploration in Deep Reinforcement Learning,” in *Proceedings of the Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=hQ9V5QN27eS.

[75] A. Jones, *Debugging Reinforcement Learning Systems*. [Online]. Available: https://andyljones.com/posts/rl-debugging.html (visited on 09/11/2024).

[76] S. Amrouni, *Selimamrouni/Deep-Portfolio-Management-Reinforcement-Learning*, original-date: 2018-07-08T15:56:59Z, Sep. 2024. [Online]. Available: https://github.com/selimamrouni/Deep-Portfolio-Management-Reinforcement-Learning (visited on 09/11/2024).

[77] C. d. S. B. Costa and A. H. R. Costa, “POE: A General Portfolio Optimization Environment for FinRL,” in *Anais do II Brazilian Workshop on Artificial Intelligence in Finance*, SBC, 2023, pp. 132–143.