Can Deep Reinforcement Learning Reliably Improve Dynamic Portfolio Allocation?

Rethyam Gupta[1], **Atharwa Pandey**[1], **and Adarsh Pandey**[1]

[1] School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14850, United States of America.
Email: rg795@cornell.edu, ap2467@cornell.edu, ap2459@cornell.edu

Abstract: This paper evaluates the effectiveness of Deep Reinforcement Learning (DRL) for dynamic portfolio allocation and benchmarks its performance against the classical Mean-Variance Optimization (MVO) framework. While DRL has gained significant attention for its ability to learn adaptive trading strategies from high-dimensional market environments, its evaluation is often limited to comparisons with other DRL variants rather than established portfolio theory methods. In this study, we implement a model-free DRL agent and compare it directly with a standard MVO approach, using an identical set of input parameters and rebalancing windows to ensure a fair comparison. Both methods are tested under realistic market conditions, incorporating transaction costs to assess robustness—an extension often ignored in earlier DRL studies that reported superior performance in cost-free environments. Through systematic backtesting across multiple market periods, we find that the DRL agent performs at least as well as, and often better than, MVO. While it does not consistently outperform MVO in every period, it is rarely worse. These results indicate that DRL holds significant potential for portfolio optimization. Its adaptability and promising results, especially in the presence of transaction costs, merit further exploration and development to achieve more consistent performance across all market regimes.

Keywords: Computational Finance, Deep Reinforcement Learning, Mean-Variance Optimization, Portfolio Allocation, Transaction Costs, Financial Backtesting

1. Introduction

Portfolio management plays a central role in the financial industry, involving the strategic allocation of capital across a variety of asset classes—including equities, fixed income instruments, and cash equivalents—with the objective of achieving favorable risk-adjusted returns while minimizing transaction costs and exposure to systemic risk. Effective portfolio construction is critical not only for institutional investors but also for individual investors seeking to maximize returns relative to the level of risk assumed.

The theoretical foundation of portfolio optimization was laid by Harry Markowitz in his seminal work on Modern Portfolio Theory (MPT), which introduced the mean-variance framework for balancing expected return and risk (Markowitz, 1952). Since then, the field has evolved significantly, with advancements in both mathematical optimization techniques and empirical implementation strategies. These include the incorporation of transaction costs, cardinality constraints, and asset weight limits into the optimization problem (Cornuéjols and Tütüncü, 2006; Kalayci et al., 2017). Moreover, researchers have explored dynamic portfolio selection models under realistic market frictions and constraints (Li and Hoi, 2014; Ghahtarani et al., 2022).

Recent years have also seen the rise of machine learning (ML) techniques in quantitative finance, particularly in the estimation of expected returns, covariance matrices, and feature selection for prediction models (Gu et al., 2020). ML methods, especially deep learning methods such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have enabled the modeling of complex, non-linear relationships between asset returns and macro-financial indicators (Fischer and Krauss, 2018).

Parallel to this evolution in ML, the domain of reinforcement learning (RL) has experienced substantial breakthroughs, especially in areas requiring sequential decision-making such as robotics and video games (Silver et al., 2016; Su et al., 2016). The adaptation of DRL, which integrates deep neural networks with RL algorithms, for financial applications has been particularly promising. Within finance, DRL has been deployed for autonomous stock trading (Yang et al., 2020; Théate and Ernst, 2021), risk management through deep hedging (Buehler et al., 2019; Cao et al., 2021), and increasingly for multi-asset portfolio optimization (Sood et al., 2021). Despite its empirical success, much of the existing literature on DRL in finance remains limited in certain aspects, one of which we aim to address in this study.

This paper evaluates the performance of a policy-gradient-based DRL framework designed to maximize risk-adjusted returns, using the MVO approach as a benchmark. The DRL agent is trained in a multi-asset trading environment that simulates the U.S. equity market through historical data replay. From asset price trajectories, the agent constructs observation states, which are converted into trading signals. Through experiential learning, it develops a continuous policy to allocate portfolio weights across multiple assets, offering a learning-based alternative to MVO's optimization logic.

We assess both strategies using industry-standard financial metrics. While previous studies, including Sood et al. (2021), have reported DRL outperforming MVO, they often do so without accounting for practical constraints such as transaction costs. In contrast, our study incorporates these costs and finds that the DRL agent almost always performs at least as well as, and often better than, the MVO benchmark. Although its performance declines relative to an unconstrained trading environment, DRL continues to exhibit strong results. These findings support prior research and highlight the

method's potential for further development into a standard tool in the industry.

The remainder of this paper is structured as follows: Section 2 presents a comprehensive literature review. Section 3 outlines the background of the models employed in this study. Section 4 describes the RL setup used in our framework. Section 5 presents the implementation, empirical results, and analysis. Finally, Section 6 discusses the limitations of the current framework and outlines directions for possible future work.

2. Literature Review

There has been a growing body of research on using DRL for financial problems, particularly in trading and portfolio optimization. Unlike traditional methods that rely on predefined statistical assumptions, DRL offers a data-driven, adaptive framework that learns asset allocation policies through interaction with the market. Several recent studies have shown that DRL agents can outperform traditional strategies by learning dynamic, nonlinear decision policies (Wang et al., 2018; Liang et al., 2018; Lu, 2017; Jiang and Liang, 2017; Wang et al., 2021; Deng et al., 2016; Cong et al., 2021).

These works vary in their algorithmic choices and experimental designs. Wang et al. (2018) applies DDPG for continuous portfolio weights, while Jiang and Liang (2017) uses CNNs to encode price data. Other approaches include deep Q-learning (Deng et al., 2016), LSTM-based recurrent RL (Lu, 2017; Liang et al., 2018), and distributional RL to account for return uncertainty (Cong et al., 2021).

To improve performance, many studies expand the agent's information set with technical indicators, volatility and volume signals (Liu et al., 2020; Du and Tanaka-Ishii, 2020), or external data such as macroeconomic variables and sentiment from news or social media (Ye et al., 2020; Du and Tanaka-Ishii, 2020; Lima Paiva et al., 2021). While these enhancements can boost DRL effectiveness, they also introduce asymmetries in comparisons with classical models like MVO, which rely solely on return and covariance estimates.

Despite recent advancements, evaluation practices in the DRL literature often fall short. Many studies benchmark against naive strategies like buy-and-hold or other DRL variants, which, while useful for validating innovation, lack practical relevance. A more meaningful comparison involves established methods like MVO, which remains widely used in asset management due to its simplicity. Although based on strong assumptions, MVO provides a robust baseline grounded in financial theory.

Some studies have attempted to incorporate MVO as a benchmark (Koratamaddi et al., 2021; Alonso et al., 2020; Zhang et al., 2020), but these comparisons are often inconsistent. Differences in training objectives, lack of methodological transparency, and asymmetric use of information limit their validity. For example, using the Sharpe ratio on daily time steps is often unstable, so many studies train DRL agents to directly

maximize returns or minimize variance. Similarly, MVO relies on at least one fixed estimate—such as expected returns or covariances—to ensure convexity. These differences in objectives and assumptions make direct comparisons very difficult. Furthermore, DRL agents frequently use a much richer information sets, creating asymmetry. Lastly, MVO implementations in prior studies often lack transparency—omitting details on return forecasts, covariance shrinkage, or portfolio constraints—hindering fair and reproducible comparisons.

A more robust comparison is provided by Benhamou et al. (2020), who evaluate DRL against MVO in a multi-asset context, emphasizing the importance of market regimes and nonlinearities. Another methodologically rigorous comparison is presented in Sood et al. (2021), which aligns objective functions, datasets, and evaluation procedures to provide a credible assessment of DRL versus MVO across rolling windows. This is also the study we aim to reproduce and extend in the present work.

In this paper, we aim to address the above discussed limitations by presenting a fair and transparent comparison between a model-free DRL agent and a classical MVO portfolio strategy. Both models are evaluated under an almost identical data universe, set of constraints, and evaluation periods, and are optimized to target the same reward function. This approach allows us to provide a clearer assessment of the practical viability of DRL in portfolio management, highlighting its strengths and weaknesses in adapting to nonlinear regimes while also acknowledging the robustness of MVO.

3. Background

In this section, we first introduce MVO, a central technique under MPT that determines the optimal asset weights to achieve the best trade-off between return and variance. This classical baseline will later serve as a point of comparison against the performance of our proposed DRL framework. We also briefly outline the key concepts of RL as a general-purpose learning paradigm, independent of its application to portfolio optimization. These preliminaries will facilitate a clearer understanding of the mechanics underlying our DRL-based investment strategy.

3.1. Mean-Variance Portfolio Optimization

MVO is a mathematical process for allocating capital across assets by optimizing portfolio weights to achieve a specific investment goal. Common objectives include:

1. Maximizing returns for a given level of risk,
2. Minimizing risk for a desired rate of return, or
3. Maximizing returns per unit of risk.

Risk is generally quantified by the volatility (i.e., variance) of the portfolio's returns.

To perform MVO for a set of assets, one requires the expected return of each asset and the covariance matrix of asset returns. Since true returns are unobservable, these are typically estimated using historical data or forecasted through various statistical or machine learning techniques.

This task is commonly formulated as a single or multi-objective optimization problem and solved using convex optimization approaches (Cornuéjols and Tütüncü, 2006; Kalayci et al., 2017; Ghahtarani et al., 2022). A standard method is to construct an efficient frontier of optimal portfolios, none of which can be improved in one metric (e.g., return) without worsening another (e.g., risk).

Let w be the vector of portfolio weights, μ the vector of expected returns, and Σ the covariance matrix of returns. The portfolio risk is represented by $w^\top \Sigma w$. To achieve a desired expected return μ^*, the optimization problem is formulated as:

$$\begin{aligned} \min_{w} \quad & w^\top \Sigma w \\ \text{s.t.} \quad & w^\top \mu \geq \mu^*, \\ & w_i \geq 0 \quad \forall i, \\ & \sum_i w_i = 1 \end{aligned} \tag{1}$$

By varying μ^*, one can trace out the efficient frontier of optimal portfolios.

Another popular objective is the Sharpe Ratio (Sharpe, 1998; Chen et al., 2011), which evaluates the expected excess return per unit of risk. For a portfolio p, the Sharpe Ratio is defined as:

$$\text{Sharpe Ratio}_p = \frac{\mathbb{E}[R_p - R_f]}{\sigma_p} \tag{2}$$

where $\mathbb{E}[R_p]$ is the expected portfolio return, σ_p is the standard deviation of returns, and R_f is the risk-free rate, often approximated as 0% for comparison purposes with different models.

Directly maximizing the Sharpe Ratio is a non-convex problem:

$$\max_{w} \quad \frac{w^\top \mu - R_f}{\sqrt{w^\top \Sigma w}} \tag{3}$$

However, this can be reformulated as a convex optimization problem using variable substitution techniques (Cornuéjols and Tütüncü, 2006). In this study, we adopt the Sharpe Ratio as our objective function along with some fixed transaction costs, enabling us to optimize for risk-adjusted returns without explicitly setting minimum

return targets or maximum risk thresholds. We will dive deep into the specifics of the MVO implementation in our study in Section 5.

3.2. Reinforcement Learning

RL is a subfield of machine learning concerned with how agents should take actions in an environment in order to maximize cumulative long-term rewards obtained through interactions with the environment (Sutton and Barto, 2018). Formally, an RL environment is modeled as a Markov Decision Process (MDP), defined by the 5-tuple $(\mathcal{S}, \mathcal{A}, P_a, R_a, \gamma)$, where:

1. $\mathcal{S}$ is a set of states,
2. $\mathcal{A}$ is a set of actions,
3. $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the transition probability of moving from state s to s' upon taking action a at time t,
4. $R_a(s, s')$ is the immediate reward received after transitioning from state s to s' via action a,
5. $\gamma \in [0, 1]$ is the discount factor that determines the present value of future rewards.

A solution to an MDP is a policy π, which maps each state s to an action $\pi(s)$ to be executed in that state. The objective of the RL agent is to learn a policy π that maximizes the expected discounted return over time:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})\right] \tag{4}$$

3.2.1. Deep Reinforcement Learning and Portfolio Allocation

The field of DRL extends classical RL by incorporating deep neural networks as function approximators. These networks are used to estimate either the state-action value function or the policy mapping π directly. This approach enables RL to scale to high-dimensional and complex decision-making tasks. DRL has achieved remarkable results across domains such as game playing (Mnih et al., 2013), robotics and continuous control (Berner et al., 2019), and financial decision-making (Nguyen and La, 2019; Hambly et al., 2021; Charpentier et al., 2021).

Given its effectiveness in complex stochastic control problems, reinforcement learning (RL) is well-suited for portfolio optimization. The application of DRL to trading and asset allocation has gained attention for its ability to model dynamic market behavior and adapt to changing conditions. Recent approaches leverage deep neural architectures—such as autoencoders and LSTMs—to learn feature embeddings that capture temporal and structural patterns in financial data.

These methods enhance decision-making by incorporating technical indicators and alternative data like financial news to reflect market sentiment. Additionally, attention mechanisms and graph-based neural networks have been used to model cross-asset dependencies, improving portfolio robustness, as discussed in detail in Section 2.

4. RL Problem Setup

As described in Section 3, RL involves an agent interacting with an environment through a sequence of states, actions, and rewards. In the portfolio optimization context, we simulate the U.S. equities market using historical market data and define observation states derived from asset prices. The agent outputs a portfolio allocation (weights), which is used to rebalance the portfolio at each timestep. Further ahead in this section, we first detail the components of the RL setup—specifically, the structure of states, actions, and the reward function. We then outline the learning algorithm and policy optimization strategy used in our framework, followed by a discussion of the specifics of the RL environment.

4.1. Actions

For portfolio allocation over N assets, the agent selects portfolio weights

$$w = [w_1, w_2, \ldots, w_N]$$

Such that:

$$\sum_{i=1}^{N} w_i = 1 \tag{5}$$

Where each weight satisfies:

$$0 \leq w_i \leq 1 \tag{6}$$

A weight of 0 indicates no holdings in a particular asset, while a weight of 1 indicates that the entire portfolio is allocated to that asset. Although extensions of this framework allow $w_i < 0$ (short selling) or $w_i > 1$ (leveraged positions), we restrict our actions to long-only, non-leveraged positions. These constraints are enforced using the softmax transformation on the continuous action vector output by the agent.

4.2. States

Let P_t denote the asset price at time t. The one-period simple return is given by:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{7}$$

The one-period gross return is:

$$\frac{P_t}{P_{t-1}} = R_t + 1 \tag{8}$$

The corresponding log return is:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right) = \log(R_t + 1) \tag{9}$$

We choose the time interval to be daily and compute daily log returns based on end-of-day close prices. The log return history for an asset over a lookback window of T days is:

$$r_t = [r_{t-1}, r_{t-2}, \ldots, r_{t-T+1}] \tag{10}$$

In our setup, $T = 60$ days. For a universe of $n + 1$ assets (including n tradable securities and one cash asset denoted by c), the state observed by the agent at time t is defined as a matrix $S_t \in \mathbb{R}^{(n+1)\times T}$:

$$S_t = \begin{bmatrix} w_1 & r_{1,t-1} & \cdots & r_{1,t-T+1} \\ w_2 & r_{2,t-1} & \cdots & r_{2,t-T+1} \\ \vdots & \vdots & \ddots & \vdots \\ w_n & r_{n,t-1} & \cdots & r_{n,t-T+1} \\ w_c & \mathrm{vol}_{20} & \mathrm{vol}_{20}/\mathrm{vol}_{60} & \mathrm{VIX}_t \end{bmatrix} \tag{11}$$

The first column represents the current portfolio allocation vector w, which may differ from the previously chosen allocation due to rebalancing and normalization. Each of the following rows contains the past time T log returns for a specific asset. Since our portfolio is always fully invested in the n risky assets, w_c is 0 at all times.

We also include three market volatility indicators at time t: vol_{20}, vol_{60}, and VIX_t (The CBOE Volatility Index). These features help the agent learn to adapt its portfolio strategy under varying market volatility conditions. A more detailed discussion of these indicators is provided in Section 5.

4.3. Reward

Rather than maximizing raw returns, most modern portfolio managers aim to optimize for risk-adjusted returns. Since we employ DRL for portfolio allocation, we design a reward function that encourages the agent to learn precisely this behavior.

The Sharpe Ratio is a widely adopted metric for evaluating risk-adjusted performance. However, it is traditionally defined over an evaluation window of size T, making it unsuitable for online learning where the agent receives rewards at each timestep.

To address this, we adopt the Differential Sharpe Ratio (DSR) D_t, introduced by Moody and Saffell (1998), which enables a per-timestep risk-adjusted reward signal and has been shown to produce more stable and profitable policies (Moody and Saffell, 2001; Dempster and Leemans, 2006).

The Sharpe Ratio S_t over t period returns R_t (after subtracting transaction costs) is defined as:

$$S_t = \frac{A_t}{\sqrt{B_t - A_t^2}} \tag{12}$$

Where:

$$A_t = \frac{1}{t}\sum_{i=1}^{t} R_i, \quad B_t = \frac{1}{t}\sum_{i=1}^{t} R_i^2 \tag{13}$$

Here, A_t and B_t are the sample mean and second moment of returns.

To make this computation recursive and suitable for streaming environments, we use exponential moving averages with time scale η^{-1}. Let:

$$A_t = A_{t-1} + \eta \Delta A_t, \quad B_t = B_{t-1} + \eta \Delta B_t \tag{14}$$

With:

$$\Delta A_t = R_t - A_{t-1}, \quad \Delta B_t = R_t^2 - B_{t-1} \tag{15}$$

Expanding S_t as a first-order Taylor series in η gives:

$$S_t \approx S_{t-1} + \eta D_t\Big|_{\eta=0} + O(\eta^2) \tag{16}$$

Therefore, the D_t is given by:

$$D_t = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}} \tag{17}$$

This formulation allows the agent to optimize for risk-adjusted return incrementally at each timestep. We initialize with $A_0 = B_0 = 0$, and typically set $\eta \approx \frac{1}{252}$ to reflect a typical trading year of 252 days.

4.4. Learning Algorithm

RL algorithms are broadly categorized into two classes: model-based and model-free, depending on whether the agent has access to or must learn a model of the environment dynamics (Sutton and Barto, 2018).

Model-free algorithms, which we utilize in this study, rely on direct experience to learn the consequences of actions without constructing an explicit model. These include methods such as Q-learning (Watkins and Dayan, 1992) and Policy Gradient

methods (Sutton et al., 2000), where the agent repeatedly interacts with the environment and adjusts its strategy (policy) based on the feedback it receives in order to maximize cumulative rewards.

4.5. Policy Optimization

Policy optimization methods focus on directly learning a policy function $\pi_\theta(a|s)$, which maps the current state s to a distribution over actions a. These methods optimize the parameters θ by performing gradient ascent on a performance objective $J(\pi_\theta)$ or by maximizing a local approximation of it. This optimization is typically performed in an on-policy setting, meaning that the experiences used for learning are generated by the current policy.

Notable examples of policy optimization algorithms include:

1. A2C / A3C (Advantage Actor-Critic) (Mnih et al., 2016)
2. PPO (Proximal Policy Optimization) (Schulman et al., 2017)

In this study, we adopt PPO due to its stability and sample efficiency for continuous control problems such as portfolio allocation.

4.6. RL Environment Specifics

The RL environment serves as a wrapper around the market, sliding through historical data in a technique known as market replay. It plays the role of a broker and exchange, processing the agent's actions, updating portfolio allocations, and feeding back observations and rewards.

At each timestep t, the environment computes the current portfolio value as:

$$port\_val_t = \sum_i P_{i,t} \cdot \text{weights}_{i,t-1} - C_t \tag{18}$$

Where $P_{i,t}$ is the price of asset i at time t, and $\text{weights}_{i,t-1}$ denotes the weights of assets held from the previous step, and C_t is the incurred transaction cost.

The environment then allocates this total portfolio value according to the agent's proposed weights $w_t = [w_{1,t}, w_{2,t}, \ldots, w_{n,t}]$, dividing it between the n assets.

After rebalancing, the environment constructs the next observation state S_{t+1} using updated returns and volatility metrics. It proceeds to the next timestep $t + 1$, computes the new portfolio value using updated prices P_{t+1}, and calculates the agent's reward:

$$R_t = D_t \tag{19}$$

Where D_t is the Differential Sharpe Ratio as defined earlier (Eq. 17).

In our setup, we assume a fixed transaction cost of 10 basis points (0.1%) per transaction and allow instantaneous rebalancing of the portfolio for simplicity and computational efficiency. Incorporating transaction costs enables a more realistic evaluation of model performance under real-world trading conditions—an aspect often omitted in prior studies such as Sood et al. (2021), which conducted similar comparisons without accounting for trading frictions.

5. Implementation and Evaluation

This section details the implementation of the MVO and DRL models from Sections 3 and 4, along with their results. We first describe the data used, then outline our benchmark MVO implementation, followed by the DRL implementation, including the training and testing of the model for portfolio allocation. Finally, we present empirical results comparing both strategies under equivalent conditions.

5.1. Data Description

We utilize daily adjusted close price data from Bloomberg (access provided by Cornell University), spanning the period from January 2006 to December 2024. The dataset includes all GICS-classified sector ETFs that comprise the S&P 500 index, the S&P 500 index itself (SPX), and the CBOE Volatility Index (VIX). The sector ETFs used in our analysis are as follows, also shown in Figure 1:

1. S5TELS – Communication Services
2. S5CONDS – Consumer Discretionary
3. S5CONS – Consumer Staples
4. S5ENRS – Energy
5. S5FINL – Financials
6. S5HLTH – Health Care
7. S5INDU – Industrials
8. S5MATR – Materials
9. S5RLST – Real Estate
10. S5INFT – Information Technology
11. S5UTIL – Utilities

To calculate returns, we first compute the daily log returns from the adjusted close prices. These log returns serve as the basis for portfolio optimization and are also used to compute volatility-based regime indicators. To capture market regime shifts, we derive three volatility-based metrics from the S&P 500 index:

1. vol20: 20-day rolling standard deviation of daily log returns.
2. vol60: 60-day rolling standard deviation of daily log returns.

3. vol_ratio (vol20/vol60): Captures short-term vs. long-term volatility trends.

A ratio greater than 1 indicates that recent volatility is higher than the longer-term volatility, potentially signifying a transition to a more volatile regime.

For the DRL model, we include vol20, vol_ratio, and the VIX in the observation space as shown in Eq. 11. These features are standardized using an expanding window mean and standard deviation to prevent look-ahead bias or information leakage.

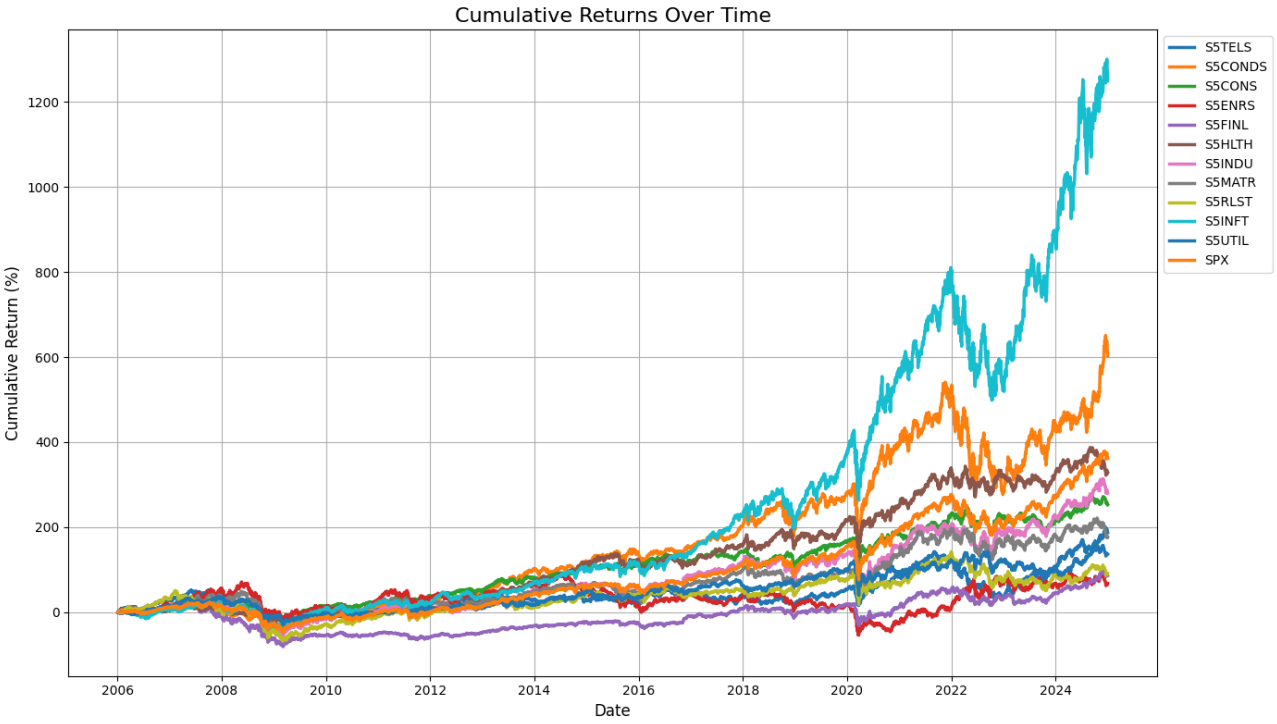

Figure 1. Cumulative returns of S&P 500 sector ETFs and SPX (2006-2024).

5.2. MVO Implementation

To provide a benchmark for our DRL-based portfolio allocation framework, we implement the MVO model under equivalent operational conditions. Specifically, we use a rolling 60-day lookback window—matching the DRL setup—to estimate both the expected returns and the covariance matrix of asset returns.

The expected returns vector μ_t at time t is computed as the sample mean of log returns over the 60 most recent trading days. However, the sample covariance matrix is known to be unstable and prone to estimation error, which can adversely affect optimization results. To address this issue, we employ the Ledoit-Wolf shrinkage estimator (Ledoit and Wolf, 2004), which provides a more robust estimate of the covariance matrix by shrinking the sample matrix toward a well-conditioned target, using the implementation provided by `sklearn.covariance.LedoitWolf` (Pedregosa et al., 2011):

$$\Sigma_{\mathrm{LW}} = \delta F + (1 - \delta) S_t \tag{20}$$

Where:

1. S_t is the empirical sample covariance matrix,
2. F is the shrinkage target, typically the identity-scaled matrix or the constant correlation matrix,
3. $\delta \in [0, 1]$ is the shrinkage intensity, computed analytically to minimize mean squared error.

To ensure numerical stability and feasibility in optimization, we further enforce that the final covariance matrix is positive semi-definite by setting any negative eigenvalues to zero and reconstructing the matrix. At each rebalancing date (daily, in both MVO and DRL), we obtain the portfolio weights by solving:

$$\min_{w \geq 0} \; -\mu^\top w + \tau \|w - w_{t-1}\|_1 \quad \text{s.t.} \quad w^\top \widehat{\Sigma}_{\mathrm{LW}} w \leq 1 \tag{21}$$

Where τ is the per-unit transaction cost and $\widehat{\Sigma}_{\mathrm{LW}}$ is the Ledoit–Wolf-shrinkage covariance matrix. This convex optimization problem is implemented using the CVXPY library in Python (Diamond and Boyd, 2016). The quadratic constraint fixes total portfolio variance to one unit; the objective therefore maximises excess return per unit risk (a Sharpe-ratio proxy) while penalising large reallocations. The ℓ_1 penalty delivers sparse, low-turnover trades, and the shrinkage estimator provides a well-conditioned risk matrix, yielding stable weights even with short look-back or large asset universes. Following optimization, the resulting portfolio is normalized to ensure full investment by rescaling the weights to sum to one.

5.3. DRL Implementation

We begin by detailing the training procedure used for the DRL agent before describing the PPO algorithm implementation in the following subsection.

5.3.1. Training Process

Although financial data is notoriously scarce, especially at the daily scale, we aim to evaluate the DRL framework across multiple years using backtesting. Moreover, financial time series exhibit non-stationarity, as noted by Cont (2001), which can degrade model performance if not addressed. To adapt to changing market dynamics, we retrain or fine-tune models using the most recently available data.

To this end, we employ a rolling-window experiment design. The dataset is partitioned into 13 overlapping groups, each shifted forward by one year. Each group

contains 7 years of data: the first 5 years are used for training, the 6th year serves as a validation period (burn-in), and the 7th year is reserved for out-of-sample backtesting.

For each training window, we initialize five DRL agents with different random seeds to account for variance in training. These agents are trained using the fixed hyperparameters detailed in the next subsection. During training, each agent's performance is periodically evaluated on the corresponding validation year. At the end of training, the agent achieving the highest mean episode reward on the validation set is selected as the best-performing policy.

This best agent is then used as the initialization (seed policy) for the next training group. For example, the first group spans the period [2006–2012), with 2006–2010 used for training, 2011 for validation, and 2012 for testing. The selected best agent is then used as the base policy for training in the next group: [2007–2013), with 2012 as validation and 2013 as the testing year. This iterative procedure continues until the final testing year, 2024.

5.3.2. PPO Implementation & Hyperparameters

We adopt the PPO algorithm using the implementation provided by `StableBaselines3` (Raffin et al., 2021). The hyperparameters (Table 1) were selected based on empirical guidelines and follow best practices from the literature (Engstrom et al., 2019; Rao et al., 2020), except for the number of timesteps, which was limited to 200,000 per training round due to computational limits. A coarse grid search over held-out validation data was also conducted to fine-tune the configuration.

Table 1. PPO Hyperparameters

Parameter	**Value**
Timesteps per round	200,000
Parallel environments (n_{envs})	10
Rollout buffer size (n_{steps})	$252 \times 3 \times n_{\text{envs}}$
Learning rate (decay)	$3 \times 10^{-4} \rightarrow 1 \times 10^{-5}$
Batch size	1260
Epochs per update	16
Discount factor (γ)	0.9
GAE lambda (λ)	0.9
Clip range	0.25
Network architecture	[64, 64], tanh activations
Initial log std ($\log \sigma_{\text{init}}$)	-1

To improve training efficiency and exploration, we utilize the `VecEnv` parallelization wrappers from `StableBaselines3`, which enable simultaneous experience collec-

tion across multiple independent environments. Specifically, we train the agent across $n_{\text{envs}} = 10$ parallel environments, allowing for diverse rollouts per PPO update step.

The learning rate follows a linear decay schedule, starting from 3×10^{-4} and annealing to 1×10^{-5} as training progresses, allowing for stable learning while accommodating the non-stationarity and noise present in financial time series.

5.4. Evaluation

We evaluate the performance of both the DRL and MVO strategies through 13 independent backtests, conducted on a yearly basis from 2012 to 2024. Each strategy begins every test year with a fresh portfolio valued at $100,000, and the portfolio is reset at the start of each new year. This design allows us to isolate annual performance and analyze behavior across varying market conditions.

Trading is carried out on a daily frequency using portfolio weights derived from each method, subject to the following constraints:

$$\sum_i w_i = 1, \quad \textit{and} \quad 0 \le w_i \le 1$$

This ensures full investment and prohibits short-selling.

Based on these weights, we compute daily portfolio values and returns. Unlike prior works that rely on third-party libraries (e.g., Pyfolio) for evaluation, we implement all performance metrics in-house using Python. The computed metrics include annualized return, annualized volatility, Sharpe ratio, Sortino ratio, Calmar ratio, maximum drawdown, and daily Value-at-Risk, as discussed in the work ahead.

DRL Agent: We evaluate the trained PPO agents in deterministic mode. For each yearly backtest, the selected DRL agent is trained on a rolling 5-year window followed by a 1-year validation. For example, the agent used to backtest 2017 is trained on data from [2011–2016), with 2016 serving as the validation year.

MVO: The MVO strategy does not require training. It computes optimal portfolio weights using a rolling 60-day lookback prior to each trading day. For instance, to initiate the 2012 backtest, MVO begins using historical price data from October 2011.

5.5. Results and Analysis

Figure 2 and Table 2 provide a comprehensive comparison of key financial performance metrics—including annual return, annual volatility, Sharpe ratio, Sortino ratio, Calmar ratio, maximum drawdown, and daily VaR—for both the DRL and MVO strategies over the period from 2012 to 2024. A clear pattern emerges across all plots: the performance of DRL is consistently at least equal to that of MVO, and in most years, DRL outperforms MVO in terms of risk-adjusted returns. However, this improved performance comes with a trade-off—DRL tends to offer less downside protection and

exhibits higher risk during adverse market conditions, a limitation we explore in more detail ahead as we discuss individual metrics.

The annual return plot shows that DRL consistently outperforms or closely matches MVO across most years, including periods of market turbulence such as 2018, 2020, and 2022. In years like 2013, 2019, and 2021, the DRL strategy delivers significantly higher returns than MVO. While the margin varies year to year, DRL rarely underperforms MVO, indicating its ability to adapt and capture market opportunities effectively. In contrast, MVO, with its reliance on historical return and covariance estimates, tends to produce more stable but generally lower returns.

The annual volatility plot shows that DRL consistently exhibits more volatility than MVO across the entire period. This difference becomes especially prominent in years like 2020 and 2022, where DRL's volatility spikes much higher than that of MVO. Even in relatively stable years, DRL tends to remain slightly more volatile. This behavior reflects a key trade-off: while DRL often delivers higher returns, it does so with increased risk and less stability. These results reinforce the observation that DRL's decisions are more aggressive and reactive to market shifts, contributing to its higher variance.

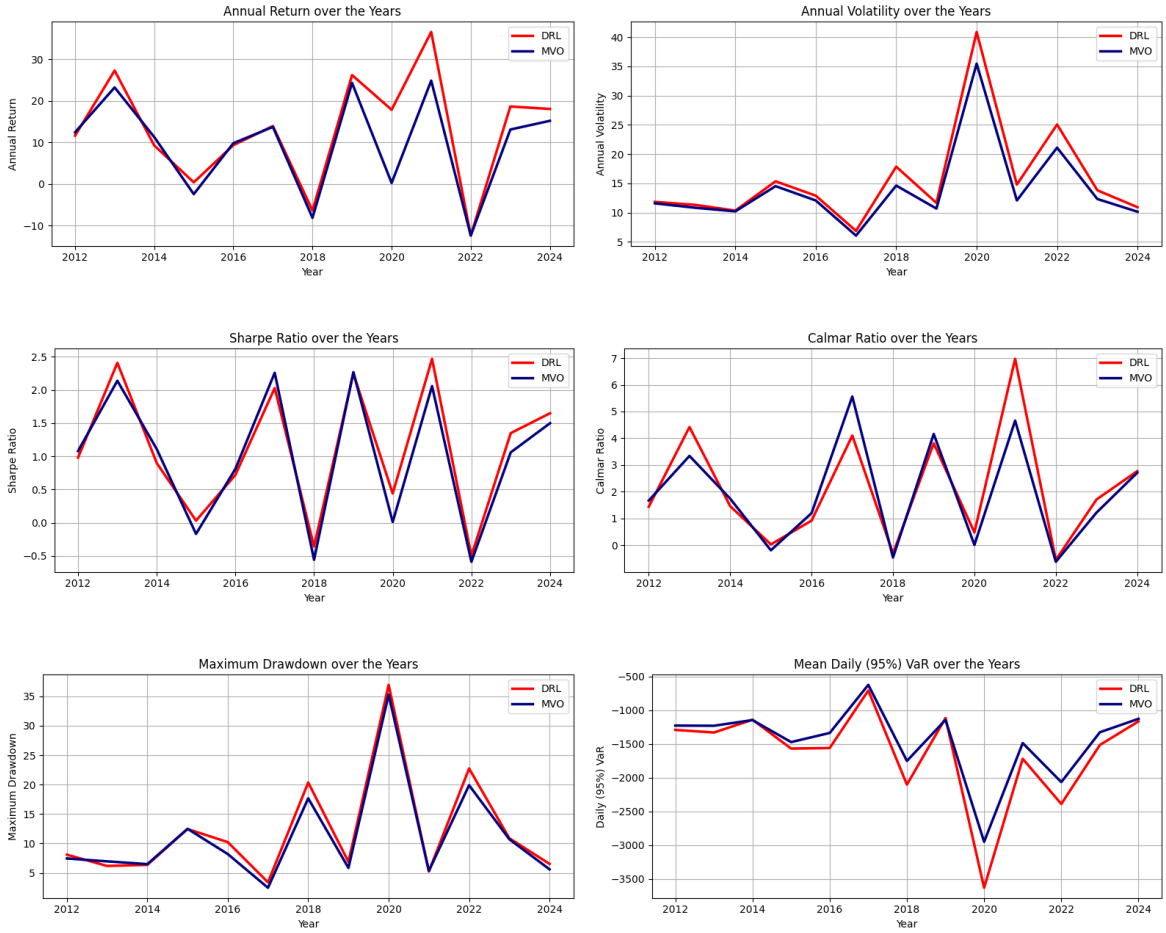

Figure 2. DRL and MVO Comparison: Standard Financial Metrics

The Calmar ratio, maximum drawdown, and Sharpe ratio plots further reinforce this trade-off. While DRL often achieves higher returns, it does so with noticeably greater variance and less downside protection. The Calmar ratio—which captures return relative to maximum drawdown—is generally lower for DRL compared to MVO, especially during volatile periods. Although both strategies suffer in crisis years, DRL's more aggressive positioning leads to deeper drawdowns in most cases.

On the other hand, the Sharpe ratio plot shows that DRL still maintains competitive—often higher—risk-adjusted returns compared to MVO in nearly all years, except for a few where the high-return, high-risk trade-off of DRL eventually leads to poorer risk-adjusted performance, such as in 2017. This highlights the central trade-off: DRL offers better upside potential but at the cost of increased volatility and downside risk.

In support of prior work—such as Sood et al. (2021)—which found that DRL consistently outperformed traditional optimization methods without any external constraints, our findings largely align with this conclusion, even after accounting for realistic market frictions. We incorporate a fixed 10 basis point cost per transaction. Despite this added constraint, DRL continues to outperform or at least match MVO in almost all years, reaffirming its robustness. However, we do observe that the margin of outperformance narrows considerably under this setting, primarily due to the high portfolio turnover associated with DRL. Since DRL frequently adjusts its allocations based on market signals, it incurs significantly higher trading costs, which ultimately erode some of its net performance gains. Table 3 presents the yearly accumulation of trading costs for both strategies, further supporting this observation.

Table 2 provides the complete numerical summary of all evaluation metrics across the years. These values were computed using custom-built Python functions to maintain consistency and transparency in our experimental framework, rather than relying on off-the-shelf financial analysis libraries. Additionally, to complement the metric-based assessment, we plot the daily portfolio values across all years for both DRL and MVO (Figure 3). These trajectories display the superior performance of DRL in most periods, though accompanied by more erratic movements and sharper drawdowns compared to the smoother paths of MVO portfolios.

Taken together, these results suggest that the advantages of DRL persist even under realistic trading conditions. Although the inclusion of transaction costs reduces the margin of outperformance, DRL still manages to outperform or at least match MVO in most periods. However, this comes at the cost of increased portfolio turnover, higher variance, and reduced downside protection. In contrast, MVO—despite its reliance on historical estimates and simplifying assumptions—continues to offer more stable and consistent performance. These findings underscore the potential of DRL while also highlighting the need for future research to develop cost-aware, risk-sensitive DRL frameworks that enhance robustness under real-world constraints.

Table 2. Performance Metrics with DRL and MVO Comparison

Year	Annual Return%		Annual Volatility%		Calmar		Max Drawdown%		Sharpe		Sortino		Daily VaR ($)	
	DRL	MVO	DRL	MVO	DRL	MVO	DRL	MVO	DRL	MVO	DRL	MVO	DRL	MVO
2012	11.64	12.46	11.84	11.59	1.44	1.67	8.09	7.45	0.98	1.08	1.51	1.67	-1294	-1229
2013	27.32	23.25	11.33	10.84	4.42	3.34	6.18	6.96	2.41	2.14	3.40	3.11	-1333	-1232
2014	9.30	11.32	10.35	10.22	1.46	1.74	6.37	6.49	0.90	1.11	1.18	1.47	-1143	-1148
2015	0.41	-2.43	15.36	14.53	0.03	-0.19	12.45	12.49	0.03	-0.17	0.04	-0.24	-1570	-1474
2016	9.36	9.83	12.90	12.08	0.92	1.20	10.23	8.20	0.72	0.81	0.96	1.09	-1563	-1338
2017	13.96	13.75	6.89	6.08	4.10	5.56	3.40	2.47	2.03	2.26	2.87	3.20	-710	-626
2018	-6.37	-8.13	17.86	14.63	-0.31	-0.46	20.36	17.67	-0.36	-0.56	-0.45	-0.67	-2105	-1754
2019	26.23	24.34	11.67	10.70	3.81	4.16	6.88	5.85	2.25	2.27	3.00	2.85	-1118	-1145
2020	17.87	0.26	40.87	35.46	0.48	0.01	36.95	35.29	0.44	0.01	0.56	0.01	-3634	-2952
2021	36.60	24.89	14.80	12.09	6.97	4.66	5.25	5.34	2.47	2.06	3.91	2.76	-1722	-1488
2022	-12.41	-12.38	25.06	21.11	-0.55	-0.62	22.74	19.90	-0.50	-0.59	-0.75	-0.91	-2392	-2067
2023	18.67	13.12	13.83	12.34	1.71	1.22	10.89	10.72	1.35	1.06	2.22	1.69	-1517	-1328
2024	18.07	15.22	10.93	10.16	2.77	2.71	6.53	5.61	1.65	1.50	2.26	1.96	-1168	-1129

Table 3. Annual Transaction Costs ($) by Year DRL and MVO Comparison

Model	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024
DRL	2603	1766	1518	2772	2027	714	2219	2489	1902	1802	1642	1442	2464
MVO	319	256	285	257	339	387	408	366	1343	750	418	387	461

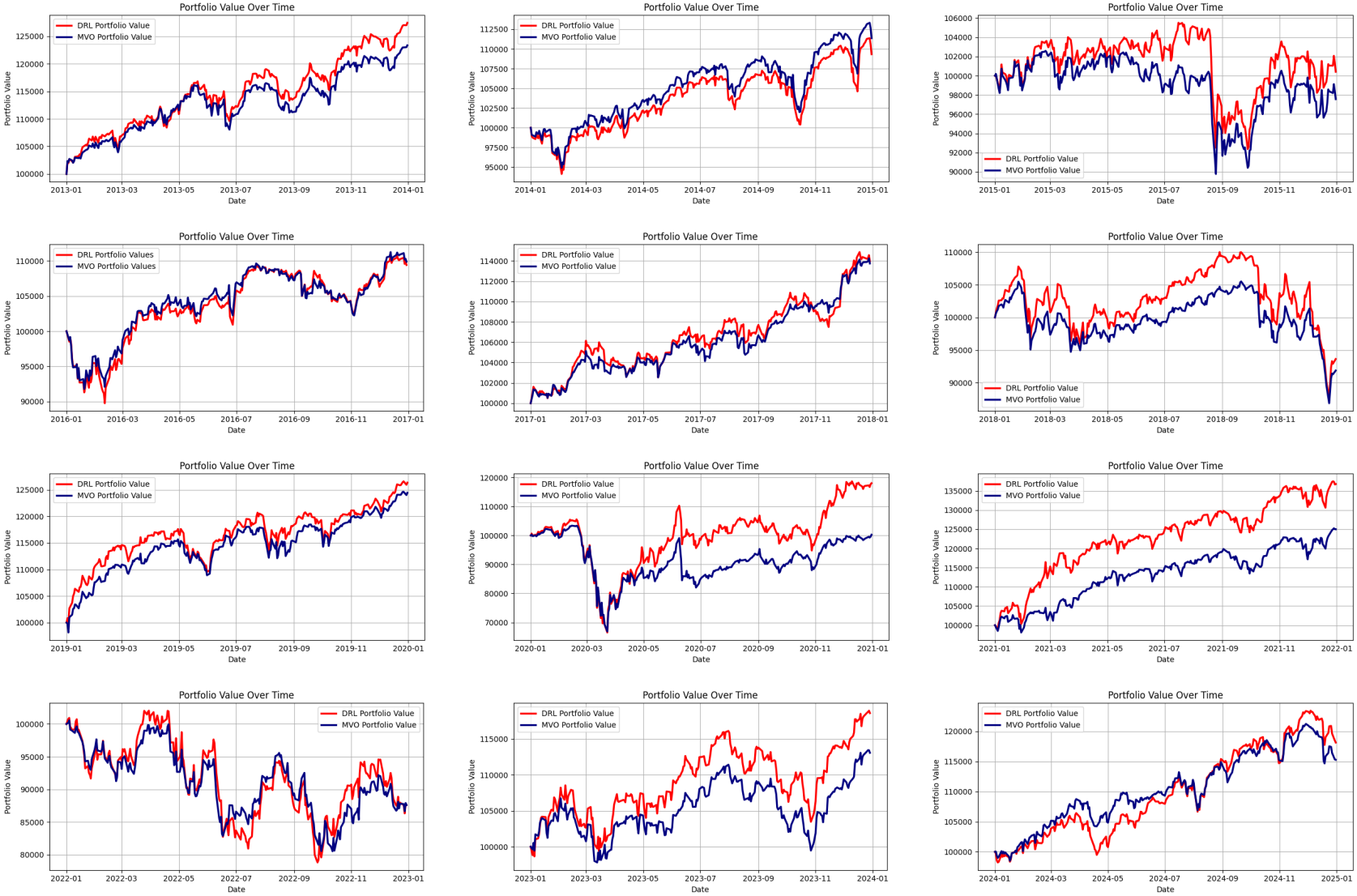

Figure 3. Yearly Portfolio Values from 2013 to 2024 (Top-left: 2013, progressing right and then downward to Bottom-right: 2024)

6. Limitations and Discussion

Our results demonstrate that DRL offers a flexible and adaptive framework for portfolio optimization, even when realistic market frictions—particularly transaction costs—are taken into account. Although the margin of outperformance narrows, DRL still delivers returns that are comparable to, or better than, those of the MVO approach in most years. However, this improved performance comes with trade-offs: higher transaction costs due to frequent rebalancing and greater portfolio instability. These factors diminish some of DRL's theoretical advantages and highlight the importance of developing more cost-aware and risk-sensitive adaptations for practical deployment.

One important limitation of this study lies in our DRL training configuration. Due to computational resource constraints, we were restricted to a maximum of 200,000 time steps per training round—a little below the typical threshold required to train high-performing DRL agents in financial applications. Despite this limitation, the resulting models exhibited behavior that closely mirrored MVO in terms of directional market exposure (Figure 2), indicating that the agent learned a reliable, if under-optimized, allocation policy. Nonetheless, further training could potentially improve policy stability and return efficiency, especially in complex or noisy market environments.

Looking ahead, we are particularly interested in experimenting with dynamic DRL architectures that incorporate regime-awareness. One promising direction is to develop a regime-switching model that allocates funds between two specialized agents based on market volatility levels. This framework would allow the system to explicitly switch behaviors rather than relying on an implicit encoding of regime characteristics within a single policy. Additionally, given the fast-paced innovation in the DRL domain, there is considerable potential to incorporate recent algorithmic advances—such as distributional RL, meta-learning, or transformer-based policy networks—into portfolio optimization frameworks. These techniques could enhance representation learning, generalization, and risk control.

We also acknowledge the need to extend our comparative framework across a wider variety of financial contexts. Future work should include testing DRL models on different asset classes (e.g., bonds, commodities, cryptocurrencies), as well as under multi-objective setups that go beyond Sharpe ratio maximization. Moreover, alternative benchmarks such as risk-parity approaches, or even hybrid rule-based strategies could provide a more comprehensive evaluation landscape.

In summary, our study shows that DRL surpasses MVO even under realistic constraints, highlighting its potential as a powerful tool for portfolio optimization. As the field continues to evolve—with improvements in training pipelines, model architectures, and regime-switching strategies—DRL-based approaches are likely to play an increasingly important role in modern asset management.

Acknowledgments

We would like to express our sincere gratitude to Professor James Renegar for his consistent support, insightful feedback, and invaluable guidance throughout the course of this project.

Supplementary Materials

The following URL provides all necessary resources to replicate the results presented in this paper.

https://drive.google.com/drive/folders/1CavtkRDBswBONEhI8stvxs90hMYJ32h8

References

Alonso, J., Srivastava, S., and Goodell, J. (2020). Portfolio management using reinforcement learning. In *Proceedings of the International Conference on Computational Intelligence*.

Benhamou, E., Buehler, H., Gonon, L., and Teichmann, J. (2020). Deep hedging of derivatives using reinforcement learning. *Risk.net*.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Hesse, C., Petrov, M., and et al. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Buehler, H., Gonon, L., Teichmann, J., and Wood, B. (2019). Deep hedging. *Quantitative Finance*, 19(8):1271–1291.

Cao, J., Wang, Y., Zhang, W., and Li, J. (2021). Deep hedging via reinforcement learning. *Expert Systems with Applications*, 182:115173.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. https://github.com/DLR-RM/stable-baselines3.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., and Madry, A. (2019). Implementation matters in deep RL: A case study on PPO and TRPO. In *International Conference on Learning Representations (ICLR)*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rao, D., Tokgoz, A., and Khandani, A. E. (2020). RL-based financial trading with stock fundamentals. *arXiv preprint arXiv:2012.06536*.

Charpentier, A., Elie, R., and Remlinger, C. (2021). Deep reinforcement learning in finance. *Risks*, 9(2):21.

Chen, L., He, H., and Zhang, J. (2011). A robust portfolio selection problem with the Sharpe ratio. *European Journal of Operational Research*, 212(2):396–404.

Cong, Y., Bai, Y., and Wang, Y. (2021). Alphaportfolio: A deep reinforcement learning framework for portfolio optimization. *Expert Systems with Applications*, 168:114340.

Cornuéjols, G. and Tütüncü, R. (2006). *Optimization Methods in Finance*. Cambridge University Press.

Dempster, M. A. H. and Leemans, V. (2006). An automated FX trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664.

Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236.

Du, L. and Tanaka-Ishii, K. (2020). Empirical study on deep learning models for financial NLP. *arXiv preprint arXiv:2003.11156*.

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669.

Ghahtarani, A., Saif, L., and Ghasemi, P. (2022). Multi-objective portfolio optimization: A machine learning approach. *Expert Systems with Applications*, 199:116926.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Hambly, B., Xu, J., and Yang, H. (2021). Deep learning for optimal investment. *Mathematics and Financial Economics*, 15(3):401–450.

Jiang, Z. and Liang, J. (2017). Cryptocurrency portfolio management with deep reinforcement learning. In *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, pages 581–589. Springer.

Kalayci, C. B., Onsel, S., and Kabak, Ö. (2017). A new model for cardinality constrained mean-variance portfolio optimization problem. *Expert Systems with Applications*, 85:26–36.

Koratamaddi, L., Reddy, R., and Srivastava, A. (2021). Deep reinforcement learning-based portfolio management using clustered RL environment. *arXiv preprint arXiv:2108.06198*.

Ledoit, O. and Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411.

Li, X. and Hoi, S. C. (2014). Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)*, 46(3):1–36.

Liang, Z., Chen, M., Zhu, Y., Jiang, B., Li, Y., Li, Y., and Wang, J. (2018). Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940*.

Lima Paiva, J., Rabelo, D., Mourelle, L., and Nobre, C. (2021). Sentiment-based reinforcement learning for financial trading. *Expert Systems with Applications*, 168:114339.

Liu, Y., Wang, Y., Shao, Y., Yang, T., and Wu, L. (2020). FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*.

Lu, Y. (2017). A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*.

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1):77–91.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1928–1937.

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

Moody, J. and Saffell, M. (2001). Learning to trade via direct reinforcement. In *Proceedings of the International Conference on Artificial Intelligence*, pages 853–859.

Nguyen, T. T. and La, Q. D. (2019). Deep reinforcement learning for dynamic portfolio optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 30(3):661–674.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sharpe, W. F. (1998). The Sharpe ratio. *The Journal of Portfolio Management*, 24(3):49–58.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... and Hassabis, D. (2016). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359.

Sood, S., Papasotiriou, K., Vaiciulis, M., and Balch, T. (2021). Deep reinforcement learning for optimal portfolio allocation: A comparative study with mean-variance optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11114.

Su, S. Y., Chen, S. H., and Hsiao, Y. C. (2016). Financial decision-making using a reinforcement learning approach. *Neurocomputing*, 205:50–58.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition.

Théate, T. and Ernst, D. (2021). Application of deep reinforcement learning to trading and asset management: A survey. *Expert Systems with Applications*, 165:113816.

Wang, Y., Jin, Y., Liu, Y., and Ma, H. (2018). A deep reinforcement learning framework for the financial portfolio management problem. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1401–1408.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3–4):279–292.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, pages 1057–1063.

Wang, Y., Chen, H., Jiang, D., and Ma, H. (2021). Adaptive deep reinforcement learning for online portfolio selection. *Expert Systems with Applications*, 168:114412.

Yang, X., Liu, Y., Xiao, W., and Liu, M. (2020). Deep reinforcement learning for automated stock trading: An ensemble strategy. *arXiv preprint arXiv:2004.06627*.

Ye, Q., Liu, Y., Li, T., Zhang, H., and Zhang, Y. (2020). Identifying financial risk via deep reinforcement learning. *Information Sciences*, 512:1284–1297.

Diamond, S. and Boyd, S. (2016). CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *Journal of Machine Learning Research*, 17(83):1–5.

Zhang, J., Zohren, S., and Roberts, S. (2020). Deep reinforcement learning for trading. *Journal of Financial Data Science*, 2(2):25–40.