Institute of Operations Research
Analytics and Statistics
Prof. Dr. Oliver Grothe

Seminar Thesis

Reinforcement Learning for Portfolio Optimization: A Comparison of PPO against MVO

Written by **Yanting Liu**
Matr. No. **2320538**
M.Sc. Economathematics

Supervisor: **Bolin Liu**

I hereby confirm truthfully that I have authored this **seminar thesis** independently and without the use of source material and aids other than those stated, that I have marked all passages literally or textually adapted from other sources as such and that I have respected the statutes of the Karlsruhe Institute of Technology (KIT) for ensuring good scientific practice.

Generative AI tools were used during the preparation of this thesis as follows:

- **ChatGPT (GPT-4o by OpenAI)** – used for checking and improving the clarity and fluency of formulations and generating code snippets relevant to the analysis.
- **Google Jules** – used for code generation and review, especially for translating implementation frameworks into working Python code.

I confirm that these tools served only as writing and coding assistants and did not replace my own scientific analysis. I remained fully responsible for evaluating, selecting, adapting, and integrating any AI-generated output. The intellectual work and decisions reflected in this thesis are entirely my own.

Karlrsuhe, 04.07.2025

. . . . . .
(Yanting Liu)

Contents

1 Introduction

This seminar thesis aims to replicate and critically evaluate the findings of Sood et al. (2023), who propose a reinforcement learning-based approach to portfolio optimization.

Portfolio optimization refers to the task of allocating capital across a set of financial assets to maximize expected returns while controlling for risk. One of the foundational frameworks in this field is mean-variance optimization introduced by Markowitz (1952), which derives closed-form optimal weights based on estimated means and covariances of returns. Yet MVO is sensitive to estimation error and relies on strong assumptions.

Over the last two decades, machine learning methods have gained traction. RL does not assume static distributions and enables learning dynamic allocation strategies with sequential feedback. It also provides a natural way to integrate transaction costs, constraints, and nonlinear reward preferences.

The idea of applying sequential decision-making to portfolio optimization dates back to Elton Elton & Gruber (1971), who suggested the problem might lend itself to dynamic programming. This remained theoretical until Neuneier formalized the task as a Markov Decision Process (MDP) using Q-learning and neural networks Neuneier (1995). Moody and Saffell Moody et al. (1998) later extended this by introducing a differentiable Sharpe ratio, resolving a key limitation of classic RL methods in handling reward averaging over time.

Since then, RL-based portfolio strategies have evolved rapidly. The study by Sood et al. (2023) combines the differential Sharpe ratio with Proximal Policy Optimization (PPO), a stable and widely used policy gradient method, and reports substantial outperformance relative to traditional MVO benchmarks on U.S. equity sector data.

This paper aims to replicate these findings and critically evaluate their robustness. We test whether the reported performance holds under different datasets and more realistic evaluation settings. Our analysis highlights several methodological issues, such as the use of a short 60-day lookback for MVO and a policy warm-start mechanism in DRL training that may introduce structural unfairness. We also extend the evaluation to MSCI country indices and compare all models against simple passive benchmarks.

This thesis is structured as follows. Section 2 introduces the portfolio optimization problem and the PPO algorithm in RL. Section 3 outlines the code implementation of RL and MVO methods as well as training and evaluation pipelines. Section 4 presents empirical results on two datasets: S&P 500 sector indices (replicating Sood et al.) and MSCI country indices (as an out-of-domain test case). Section 5 summarizes key findings, limitations, and possible extensions.

2 Theoretical Background

2.1 Problem Statement

Portfolio optimization refers to the problem of allocating wealth across a set of financial assets in order to achieve desirable investment outcomes, typically balancing the objectives of high return and low risk. In its most basic form, the investor selects a portfolio weight vector $\mathbf{w}_t \in \mathbb{R}^N$ at each time t, where N is the number of available assets. The weights should satisfy some budget constraint, and in long-only settings we require that all components $w_t^{(i)} \geq 0$.

Let $\mathbf{r}_t \in \mathbb{R}^N$ denote the vector of asset returns at time t, modeled as a random variable. The corresponding portfolio return is then given by $R_t = \mathbf{w}_t^\top \mathbf{r}_t$.

Portfolio optimization methods aim to select $\mathbf{w}_t$ to maximize a particular notion of performance, which may focus on cumulative return $\sum_t R_t$, or more typically, on a *risk-adjusted* objective that accounts for the variance of returns.

Among such objectives, the *Sharpe ratio* (1966) is one of the most widely used. It quantifies the excess return per unit of risk and is defined as

$$\text{Sharpe Ratio} = \frac{\mathbb{E}[R_t] - R_f}{\sigma(R_t)} \tag{2.1}$$

where R_f is the risk-free rate and σ the standard deviation. While alternative risk-adjusted metrics such as the Sortino ratio (1991), Calmar ratio (1991), or Omega ratio (2002) exist, this study focuses exclusively on the Sharpe ratio for both evaluation and optimization.

In practice, portfolio optimization is complicated by many real-world challenges: returns are non-stationary and difficult to estimate reliably; volatility is time-varying; and optimization procedures are often highly sensitive to estimation error in inputs. These difficulties can lead to unstable or concentrated portfolios that perform poorly out-of-sample. Additionally, frequent portfolio rebalancing incurs transaction costs, which can substantially undermine net returns.

2.2 Mean-Variance Optimization and Modern Portfolio Theory

Modern Portfolio Theory (MPT), introduced by Markowitz (1952) provides a systematic way to construct portfolios that balance the trade-off between risk and return. The core insight of MPT is simple but powerful: diversifying across multiple assets can reduce the overall risk of a portfolio without necessarily sacrificing expected return.

At first glance, one might think that the best investment strategy is to allocate all capital to the asset with the highest expected return. However, this approach is dangerous because it ignores the risk associated with that asset. High returns often come with high volatility, meaning the asset's value can fluctuate widely and unpredictably.

Instead, combining assets with different risk-return profiles can lead to more stable portfolios. This principle is known as *diversification*, and it lies at the heart of Markowitz's theory.

Optimization Problem

Suppose we have N assets, and we want to decide how to allocate our wealth among them. We define a portfolio weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_N)^\top$, where w_i represents the fraction of capital invested in asset i. The weights must satisfy the budget constraint:

$$\sum_{i=1}^{N} w_i = 1.$$

In a long-only portfolio (which we assume in this study), all weights must also be non-negative: $w_i \geq 0$.

Let $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_N)^\top$ be the vector of expected returns for each asset, and let Σ be the $N \times N$ covariance matrix of asset returns. Then the expected return of the portfolio is $\mu_p = \mathbf{w}^\top \boldsymbol{\mu}$ and the risk (standard deviation) of the portfolio is $\sigma_p = \sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}$.

Markowitz posed the problem of portfolio selection as an optimization task. A common formulation is to *minimize portfolio risk* subject to achieving at least a target expected return:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbf{w}^\top \Sigma \mathbf{w} \\ \text{subject to} \quad & \mathbf{w}^\top \boldsymbol{\mu} \geq \mu^*, \\ & \sum_{i=1}^{N} w_i = 1, \\ & w_i \geq 0 \quad \text{for all } i. \end{aligned} \tag{2.2}$$

Here, μ^* is a desired level of expected return. This problem is convex and can be solved efficiently.

Solving the Optimization Problem

By solving the optimization problem for different values of μ^*, one obtains a set of optimal portfolios known as the *efficient frontier*. Each point on this curve represents a portfolio that offers the lowest possible risk for a given return. Investors can then select a point on this frontier depending on their individual risk tolerance.

TODO : add figure

An alternative objective is to directly maximize the *Sharpe ratio* as defined in equation 2.1. In essence, the Sharpe ratio quantifies how much excess return an investor earns per unit of risk. Maximizing the Sharpe ratio also avoids the need to pre-specify a target return μ^* and leads to a single-objective optimization problem. This approach is commonly used in practice and is also adopted in our evaluation framework.

Limitations

Despite its theoretical appeal, Mean-Variance Optimization has several well-documented limitations. It is highly sensitive to errors in the estimation of expected returns and covariances, often resulting in portfolios that are overly concentrated in a few assets or change drastically in response to small input variations.

TODO : add simple example with citation to DeMiguel

DeMiguel et al. (2007)

Consider the following extreme two-asset example. Suppose that the true per annum mean and volatility of returns for both assets are the same, 8% and 20%, respectively, and that the correlation is 0.99. In this case, because the two assets are identical, the optimal mean-variance weights for the two assets would be 50%. If, on the other hand, the mean return on the first asset is not known and is estimated to be 9% instead of 8%, then the mean-variance model would recommend a weight of 635% in the first asset and -535% in the second. That is, the optimization tries to exploit even the smallest difference in the two assets by taking extreme long and short positions without taking into account that these differences in returns may be the result of estimation error. As we describe in Section 3, the weights from mean-variance optimization when using actual data and more than just two assets are even more extreme than the weights in the given example.

TODO : move this to results section ?

DeMiguel et al. (2007) actually show that for many financial datasets on monthly frequency, the MVO optimized model actually performs worse than even the naive 1/N portfolio. And that only by using future data to estimate mu and covariance does the MVO strategy outperform the 1/N portfolio. ie in their setting they evaluate on 1980 to 2000. and if we use the entire 20 years to estimate mu and sigma, and calculate one single optimal portfolio for this time period, does the sharpe ratio outperform the 1/N portfolio.

Naive 1/N strategy is actually quite similar to just simply buy and holding the SnP500 index directly at the start of the eval period ? at least it roughly lines up, excep some slight mispricings of the sector ETFs i guess ?

actually with monthly data the divergence between 1/N and simply buy and hold SnP500 is quite large, especially compared to the daily data ... why is this the case ???

These limitations have motivated numerous extensions and alternatives, including robust optimization, shrinkage estimators such as Ledoit-Wolf, Bayesian methods, and more.

Ledoit–Wolf Shrinkage Estimator

The *Ledoit–Wolf shrinkage estimator* Ledoit & Wolf (2004) improves the stability of the sample covariance matrix S by shrinking it towards a structured target, i.e. the identity matrix I. Like in ridge regression, this introduces bias to reduce variance, which can improve out-of-sample performance—especially when the number of observations is small relative to the number of variables (i.e. number of assets).

The regularised (oracle) covariance is

$$\Sigma^* = (1 - \lambda)S + \lambda\mu I \tag{2.3}$$

where S is the empirical covariance matrix, I the identity matrix, $\lambda = \frac{\beta^2}{\alpha^2+\beta^2}$ and $\mu = \langle \Sigma, I \rangle$ is the average eigenvalue of the theoretical "true" covariance matrix or average "true" variance.

- $\alpha^2 = \|\Sigma - \mu I\|^2$ (deviation from target)
- $\beta^2 = \mathbb{E}[\|S - \Sigma\|^2]$ (estimation error)

Intuitively, the shrinkage intensity $\frac{\beta^2}{\alpha^2+\beta^2}$ increases with the estimation error β^2, applying more shrinkage when the estimate S is less reliable. More shrinkage means higher weighting to the diagonal matrix containing the average variance. In fact this is the optimal linear combination that minimizes $\mathbb{E}[|\Sigma^* - \Sigma|^2]$, which decomposes into bias squared plus variance. The shrinkage target μI has zero variance but is biased, while the sample covariance S is unbiased but high-variance. The estimator balances the two.

But, Σ^* is not yet a *bona fide* estimator since it depends on the unknown Σ in the shrinkage factor or to be precise in α and β as well as μ. Therefore it was denoted as the "oracle" estimator. In practice, Ledoit and Wolf propose using plug-in estimators for each unknown quantity in the oracle estimator Σ^*:

$$S_n^* = (1 - \hat{\lambda})S + \hat{\lambda} m_n I \tag{2.4}$$

where S again is the empirical covariance matrix, $m_n = \langle S_n, I_n \rangle$ is the empirical average variance estimate and $\hat{\lambda} = \frac{b_n^2}{d_n^2}$

- $b_n^2 = \min(\frac{1}{n^2} \sum_{k=1}^{n} \|x_n^k (x_n^k)' - S_n\|^2, d_n^2)$
- $d_n^2 = \|S_n - m_n I_n\|^2$

This yields a consistent, bona fide estimator with the same asymptotic loss as the oracle:

$$\|S_n^* - \Sigma_n^*\| \xrightarrow{q.m.} 0, \quad \mathbb{E}[\|S_n^* - \Sigma_n\|^2] - \mathbb{E}[\|\Sigma_n^* - \Sigma_n\|^2] \to 0. \tag{2.5}$$

S_n^* is shown in the paper (Theorem 3.7) to be asymptotically optimal within a broad class of linear shrinkage estimators under Frobenius norm loss.

2.3 Reinforcement Learning and Proximal Policy Optimization

Many real-world decision-making problems can be modeled as a *Markov Decision Process* (MDP), where decisions must be made sequentially and each decision influences the future. In an MDP, the environment is assumed to have the *Markov property*: the next state and reward depend only on the current state and action, not on the full history.

To illustrate this, consider a robot navigating a maze. At each step, the robot chooses a direction to move. Its goal is to reach a target location with as few steps as possible. Although the robot cannot foresee the full path in advance, it can learn which immediate choices lead to long-term success by observing how its actions change the environment over time.

This idea is directly relevant in finance. A portfolio manager repeatedly adjusts asset allocations. The reward — such as the long-term, risk-adjusted return — depends on how these decisions unfold across time. Reinforcement Learning (RL) offers a systematic approach to making such sequential decisions under uncertainty.

Formally, an MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where:

- $\mathcal{S}$ is the set of possible states,
- $\mathcal{A}$ is the set of possible actions,
- $P(s' \mid s, a)$ is the transition probability to state s' given current state s and action a,
- $r(s, a)$ is the reward function,
- $\gamma \in [0, 1)$ is a discount factor that prioritizes immediate rewards.

What is Reinforcement Learning?

The agent's goal is to find a *policy* π — a mapping from states to actions — that maximizes expected cumulative reward over time. Reinforcement Learning refers to the family of algorithms

that learn such a policy through interaction with the environment.

Algorithm 1: Agent–Environment Interaction in Reinforcement Learning

1: **for** each time step $t = 0, 1, 2, \ldots$ **do**
2: Observe current state s_t from the environment
3: Select action $a_t \sim \pi(a_t \mid s_t)$
4: Execute a_t, receive reward $r_t = r(s_t, a_t)$ and next state s_{t+1}
5: Update internal state or policy (optional, depending on algorithm)
6: **end for**

Over time, the agent adjusts its behavior to improve long-term performance. The formal objective is to maximize the expected discounted return:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right],$$

where r_t is the reward received at time t, and γ discounts future rewards. Alternatively, this can be written in terms of trajectories $\tau = (s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1})$:

$$\hat{\pi} = \arg\max_{\pi \in \Pi} \mathbb{E}_{\tau \sim \rho^{\pi}}\left[R(\tau)\right], \quad \text{where} \quad R(\tau) := \sum_{t=0}^{T-1} \gamma^t r_t.$$

To ease notation, we assume a finite horizon T and that the state space implicitly encodes the timestep. This allows us to suppress explicit time dependence in the policy. Thus, $R(\tau)$ denotes the total (possibly discounted) reward over a trajectory.

In environments with a known and finite state-action space, dynamic programming methods such as policy iteration or value iteration can be used to compute the optimal policy. However, in complex or unknown environments, exact planning becomes intractable. In such cases, we begin with a simple (often random) policy and iteratively improve it through trial and error by interacting with the environment.

In the next sections, we introduce parameterized policies and show how reinforcement learning methods can optimize them using gradients. This leads to policy gradient algorithms, including the Proximal Policy Optimization (PPO) method used in our application.

Policy-Based Methods

In small, discrete environments, policies can be represented as finite lookup tables that assign an action to each state. However, in high-dimensional settings or when state and action spaces are both continuous — such as controlling a robot arm or managing a financial portfolio — this approach becomes infeasible. Instead, we turn to *parameterized policies*, which define the policy as a function $\pi_\theta(a \mid s)$ with parameters θ.

In our case, the policy is represented by a neural network that outputs an action distribution given the current state. The idea is to find the parameters θ that yield a policy maximizing expected cumulative reward:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right].$$

To optimize $J(\theta)$, we use gradient descent:

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta),$$

where α is the learning rate. The key insight is that this gradient can be estimated directly from samples collected by interacting with the environment, using the *Policy Gradient Theorem*:

$$\nabla_\theta J(\theta) \propto \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot A_t \right].$$

Q = where does this come from ? more intuition ...

Here, A_t is called the *advantage function*, which quantifies how much better the action a_t was compared to a baseline (e.g., the average action in that state). Using the advantage helps reduce the variance of gradient estimates and improves learning stability.

Policy gradient methods have enabled impressive real-world applications, including AlphaGo, OpenAI Five, and large language models, where actions are sequences of tokens. These methods are particularly well suited for problems with continuous or stochastic action spaces, and where value-based methods like Q-learning are difficult to apply directly.

Q = why ? are value-based methods like Q-learning difficult to apply directly

In the next section, we introduce the Proximal Policy Optimization (PPO) algorithm from Schulman et al. (2017), a practical and robust policy gradient method that incorporates safeguards against unstable updates.

Proximal Policy Optimization

While policy gradient methods are conceptually simple and effective, they can suffer from instability in practice. Large updates to the policy parameters may lead to drastic behavior changes, which could degrade performance. This motivates a more conservative update strategy — one that makes consistent, small improvements without destabilizing learning.

A principled way to enforce small policy changes is to restrict the updated policy to stay within a "trust region" around the old one. This can be quantified using the Kullback–Leibler (KL) divergence between the old and new policies. The Trust Region Policy Optimization (TRPO) algorithm formalizes this idea using constrained optimization, but it involves second-order derivatives and complex line searches.

Proximal Policy Optimization (PPO) simplifies this idea by introducing a clipped surrogate objective that approximates a trust region using only first-order methods. Let π_θ be the current policy and $\pi_{\theta_{\text{old}}}$ the previous one. Define the probability ratio:

$$r_t(\theta) := \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}.$$

The PPO objective is:

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

TODO : add intuition on this formula ? why exactly this objective function ?

where ϵ is a small hyperparameter (typically around 0.1–0.3).

The clipping mechanism works as follows:

- If $r_t(\theta)$ is close to 1 (i.e., the new policy is similar to the old one), the update proceeds normally.
- If $r_t(\theta)$ deviates too much from 1, the update is clipped to prevent the policy from changing too aggressively.

This ensures that good actions are still reinforced, but not in an overly confident way, and bad actions are not overly punished if the update would have changed their probability too much. The result is a training process that is both stable and sample-efficient, without requiring expensive second-order methods.

Q = what does "good actions are reinforced" mean here ?

PPO has become one of the most widely used reinforcement learning algorithms due to its balance between simplicity, theoretical motivation, and empirical robustness.

Training PPO in Practice

The PPO training process alternates between collecting data from the environment and updating the policy using that data. This makes PPO an example of an *on-policy* algorithm: it learns from trajectories generated by the current version of the policy, and must continuously re-sample data as the policy changes.

The full PPO training loop typically follows these steps:

Algorithm 2: Proximal Policy Optimization (PPO) Training Loop

1: **for** each iteration **do**
2: Collect a batch of experience using the current policy π_θ
3: Estimate the advantage A_t using a learned value function $V_\phi(s_t)$
4: Perform stochastic gradient ascent on $L^{\text{CLIP}}(\theta)$ using mini-batches
5: Optionally update V_ϕ to minimize prediction error in returns
6: **end for**

A popular technique for computing the advantage A_t is *Generalized Advantage Estimation (GAE)*, which balances bias and variance using a discount factor λ:

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \dots,$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the temporal-difference (TD) error. GAE provides smoother and more stable advantage signals, improving training efficiency.

In our application, the PPO agent is trained to allocate a financial portfolio using historical market data. Unlike traditional benchmarks based on simple returns, we optimize a risk-adjusted objective that encourages smooth and consistent growth.

Specifically, the reward signal is based on the *Differential Sharpe Ratio* (DSR), an online proxy for the Sharpe ratio that is compatible with gradient-based learning.

TODO : rewrite the following paragraph

The definition of the reward functions suitable for financial trading and portfolio optimization applications is not trivial, due to the fact that objectives such as the Sharpe or Sortino ratios are computed using moments estimated from samples over time, whereas RL is formulated as a sequential decision making problem and thus requires reward functions that reflect the sequential (step-by-step) nature of the process, as well as the fact that in RL the returns are additive on (discounted) rewards, whereas the Sharpe or Sortino ratios are not additive functions over time. To account for this, Moody et al. (1998) proposed the use of a Differential Sharpe ratio.

END TODO.

The classical Sharpe ratio (eq. 2.1) is non-stationary and non-differentiable over time.

Q = why ?

The Differential Sharpe Ratio instead uses exponentially weighted moving averages to approximate the numerator and denominator:

$$\mathrm{DSR}_t \propto \frac{\Delta\mu_t}{\sigma_t}, \tag{2.6}$$

where μ_t and σ_t are exponentially smoothed estimates of mean and standard deviation of returns, and $\Delta\mu_t$ is the change in mean. This formulation allows the agent to be trained online using mini-batches of data.

Such online updates make PPO well suited to financial environments, where reward signals evolve with market conditions and long-term performance depends on cumulative returns and volatility management.

Ultimately, PPO learns a dynamic policy that adjusts portfolio allocations based on recent market features — a learned alternative to hand-crafted strategies or classical optimization routines.

3 Data & Methodology

3.1 Data

We evaluate all strategies on two datasets: U.S. equity sector data via SPDR S&P 500 ETFs, and international equity data via MSCI country indices. Both datasets consist of daily adjusted closing prices from January 2006 to December 2021 and are sourced from Yahoo Finance.

S&P 500 Sector ETFs

The S&P 500 index is the primary benchmark for U.S. equity markets, covering over 80% of total market capitalization. Its 11 sector components—defined by the Global Industry Classification Standard (GICS) S&P Dow Jones Indices and MSCI (2023)—enable more interpretable and targeted allocation strategies. We use the corresponding SPDR sector ETFs as our investable universe.

Table 1 lists all 11 ETFs and their sectors. Notably, `XLK` represents Information Technology (e.g., Apple, Microsoft), `XLF` covers Financials (e.g., JPMorgan, Bank of America), and `XLE` tracks the Energy sector (e.g., ExxonMobil, Chevron). Figure 1 shows cumulative returns from 2006 to 2021, highlighting substantial dispersion across sectors and motivating dynamic allocation strategies.

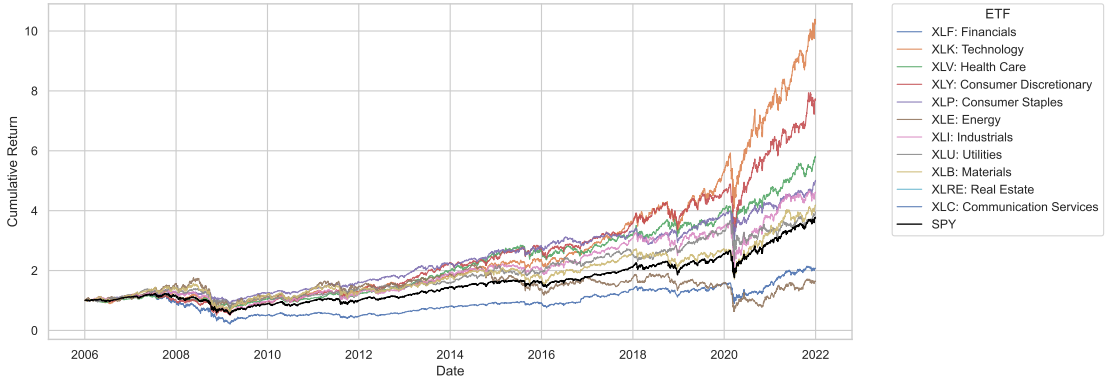

Figure 1: Cumulative return of the S&P 500 and its 11 sector indices, 2006–2021.

MSCI Country Indices

To evaluate whether the findings generalize beyond the U.S. market, we include a second dataset consisting of daily prices from MSCI country indices. These indices track the performance of large- and mid-cap equities in developed markets, as defined by MSCI.

We take inspiration from DeMiguel et al. (2007), who list a similar cross-country dataset as a standard benchmark for evaluating portfolio selection strategies. Following their setup, we focus

Ticker	Sector Name	Inception Year
XLC	Communication Services	2018
XLY	Consumer Discretionary	1998
XLP	Consumer Staples	1998
XLE	Energy	1998
XLF	Financials	1998
XLV	Health Care	1998
XLI	Industrials	1998
XLB	Materials	1998
XLRE	Real Estate	2015
XLK	Information Technology	1998
XLU	Utilities	1998

Table 1: SPDR Sector ETFs used in this study.

on eight major developed economies: Canada, France, Germany, Italy, Japan, Switzerland, the United Kingdom, and the United States. Table 2 lists the countries along with the corresponding ETF tickers used in our implementation.

All price series are aligned to a common daily frequency and span the same time period as the U.S. sector dataset (2006–2021).

Country	ETF Ticker
Canada	`EWC`
France	`EWQ`
Germany	`EWG`
Italy	`EWI`
Japan	`EWJ`
Switzerland	`EWL`
United Kingdom	`EWU`
United States	`SPY`

Table 2: MSCI country indices used in the international dataset.

Preprocessing & Volatility Features for DRL

For each asset i, daily log returns are computed as

$$r_t^{(i)} = \log\left(\frac{P_t^{(i)}}{P_{t-1}^{(i)}}\right), \tag{3.1}$$

where $P_t^{(i)}$ is the adjusted close of asset i on day t. Log returns are preferred due to their time-additive and approximately Gaussian properties.

The DRL agent receives three additional market indicators based on `SPY` or `MSCI` returns and the VIX index:

- vol20: 20-day rolling volatility

- vol_ratio: vol20 divided by vol60
- VIX: CBOE Volatility Index CBOE 2024 (daily close)

All features are standardized using an expanding window mean and standard deviation to avoid lookahead bias.

3.2 Code Implementation

The project is organized as follows with the full repository available at: https://github.com/hikotei/2025-rl-portfolio-opt

Project Directory

```
2025-rl-portfolio-opt/

 utils/
    config.py           # Configuration dataclasses
    drl_agent.py        # PPO-based DRL agent class
    drl_train.py        # DRL training pipeline helpers
    portfolio.py        # Unified Portfolio logic
    portfolio_env.py    # Custom Gym environment
    mvo_strategy.py     # Mean-Variance Optimization

 notebooks/
    get_data.ipynb       # Load and preprocess data
    drl_train.ipynb      # Train PPO agents
    drl_evaluate.ipynb   # Evaluate trained agents
    mvo_backtest.ipynb   # MVO backtest

 models/       # Saved PPO agents
 results/      # Backtest outputs (returns, weights)
 plots/        # Evaluation plots and charts
 logs/         # TensorBoard logs
 README.md
```

3.2.1 The Portfolio Class

At the core of the implementation is a unified `Portfolio` class (`portfolio.py`), which abstracts all logic related to portfolio valuation, rebalancing, and feature generation. This class is used by both the MVO strategy and the DRL agent, ensuring consistency and modularity across the entire codebase.

The `Portfolio` object maintains the current and past allocation weights, computes integer share quantities (rounded down), and simulates rebalancing with exact cash handling. It also provides standardized access to rolling return windows and exogenous features, such as VIX and volatilities, for use by both methods.

In the MVO pipeline, a dedicated `MvoStrategy` object queries the portfolio for return histories, performs daily optimization, and updates allocations. In the DRL pipeline, the `Portfolio` is embedded within a custom Gym environment, `PortfolioEnv` (`portfolio_env.py`), which wraps market simulation, reward computation (Differential Sharpe Ratio), and observation construction. The environment receives actions from the agent, applies them via the portfolio interface, and returns the resulting reward and state.

This modular design ensures a shared execution logic and avoids code duplication, while enabling strategy-specific logic to be implemented externally.

3.2.2 Mean-Variance Optimization Strategy

The MVO strategy allocates portfolio weights by maximizing the Sharpe ratio under long-only and fully invested constraints. Given a portfolio weight vector $\mathbf{w}$, estimated expected returns $\boldsymbol{\mu}$, and covariance matrix Σ, the optimization problem is

$$\begin{aligned} \max_{\mathbf{w}} \quad & \frac{\mathbf{w}^\top \boldsymbol{\mu}}{\sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}} \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i = 1, \quad w_i \geq 0. \end{aligned} \tag{3.2}$$

Both $\boldsymbol{\mu}$ and Σ are estimated using a 60-day rolling window of log returns from the `Portfolio` object. The mean is computed as the simple average, while the covariance matrix is regularized via Ledoit-Wolf shrinkage to improve stability in the presence of noise.

Optimization is performed manually using `scipy.optimize.minimize` by default, with modular support for alternative solvers such as `PyPortfolioOpt` or `skfolio`.

Algorithm 3: MVO Daily Allocation Routine

1: **for** each trading day t in the backtest period **do**
2: Extract 60-day log returns
3: Estimate $\boldsymbol{\mu}$ and Σ (with shrinkage)
4: Solve Sharpe ratio maximization under constraints
5: Apply allocation using `Portfolio.rebalance()`
6: Record returns and update state
7: **end for**

This daily rebalancing loop is repeated over each test window. The strategy uses the same asset universe, data, lookback window, and execution pipeline as the DRL agent to ensure comparability. Performance metrics are computed identically using shared valuation logic.

3.2.3 RL Environment and Agent

The DRL agent operates in a custom OpenAI Gym environment, `PortfolioEnv` (`portfolio_env.py`), which simulates daily portfolio rebalancing via historical market replay. The environment uses the `Portfolio` class to manage trades, compute portfolio value, and track previous allocations. It simulates n risky assets and one cash asset, with all trades assumed to execute instantaneously at end-of-day (EOD) close prices.

Observation / State. The agent's observation at time t is a $(n+1) \times (T+1)$ matrix, where $T = 60$ denotes the lookback period in trading days. Each row contains the log returns of one asset over the past T days and its current portfolio weight w_i. The final row includes three market-wide indicators: 20-day rolling volatility (`vol20`), the volatility ratio `vol20/vol60`, and the VIX index at time t. All features are standardized using an expanding mean and standard deviation computed up to time $t-1$, ensuring that no future information is leaked (i.e., avoiding lookahead bias).

$$S_t = \begin{bmatrix} w_1 & r_{1,t-1} & r_{1,t-2} & \cdots & r_{1,t-T+1} \\ w_2 & r_{2,t-1} & r_{2,t-2} & \cdots & r_{2,t-T+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_n & r_{n,t-1} & r_{n,t-2} & \cdots & r_{n,t-T+1} \\ w_c & \text{vol}_{20} & \frac{\text{vol}_{20}}{\text{vol}_{60}} & \text{VIX}_{\boldsymbol{t}} & \cdots \end{bmatrix} \tag{3.3}$$

Note that the weights $\mathbf{w}$ in the state matrix correspond to the realized portfolio allocation entering timestep t, which may differ slightly from the action chosen at the previous timestep due to discretization or rounding in the environment implementation.

Action. The agent outputs a continuous portfolio allocation vector $\mathbf{w}_t \in [0,1]^n$, representing the proportion of total capital allocated to each asset. The vector is mapped to the probability simplex via a softmax function to ensure all entries are non-negative and sum to one. The environment assumes long-only, fully invested portfolios with no leverage, shorting, transaction costs, or slippage. This constraint setup mirrors the MVO baseline for fair evaluation.

Reward. The reward at each timestep is the Differential Sharpe Ratio (DSR), an online, differentiable proxy for risk-adjusted return introduced by **?**. It enables smooth gradient-based optimization of the Sharpe Ratio, aligning training objectives with common financial performance metrics.

Algorithm 4: PPO-Based DRL Portfolio Optimization Loop

```
Load market data (e.g., prices, VIX)
Initialize environment PortfolioEnv with data
for total training steps do
    Extract state: (n+1) × (T+1) matrix of returns and features
    Pass state to PPO agent to obtain action (portfolio weights)
    Apply action via Portfolio.rebalance()
    Compute reward using Differential Sharpe Ratio (DSR)
    Advance environment to next timestep and return next state
    Update PPO agent using collected (state, action, reward, next state) tuples
end for
```

3.3 Experimental Setup

We adopt the same experimental framework to compare Deep Reinforcement Learning (DRL) and Mean-Variance Optimization (MVO) as in Sood et al. (2023), but adapts some ambiguous components to ensure reproducibility and include two additional experiments for further analyis.

1 = Paper Setting

1a) vanilla with policy seeding 1b) fully random policy initialization each time

2 = MSCI data with full random setting

Assumptions

To ensure comparability between strategies, we make the following simplifying assumptions:

- No transaction costs or slippage: Trading is frictionless and instantaneous.
- Zero risk-free rate: The Sharpe ratio is computed under the assumption $r_f = 0$.
- Long-only, fully invested portfolios: Portfolio weights satisfy $\sum w_i = 1$, with $w_i \geq 0$ for all i, including a dedicated cash component.
- Integer share rounding: Rebalancing converts weights into whole-share quantities and adjusts residuals to cash, as implemented in the `Portfolio` class.

Train Test Split / Sliding Window Approach

The training follows a sliding window regime. Each cycle consists of five years for training, one year for validation, and one year for testing. For each window, five agents are trained with different random seeds. The best-performing model on the validation set is selected for deterministic evaluation on the test set.

To simulate real-world generalization and adaptivity, we employ a sliding-window backtesting scheme. Each evaluation window spans 7 years: 5 years for training, 1 year for validation (used only for DRL), and 1 year for out-of-sample testing. This results in 10 evaluation periods from 2012 to 2021. Each test starts with a $100,000 cash allocation.

Multiple DRL agents (we select 5 as in Sood et al. (2023)) are trained on the training period, the best one is selected based on validation performance, and evaluated deterministically on the test year.

The MVO strategy doesnt need a training period since it only re-estimates means and covariances daily using a 60-day lookback window. Therefore we can simply run it over the entire testing range of 2012 to 2021 and evaluate return metrics etc. on an annual basis to compare with the DRL agents that are evaluated on one backtesting year only.

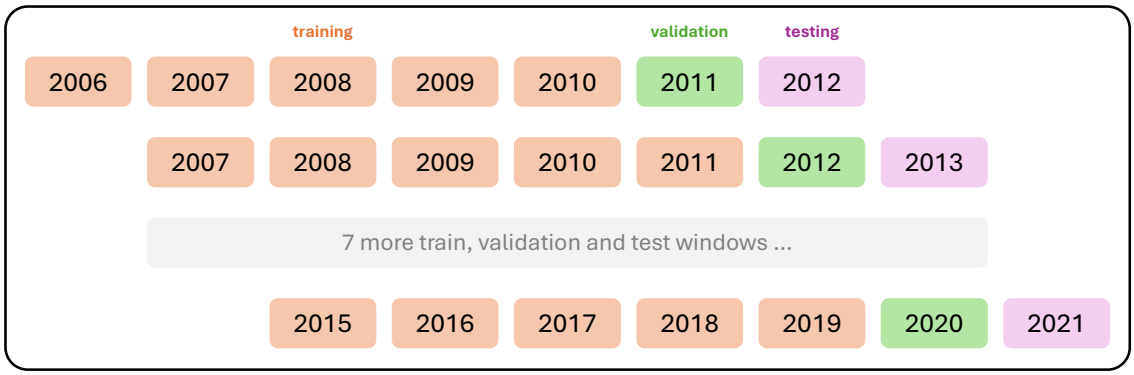

Figure 2: Illustration of the sliding window approach. Each window consists of 5 years for training, 1 year for validation (DRL only), and 1 year for testing.

DRL Training

Training is performed using Proximal Policy Optimization (PPO) via the `Stable-Baselines3` library (Raffin et al. 2021). The agent interacts with 10 parallel environments (`SubprocVecEnv`) to improve sample efficiency and reduce correlation between experiences. Each training run lasts

for 7.5 million timesteps, which corresponds to approximately 600 episodes per environment:

$$7.5\,\mathrm{M} \approx 252 \times 5 \times 10 \times 600 \text{ episodes}$$

Experience is collected using a rollout buffer of $n_{\text{steps}} = 252 \times 3 \times 10 = 7560$ timesteps across all environments. PPO updates are performed using mini-batches of size 1260 over 16 epochs. The learning rate decays linearly from $3 \cdot 10^{-4}$ to $1 \cdot 10^{-5}$ over the course of training. The initial log standard deviation is set to -1.

TODO : what exactly is rollout buffer ? explain that 252 is days per year and 10 is $\mathrm{n}_e nvbutwhytimes3?TODO$: $explainwhyminibatchis1260 = 252 * 5...why5?$

The policy network consists of two fully connected layers with 64 units each and `tanh` activations. It outputs a Gaussian distribution over portfolio weights, which are mapped to the probability simplex via a softmax function to ensure long-only, fully invested portfolios.

TODO : short intro to introduce the PPO architecture

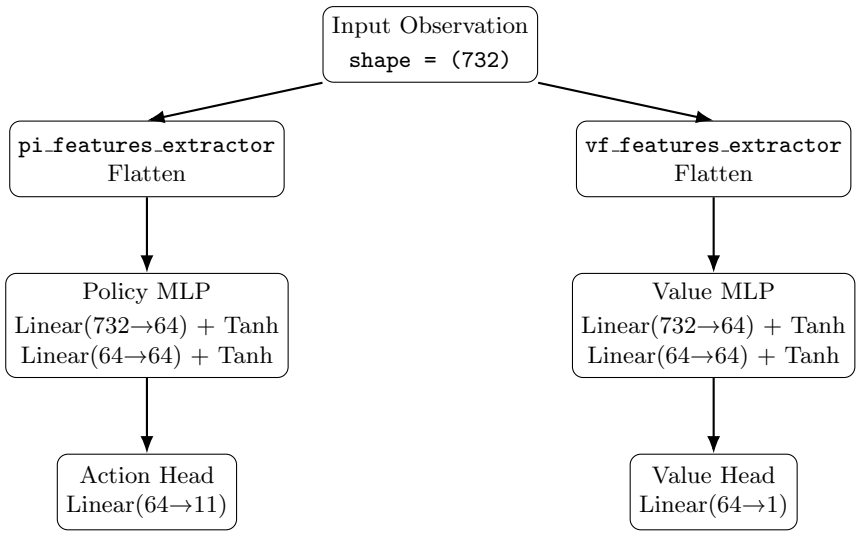

Figure 3: test

To replicate the training regime described in Sood et al. (2023), we implemented a policy warm-start initialization strategy across sliding windows. While the original paper states that "the [best] agent [of the previous window] is used as a seed policy for the next group of 5 agents," it does not specify how this is implemented.

If we were to pass the seed of the best agent from the first window to all agents of the seconf window, this would lead to identical seed initializations for all following runs. On the other hand, reusing the exact policy network weights of the previous best agent makes all agents converge to the same optimum since they are trained on the same data. Both reasons lead to identical performance of all 5 agents and therefore do not make sense.

To address this, we adopted a modified warm-start procedure: at each window, five PPO agents are initialized with the policy of the best-performing agent from the previous window, but with 5% Gaussian noise added to the weights. This allows for knowledge transfer while maintaining diversity and exploration. Empirically, this strategy produced more stable and robust generalization performance on the test set.

The full set of PPO hyperparameters used during training is summarized in Table 3.

Parameter	Value
training_timesteps	7.5M
n_envs	10
n_steps	756
batch_size	1260
n_epochs	16
gamma	0.9
gae_lambda	0.9
clip_range	0.25
learning_rate	3e-4 annealed to 1e-5
log_std_init	-1
architecture	[64, 64], `tanh` activations

Table 3: PPO hyperparameters used for training

Evaluation Metrics

Performance is evaluated using a series of return- and risk-based statistics computed on daily returns. Table 2 in Sood et al. (2023) shows a tabular comparison of the average values of these between DRL and MVO which we replicate in the results section.

- **Return-based:** cumulative and annualized returns
- **Risk-based:** annualized volatility, maximum drawdown, daily value-at-risk (VaR)
- **Risk-adjusted:** Sharpe, Sortino, Calmar, and Omega ratios
- **Distributional:** skewness, kurtosis, tail ratio
- **Stability:** average annual turnover

In Figure 2 of Sood et al. (2023), line plots are presented for Sharpe Ratio, Maximum Drawdown, and Average Daily Change in Portfolio Weights. However, the latter is not formally defined, while the term *Turnover* appears repeatedly throughout the paper. To ensure clarity, we choose to use annual turnover as our measure of trading intensity and define it according to the Securities & Commission (1998) (SEC) standard and calculate the average over all testing years as final metric.

$$\text{Annual Turnover} = \frac{\min(\text{Total Purchases}, \text{Total Sales})}{\text{Average Portfolio Value}} \tag{3.4}$$

In Figures 3 (and 4) of Sood et al. (2023) they show a) a heatmap of monthly returns across all testing years, b) a horizontal bar plot of annual returns for all years, and c) a histogram of monthly returns across all years for the DRL agent (and MVO with lookback of 60 days).

All metrics are computed using internal utilities of the `Portfolio` class, based on the recorded portfolio weights and value history.

Hardware and Runtime

All experiments were conducted on a personal MacBook laptop (Apple M3 Max, 64 GB RAM). Training all 10 windows with 5 agents per window (50 agents total) under the paper settings required approximately 16 hours.

4 Results

4.1 Key Hypotheses

We aim to replicate the results presented in Sood et al. (2023), using the same dataset of daily S&P 500 sector index returns from 2010 to 2022. We then extend the evaluation to a different dataset consisting of daily returns from MSCI country indices.

Before comparing the performance of reinforcement learning (RL) and classical optimization methods, we critically examine the choice of baseline in Sood et al. (2023). Their study evaluates a DRL agent trained with Proximal Policy Optimization (PPO) and compares it exclusively against a Mean-Variance Optimization (MVO) strategy with a 60-day lookback horizon. However, no justification or citation is provided for selecting this specific configuration, and no sensitivity analysis is performed across other lookback windows.

The following hypotheses structure our empirical evaluation:

- **H1**: MVO performance is highly sensitive to the choice of lookback window, and the 60-day setting used by Sood et al. (2023) represents one of the weakest configurations.
- **H2**: When replicated with the same data and hyperparameters, the DRL agent trained with policy seeding achieves similar performance to the results in Sood et al. (2023).
- **H3**: The policy seeding scheme introduces structural unfairness, as each successive DRL agent inherits weights trained on data from previous windows, effectively increasing its training horizon.
- **H4**: Naive baselines such as the equally weighted portfolio (1/N) and a passive buy-and-hold strategy perform surprisingly well, especially on S&P 500 data. This raises questions about the appropriateness of MVO(60) as the sole classical benchmark.

We begin with a detailed analysis of MVO lookback sensitivity before evaluating DRL performance under both the paper's setup and a more realistic random initialization regime. All methods are compared on both the S&P 500 and MSCI datasets using consistent metrics.

4.2 MVO Lookback Analysis

Before comparing DRL to MVO, we first examine whether MVO with a 60-day lookback—used as the sole benchmark in Sood et al. (2023)—constitutes a representative baseline. Since the performance of MVO depends heavily on the choice of lookback horizon for estimating means and covariances, we conduct a sensitivity analysis across a wide range of lookback windows.

Figure 4 and Figure 5 show the cumulative portfolio value for different MVO lookback windows on the S&P 500 and MSCI datasets, respectively. Notably, the 60-day lookback performs worse than both a naive equally weighted (1/N) portfolio and a passive buy-and-hold strategy on the S&P 500 data.

To better understand this behavior, Figure 6 and Figure 7 present three performance met-

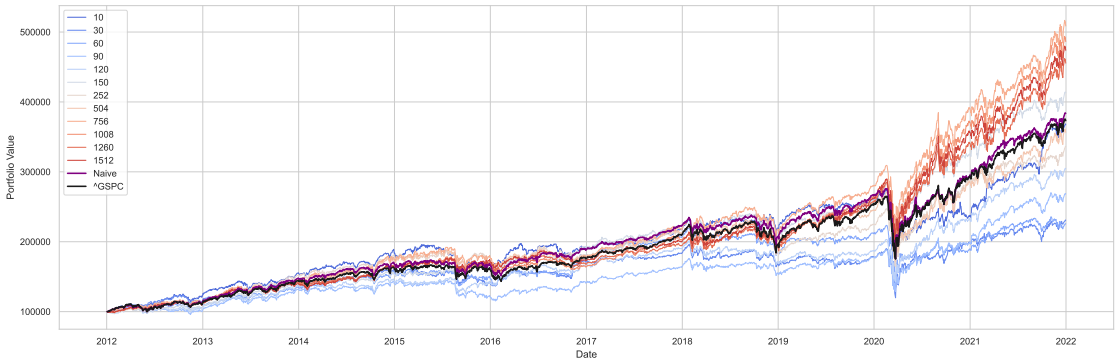

Figure 4: Cumulative portfolio value of MVO with varying lookback windows on S&P 500 data (2012–2021).

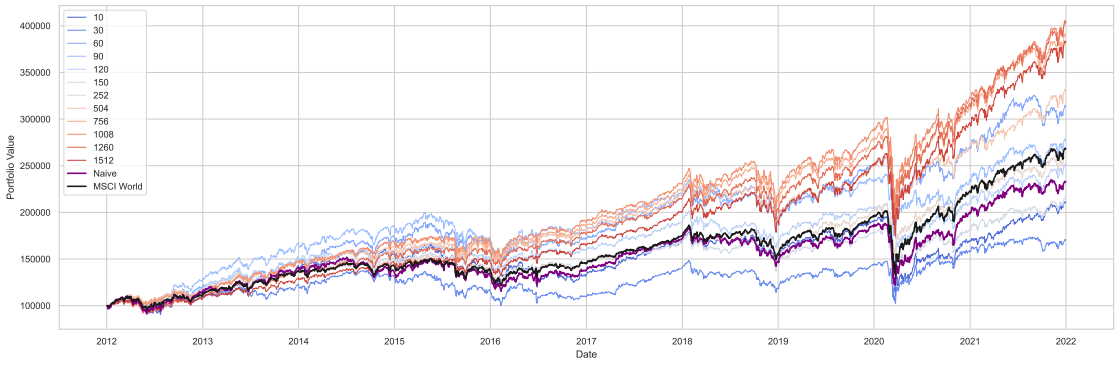

Figure 5: Cumulative portfolio value of MVO with varying lookback windows on MSCI data (2012–2021).

rics—Sharpe ratio, maximum drawdown, and average annual turnover—as a function of the lookback window.

On both datasets, Sharpe ratio generally increases with the lookback window up to around 756 days (roughly three years), after which performance tends to decline. Maximum drawdown initially spikes and then decays slowly, suggesting that short-term estimates may lead to unstable or overreactive portfolios. Turnover decays exponentially with lookback length, likely due to more stable estimates of covariance and expected return, resulting in less frequent rebalancing.

Interestingly, for lookback horizons below 90 days, MVO often underperforms simple benchmarks and occasionally fails entirely due to negative returns across all assets in the estimation window—leading to infeasible or unstable optimization results.

This analysis confirms that MVO with a 60-day window is one of the weakest-performing configurations in terms of both return and risk. Consequently, using it as the sole classical benchmark—as done in Sood et al. (2023)—potentially exaggerates the advantages of DRL.

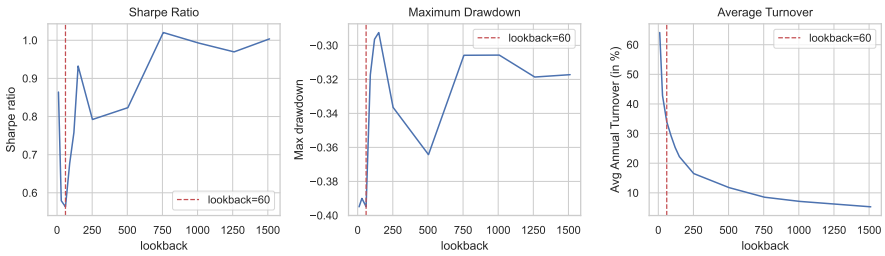

Figure 6: Sharpe ratio, max drawdown, and turnover of MVO on S&P 500 data by lookback window (2012–2021).

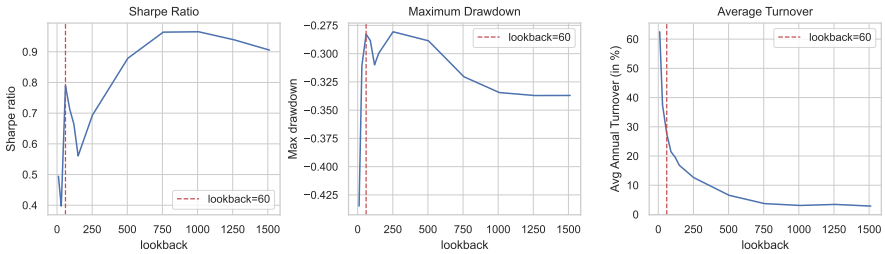

Figure 7: Sharpe ratio, max drawdown, and turnover of MVO on MSCI data by lookback window (2012–2021).

4.3 S&P500 Sector Indices

We now evaluate the performance of different portfolio strategies on the S&P500 sector data. We first compare MVO with short and long lookback horizons. Then we assess DRL performance under two training settings: the original policy seeding approach described in Sood et al. (2023), and a fairer variant using fully random initialization. Finally, we summarize and benchmark all strategies against a naive equally weighted (1/N) portfolio and a passive buy-and-hold strategy.

4.3.1 MVO Results

Figure 8 replicates the MVO performance with a 60-day lookback window as shown in Figure 4 of Sood et al. (2023). This configuration underperforms both DRL and naive benchmarks across most years. In contrast, Figure 9 shows that a 756-day lookback substantially improves performance, achieving higher average returns and smoother monthly outcomes.

4.3.2 RL Results

We now evaluate DRL performance under two training regimes. Figure 10 shows results using the policy warm-start approach described in Sood et al. (2023), where each new agent is initialized with weights from the best agent of the previous window. This setting yields high returns and consistent performance, with only minor losses in 2018 and a strong recovery in 2020 following the initial COVID drawdown.

To test a more realistic scenario, we also train agents from scratch using random policy initialization in each window. Figure 11 shows that while performance remains strong, returns in 2015,

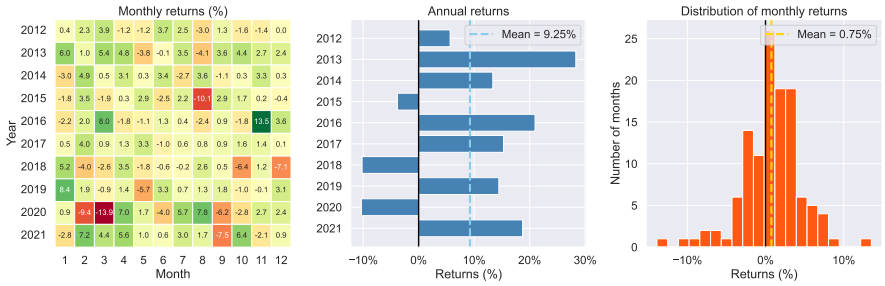

Figure 8: Detailed MVO performance on S&P500 data (2012–2021) using a 60-day lookback.

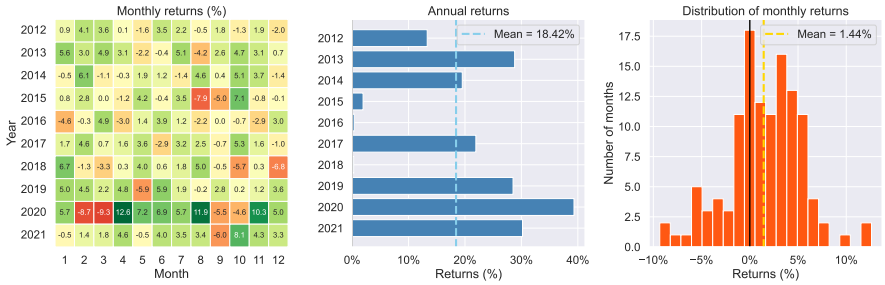

Figure 9: Detailed MVO performance on S&P500 data (2012–2021) using a 756-day lookback.

2018, and 2020 drop closer to zero or slightly negative—mirroring the behavior of MVO in those years. Nevertheless, the overall Sharpe and Sortino ratios remain high, suggesting robustness even without seeding.

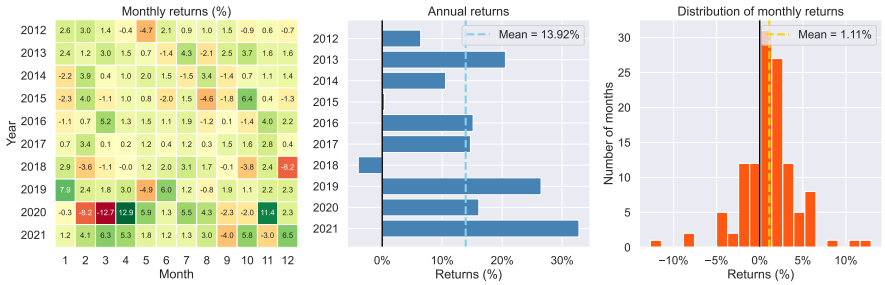

Figure 10: DRL performance on S&P500 data (2012–2021) using policy warm-start.

4.3.3 Summary

Table 4 compares the performance of all strategies across a comprehensive set of metrics. DRL outperforms MVO(60) on all risk-adjusted metrics, with Sharpe ratios exceeding 1.3 in both the paper and random initialization settings. Notably, the DRL agent with random initialization performs nearly as well as the warm-start version, suggesting that the advantage gained from seeding may be smaller than expected.

MVO with a 756-day lookback achieves the highest average return (18.6%), outperforming DRL

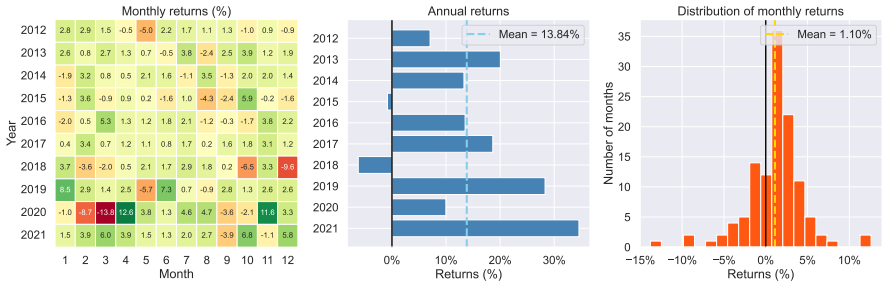

Figure 11: DRL performance on S&P500 data (2012–2021) with random policy initialization.

and naive baselines in absolute terms. However, this comes at the cost of slightly higher volatility. Both naive and buy-and-hold strategies perform strongly on the S&P500 data, with Sharpe ratios above 1.2 and minimal turnover.

	DRL paper	DRL random	MVO 60	MVO 756	Naive 1/N	Buy & Hold
Annual Returns	0.1405	0.1398	0.0941	0.1856	0.1508	0.1464
Cumulative Returns	0.1392	0.1383	0.0936	0.1848	0.1501	0.1455
Stability	0.8933	0.8904	0.8747	0.8705	0.8801	0.8749
Annual Volatility	0.1237	0.1276	0.1484	0.1539	0.1405	0.1469
Max Drawdown	-0.0955	-0.0989	-0.1200	-0.1101	-0.1092	-0.1133
Daily VaR (95%)	-0.0122	-0.0129	-0.0157	-0.0155	-0.0138	-0.0149
Tail Ratio	1.0052	0.9918	0.9333	1.0392	1.0413	1.0310
Sharpe Ratio	1.3410	1.3595	0.9364	1.3232	1.3356	1.2173
Calmar Ratio	2.9300	3.3879	1.8984	2.3495	2.9573	2.5227
Omega Ratio	1.2746	1.2852	1.1825	1.2698	1.2737	1.2511
Sortino Ratio	1.8423	1.8990	1.3545	1.8071	1.8436	1.6445
Skew	-0.3828	-0.3605	-0.2920	-0.3572	-0.3732	-0.3735
Kurtosis	2.5229	2.3493	2.1371	2.3428	2.3876	2.5944
Turnover	31.5686	49.9076	33.9906	8.5254	0.6169	NaN

Table 4: Performance metrics for all strategies on S&P500 data (2012–2021).

4.4 MSCI Country Indices

We now repeat the evaluation on a broader international dataset consisting of MSCI country indices. As before, we first compare MVO performance with short and long lookback horizons, then assess the robustness of DRL under random initialization. Finally, we summarize all results relative to baseline strategies.

4.4.1 MVO Results

Figure 12 shows the performance of the MVO strategy using a 60-day lookback on the MSCI dataset. While this configuration underperforms long-horizon MVO, it performs slightly better here than on the S&P500 data. Figure 13 displays results for a 756-day lookback, which again improves average returns, smooths monthly returns, and reduces downside risk.

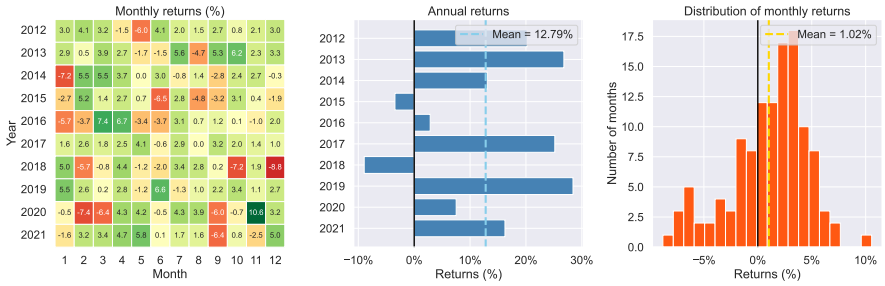

Figure 12: Detailed MVO performance on MSCI data (2012–2021) using a 60-day lookback.

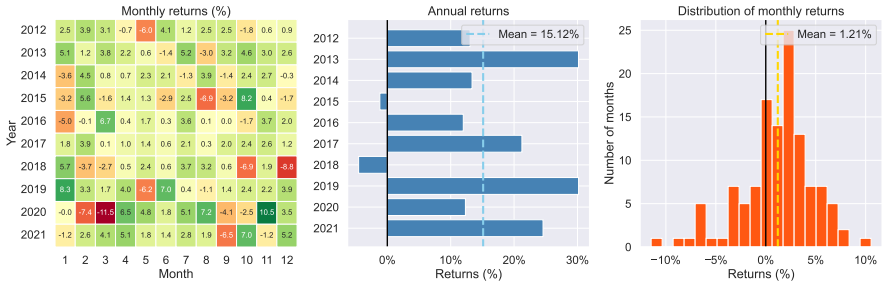

Figure 13: Detailed MVO performance on MSCI data (2012–2021) using a 756-day lookback.

4.4.2 RL Results

We evaluate DRL performance on the MSCI dataset using the more realistic setting of random policy initialization. As shown in Figure 14, the DRL agent performs well overall, but exhibits negative or flat returns in several individual years (notably 2014, 2015, and 2018). These results reflect the more volatile and heterogeneous structure of international equity markets, which may present additional challenges for reinforcement learning.

Figure 14: DRL performance on MSCI data (2012–2021) with random policy initialization.

4.4.3 Summary

Table 5 compares all strategies on the MSCI dataset. Results are broadly consistent with the S&P500 case: MVO with a 60-day lookback underperforms across most metrics, while a longer

756-day horizon improves results significantly. Interestingly, the relative performance gap between MVO and DRL is smaller here, with MVO(756) achieving the highest annual return.

DRL maintains strong risk-adjusted metrics across the board, particularly in Sharpe, Sortino, and Omega ratios. The naive 1/N and buy-and-hold strategies remain competitive, but less dominant than in the S&P500 setting. This suggests that diversification across countries reduces the edge of simple allocation heuristics and increases the benefit of data-driven approaches like MVO and DRL.

	DRL paper	DRL random	MVO 60	MVO 756	Naive 1/N	Buy & Hold
Annual Returns	0.1405	0.1397	0.0941	0.1856	0.1508	0.1464
Cumulative Returns	0.1392	0.1384	0.0936	0.1848	0.1501	0.1455
Stability	0.8933	0.8903	0.8747	0.8705	0.8801	0.8749
Annual Volatility	0.1237	0.1276	0.1484	0.1539	0.1405	0.1469
Max Drawdown	-0.0955	-0.0987	-0.1200	-0.1101	-0.1092	-0.1133
Daily VaR (95%)	-0.0122	-0.0129	-0.0157	-0.0155	-0.0138	-0.0149
Tail Ratio	1.0052	0.9906	0.9333	1.0392	1.0413	1.0310
Sharpe Ratio	1.3410	1.3595	0.9364	1.3232	1.3356	1.2173
Calmar Ratio	2.9300	3.3853	1.8984	2.3495	2.9573	2.5227
Omega Ratio	1.2746	1.2850	1.1825	1.2698	1.2737	1.2511
Sortino Ratio	1.8423	1.8985	1.3545	1.8071	1.8436	1.6445
Skew	-0.3828	-0.3618	-0.2920	-0.3572	-0.3732	-0.3735
Kurtosis	2.5229	2.3402	2.1371	2.3428	2.3876	2.5944
Turnover	31.5686	49.9076	33.9906	8.5254	0.6169	NaN

Table 5: Performance metrics for all strategies on MSCI data (2012–2021).

5 Conclusion

5.1 Summary of Results

This study set out to replicate and critically evaluate the results of Sood et al. (2023), who propose a deep reinforcement learning (DRL) approach to portfolio optimization using Proximal Policy Optimization (PPO) with a differential Sharpe ratio reward. Their method is compared against a classical mean-variance optimization (MVO) strategy with a fixed 60-day lookback window. Using daily return data from S&P500 sector ETFs and MSCI country indices, we conducted a systematic comparison across multiple model configurations and baselines.

Our findings largely confirm the conclusions of Sood et al. (2023) within their original experimental setup. When DRL agents are trained with policy seeding, they achieve consistently strong performance across both datasets, clearly outperforming MVO with a 60-day lookback on return- and risk-adjusted metrics.

However, we find that this comparison is heavily dependent on the chosen benchmark. MVO with a 60-day lookback performs worse than simple baseline strategies such as naive 1/N or buy-and-hold on the S&P500 data. Our lookback sensitivity analysis shows that longer MVO horizons—particularly between 252 and 756 trading days—significantly improve both return and stability. In some cases, MVO(756) even outperforms DRL in terms of annual return, especially on the MSCI dataset.

We also identify a methodological concern with the policy seeding strategy: each successive DRL agent inherits weights trained on earlier data, effectively expanding the training horizon and introducing a structural advantage. When this mechanism is removed and agents are initialized from scratch, performance remains strong but becomes more variable across years—especially during market downturns such as 2015, 2018, and 2020.

Finally, we observe clear differences between datasets. On the S&P500, naive strategies perform surprisingly well, suggesting that strong sector-wide trends can be captured even with simple heuristics. On MSCI indices, by contrast, DRL and long-horizon MVO offer more consistent advantages, reflecting greater cross-sectional dispersion and weaker passive baselines.

Overall, DRL demonstrates robust performance across both datasets, but its relative advantage depends strongly on the choice of benchmark and the fairness of the evaluation setup.

While this study tries to carefully replicate the experimental setting of Sood et al. (2023), differences in training seeds and implementation details may still lead to variation in DRL performance across runs. We make all code available to support full reproducibility

5.2 Limitations and Possible Extensions

TODO : be clearer

Several limitations of this study should be acknowledged.

First, the DRL agent has access to additional exogenous inputs—such as VIX and SPY-based

volatility indicators—that are not available to the MVO strategy, which relies only on past asset returns. This feature asymmetry may contribute to DRL's superior performance and complicates direct comparisons.

Second, the training regime for DRL follows the hyperparameter configuration and architecture proposed in Sood et al. (2023), which was selected through an extensive grid search. Due to computational constraints, we did not re-tune these parameters for our replication. As such, the results likely reflect an upper bound on DRL performance in this setting, and may not generalize without careful tuning.

Third, our evaluation starts in 2012 and does not cover earlier financial crises such as 2008. While this avoids confounding from the immediate post-crisis period, it limits insights into DRL's behavior during severe systemic shocks.

Finally, transaction costs and slippage are not included in any of the backtests. This omission is especially relevant for DRL, which exhibits significantly higher turnover than long-horizon MVO or passive strategies. Incorporating realistic trading frictions could materially affect the ranking of strategies in practice.

Potential extensions of this work include:

- Incorporating transaction cost models and retraining DRL agents under cost-penalized objectives.
- Exploring monthly or lower-frequency environments to test DRL's adaptability under reduced rebalancing frequency.
- Investigating regime-aware or hybrid approaches that combine DRL with MVO based on market volatility or drawdown signals.
- Comparing MVO performance under adaptive or rolling lookback windows, rather than fixed horizons.

Bibliography

CBOE (2024), 'CBOE Volatility Index (VIX)'. Accessed: 2025-07-02.
URL: *https://www.cboe.com/tradable_products/vix/*

DeMiguel, V., Garlappi, L. & Uppal, R. (2007), 'Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy?', *The Review of Financial Studies* **22**(5), 1915–1953.
URL: *https://doi.org/10.1093/rfs/hhm075*

Elton, E. J. & Gruber, M. J. (1971), 'Dynamic programming applications in finance', *The Journal of Finance* **26**(2), 473–506.

Ledoit, O. & Wolf, M. (2004), 'A well-conditioned estimator for large-dimensional covariance matrices', *Journal of Multivariate Analysis* **88**(2), 365–411.
URL: *https://www.sciencedirect.com/science/article/pii/S0047259X03000964*

Markowitz, H. (1952), 'Portfolio selection', *The Journal of Finance* **7**(1), 77–91.
URL: *http://www.jstor.org/stable/2975974*

Moody, J., Wu, L., Liao, Y. & Saffell, M. (1998), 'Performance functions and reinforcement learning for trading systems and portfolios', *Journal of Forecasting* **17**(5-6), 441–470.

Neuneier, R. (1995), Optimal asset allocation using adaptive dynamic programming, *in* 'Advances in Neural Information Processing Systems 8 (NIPS 1995)', MIT Press, pp. 952–959.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M. & Dormann, N. (2021), 'Stable-baselines3: Reliable reinforcement learning implementations', *Journal of Machine Learning Research* **22**(268), 1–8.
URL: *http://jmlr.org/papers/v22/20-1364.html*

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. (2017), 'Proximal policy optimization algorithms'.
URL: *https://arxiv.org/abs/1707.06347*

Securities, U. S. & Commission, E. (1998), *SEC Docket*, number v. 66, Securities and Exchange Commission.
URL: *https://books.google.de/books?id=kDsjWfruBzEC*

Shadwick, W. F. (2002), A universal performance measure con keating.
URL: *https://api.semanticscholar.org/CorpusID:16222368*

Sharpe, W. F. (1966), 'Mutual fund performance', *The Journal of Business* **39**(1), 119–138.
URL: *http://www.jstor.org/stable/2351741*

Sood, S., Papasotiriou, K., Vaiciulis, M. & Balch, T. (2023), Deep reinforcement learning for optimal portfolio allocation: A comparative study with mean-variance optimization, *in* 'Proceedings of the ICAPS 2023 Workshop on Financial Planning (FinPlan)'.
URL: *https://icaps23.icaps-conference.org/papers/finplan/FinPlan23_paper_4.pdf*

Sortino, F. & van der Meer, R. (1991), 'Downside risk: Capturing what's at stake in investment situations', *Journal of Portfolio Management* **17**(4), 27–31.

S&P Dow Jones Indices and MSCI (2023), 'Gics methodology', https://www.spglobal.com/spdji/en/documents/methodologies/methodology-gics.pdf. Accessed: 2025-07-01.

Young, T. W. (1991), 'Calmar ratio: A smoother tool', *Futures* **20**, 40.