Deep Reinforcement Learning for Optimal Portfolio Allocation: A Comparative Study with Mean-Variance Optimization

Srijan Sood, [1] **Kassiani Papasotiriou,** [1] **Marius Vaiciulis,**[2] **Tucker Balch** [1]**.**
[1] J.P. Morgan AI Research
[2] J.P. Morgan Global Equities; Oxford-Man Institute of Quantitative Finance
{srijan.sood, kassiani.papasotiriou, marius.vaiciulis, tucker.balch}@jpmorgan.com

Abstract

Portfolio Management is the process of overseeing a group of investments, referred to as a portfolio, with the objective of achieving predetermined investment goals and objectives. Portfolio Optimization is a key component that involves allocating the portfolio assets so as to maximize returns while minimizing risk taken. Portfolio optimization is typically carried out by financial professionals who use a combination of quantitative techniques and investment expertise to make decisions about the portfolio allocation.

Recent applications of Deep Reinforcement Learning (DRL) have shown promising results when used to optimize portfolio allocation by training model-free agents on historical market data. Many of these methods compare their results against basic benchmarks or other state-of-the-art DRL agents but often fail to compare their performance against traditional methods used by financial professionals in practical settings. One of the most commonly used methods for this task is Mean-Variance Portfolio Optimization (MVO), which uses historical timeseries information to estimate expected asset returns and covariances, which are then used to optimize for an investment objective.

Our work is a thorough comparison between model-free DRL and MVO for optimal portfolio allocation. We detail the specifics of how to make DRL for portfolio optimization work in practice, also noting the comparable tweaks needed to get MVO operational. Backtest results display strong improved performance of the DRL agent in terms of many performance metrics, including Sharpe ratio, maximum drawdowns, and absolute returns.

Introduction

Portfolio management is a key issue in the financial services domain. It constitutes allocating funds across a diverse variety of assets, typically to generate uncorrelated returns while minimizing risk and operational costs. Portfolios can constitute holdings across asset classes (cash, bonds, equities, etc.), or can also be optimized within a specific asset class (e.g., picking the appropriate composition of stocks for an equity portfolio). Investors may choose to optimize for various performance criteria, often centered around maximizing portfolio returns relative to risk taken. Since the advent of Modern Portfolio Theory (Markowitz 1952), a lot of progress has been made in both theoretical and applied aspects of portfolio optimization. These range from improvements in the optimization process, to the framing of additional constraints that might be desirable to rational investors (Cornuejols and Tütüncü 2006; Li and Hoi 2014; Kalayci et al. 2017; Ghahtarani, Saif, and Ghasemi 2022). Recently, the community has tapped the many advancements in Machine Learning (ML) to aid with feature selection, forecasting and estimation of asset means and covariances, as well as using gradient based methods for optimization.

Concurrently, the past decade has witnessed the success of Reinforcement Learning (RL) in the fields of gaming, robotics, natural language processing etc. (Silver et al. 2017; Nguyen and La 2019; Su et al. 2016). The sequential decision making nature of Deep RL, along with its success in applied settings, has captured the attention of the finance research community. In particular, some of the most popular areas of focus of the application of DRL in finance have been on automated stock trading (Yang et al. 2020; Théate and Ernst 2021; Zhang, Zohren, and Roberts 2020; Wu et al. 2020), risk management through deep hedging (Buehler et al. 2019; Du et al. 2020; Cao et al. 2021; Benhamou et al. 2020b) and portfolio optimization. In the upcoming section, we'll examine the landscape of DRL in portfolio optimization and trading problems. While these approaches exhibit improved performance than previous studies, they do have some shortcomings. For instance, some generate discrete asset trading signals which limit their use in broader portfolio management. Additionally, majority of these approaches compare results against ML or buy-and-hold baselines, and don't consider classical portfolio optimization techniques, such as Mean-Variance Optimization.

In our work, we aim to compare a simple and robust DRL framework, that was designed around risk-adjusted returns, with one of the traditional finance methods for portfolio optimization, MVO. We train policy gradient based agents on a multi asset trading environment that simulates the US Equities market (using market data replay), and create observation states derived from the observed asset prices. The agents optimize for risk-adjusted returns, not dissimilar to the traditional MVO methods. We compare the performance of the DRL strategy against MVO through a series of systematic backtests, and observe improved performance along many performance metrics, including risk adjusted returns, max drawdown, and portfolio turnover.

Related Work

There is a lot of recent research interest into the application of Deep RL in trading and portfolio management problems. For portfolio optimization, a lot of the research focuses on defining various policy network configurations and reports results that outperform various traditional baseline methods (Wang et al. 2019; Liang et al. 2018; Lu 2017; Jiang and Liang 2017; Wang et al. 2021; Deng et al. 2016; Cong et al. 2021). Other work explores frameworks that inject information in the RL agent's state by incorporating asset endogenous information such as technical indicators (Liu et al. 2020; Sun et al. 2021; Du and Tanaka-Ishii 2020) as well as exogenous information such as information extracted from news data (Ye et al. 2020; Lima Paiva et al. 2021).

The current benchmarks for DRL frameworks typically involve comparing results against other DRL or ML approaches, a buy-and-hold baseline, or market/index performance. However, these benchmarks may be overly simplistic or provide only a relative comparison. To truly gauge the effectiveness of a DRL agent, it would be more meaningful to benchmark it against methodologies used by financial professionals in practice, such as Mean Variance Optimization (MVO).

While there are some approaches that compare DRL performance with MVO (Li et al. 2019; Koratamaddi et al. 2021; i Alonso, Srivastava et al. 2020), the comparison simply serves as another baseline, and the methodology is not clearly described because an in-depth comparison is not the primary focus of their study. To our knowledge, there is only one study that goes into a robust in-depth comparison of MVO and DRL (Benhamou et al. 2020a). However, across all these studies, there is usually a discrepancy between the reward function used to train the RL agent, and the objective function used for MVO (for e.g., daily returns maximization vs risk minimization). In order to make a fair comparison, it is crucial that both approaches optimize for the same goal. Additionally, some of these approaches provide exogenous information (e.g., signals from news data) to the DRL agent, which makes for a biased comparison with MVO. Additionally, none of these works provide implementation details for the MVO frameworks they used for their comparison. We aim to address these issues by conducting a robust comparison of Deep RL and Mean-Variance Optimization for the Portfolio Allocation problem.

Background

The goal of portfolio optimization is to continuously diversify and reallocate funds across assets with the objective of maximizing realized rewards while simultaneously restraining the risk. In practice, portfolio management often aims to not only maximize risk-adjusted returns but also to perform as consistently as possible over a given time interval (e.g. on a quarterly or yearly basis).

Markowitz introduced the modern portfolio theory (MPT) (Markowitz 1952), a framework that allows an investor to mathematically balance risk tolerance and return expectations to obtain efficiently diversified portfolios. This framework relies on the assumption that a rational investor will prefer a portfolio with less risk for a specified level of return and concludes that risk can be reduced by diversifying a portfolio. In this section, we will introduce Mean Variance Optimization (MVO) – one of the main techniques of the MPT – which we later compare to the performance of our DRL framework. Additionally, we introduce RL preliminaries, describing the technique independent of portfolio optimization.

Mean-Variance Portfolio Optimization

Mean-Variance Optimization (MVO) is the mathematical process of allocating capital across a portfolio of assets (optimizing portfolio weights) to achieve a desired investment goal, usually: 1. Maximize returns for a given level of risk, 2. Achieve a desired rate of return while minimizing risk, or 3. Maximize returns generated *per* unit risk. Risk is usually measured by the volatility of a portfolio (or asset), which is the variance of its rate of return. For a given set of assets, this process requires as inputs the rates of returns for each asset, along with their covariances. As the true asset returns are unknown, in practice, these are estimated or forecasted using various techniques that leverage historical data.

This task is then framed as an optimization problem, single or multi-objective, which can be solved in a variety of ways (Cornuejols and Tütüncü 2006; Kalayci et al. 2017; Ghahtarani, Saif, and Ghasemi 2022).

A typical procedure is to solve it as a convex optimization problem and generate an efficient frontier of portfolios such that no portfolio can be improved without sacrificing some measure of performance (e.g., returns, risk). Let w be the weight vector for a set of assets, μ be the expected returns, the portfolio risk can be described as $w^T\Sigma w$, for covariance matrix Σ. To achieve a desired rate of return μ^*, we can solve the portfolio optimization problem:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & w^T\Sigma w \\ \text{subject to} \quad & w^T\mu \geq \mu^*, \\ & w_i \geq 0, \\ & \sum w_i = 1 \end{aligned}$$

Varying μ^* gives us the aforementioned efficient frontier.

Another common objective is the Sharpe Ratio (Sharpe 1998; Chen, He, and Zhang 2011), which measures the return per unit risk. Formally, for portfolio p, the Sharpe Ratio is defined as:

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

where R_p are the returns of the portfolio, σ_p is the standard deviation of these returns, and R_f is a constant risk-free rate (e.g., US Treasuries, approximated by 0.0% in recent history). Although tricky to solve in its direct form –

$$\max_w \frac{\mu^T w - R_f}{(w^T\Sigma w)^{1/2}}$$

– it can be framed as a convex optimization problem through the use of a variable substitution (Cornuejols and Tütüncü 2006). We choose the Sharpe Ratio as our desired objective function for this study as we can optimize for risk-adjusted returns without having to specify explicit figures for minimum expected returns or maximum risk tolerance.

Reinforcement Learning

Reinforcement Learning (RL) is a sub-field of machine learning that refers to a class of techniques that involve learning by optimizing long-term reward sequences obtained by interactions with an environment (Sutton and Barto 2018). An environment is typically formalized by means of a Markov Decision Process (MDP). An MDP consists of a a 5-tuple (S, A, P_a, R_a, γ), where:

- S is a set of states
- A is a set of actions
- $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$
- $R_a(s, s')$ is the immediate reward received after transitioning from state s to state s', due to action a
- γ is a discount factor between $[0, 1]$ that represents the difference in importance between present and future rewards

A solution to an MDP is a policy π that specifies the action $\pi(s)$ that the decision maker will choose when in state s. The objective is to choose a policy π that will maximize the expected discounted sum of rewards over a potentially infinite horizon:

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

The field of Deep Reinforcement Learning (DRL) leverages the advancements in Deep Learning by using Neural Networks as function approximators to estimate state-action value functions, or to learn policy mappings π. These techniques have seen tremendous success in game-playing, robotics, continuous control, and finance (Mnih et al. 2013; Berner et al. 2019; Nguyen and La 2019; Hambly, Xu, and Yang 2021; Charpentier, Elie, and Remlinger 2021).

RL for Portfolio Allocation Given its success in stochastic control problems, RL extends nicely to the problem of portfolio optimization. Therefore, it is not surprising that the use of DRL to perform tasks such as trading and portfolio optimization has received a lot of attention lately. Recent methods focus on learning deep features and state representations, for example, through the use of embedding features derived from deep neural networks such as autoencoders and LSTM models. These embeddings capture price related features which can range from technical indicators (Wang et al. 2019; Soleymani and Paquet 2020; Wang et al. 2021), to information extracted from news in order to account for price fluctuations (Ye et al. 2020). Other proposed features use attention networks or graph structures (Wang et al. 2021, 2019) to perform cross-asset interrelationship feature extraction.

Problem Setup

We frame the portfolio optimization problem in the RL setting. As described in the Background section, RL entails learning in a framework with interactions between an agent and an environment. For the portfolio optimization setting, we create an environment that simulates the US Equities market (using market data replay), and create observation states derived from the observed asset prices. The agent's actions output a set of a portfolio weights, which are used to rebalance the portfolio at each timestep.

Actions

For portfolio allocation over N assets, an agent selects portfolio weights $\boldsymbol{w} = [w_1, \ldots, w_n]$ such that $\sum_{i=1}^{N} w_i = 1$, where $0 \leq w_i \leq 1$. An asset weight of 0 indicates zero holdings of a particular asset in a portfolio, whereas a weight of 1 means the entire portfolio is concentrated in said asset. In extensions of this framework, $w_i < 0$ would allow for shorting an asset, whereas $w_i > 1$ indicates a leveraged position. However, for our case, we restrict actions to non-leveraged long-only positions. These constraints can be enforced by applying the softmax function to an agent's continuous actions.

States

An asset's price at time t is denoted by P_t. The one-period simple return is defined as $R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$. Consequently, the one-period gross return can be defined as $\frac{P_t}{P_{t-1}} = R_t + 1$. Further, we can define the one-period log return as $r_t = log(\frac{P_t}{P_{t-1}}) = log(R_t + 1)$. For our setting, we choose the time period to be daily, and therefore calculate daily log returns using end-of-day close prices. An asset's log returns over a lookback period T can then be captured as $\boldsymbol{r_t} = [r_{t-1}, r_{t-2}, \ldots, r_{t-T+1}]$. In our case, the lookback period is $T = 60$ days.

For a selection of $n + 1$ assets - n securities and cash (denoted by c) - we form the agent's observation state at time t, $\boldsymbol{S_t}$ as a $[(n + 1) \times T]$ matrix:

$$S_t = \begin{bmatrix} w_1 & r_{1,t-1} & \cdots & r_{1,t-T+1} \\ w_2 & r_{2,t-1} & \cdots & r_{2,t-T+1} \\ \vdots & & \ddots & \vdots \\ w_n & r_{n,t-1} & \cdots & r_{n,t-T+1} \\ w_c & vol_{20} & \frac{vol_{20}}{vol_{60}} & VIX_t \ldots \end{bmatrix}$$

The first column is the agent's portfolio allocation vector $\boldsymbol{w}$ as it enters timestep t. This might differ slightly from the portfolio weights it chooses at the timestep before, as we convert the continuous weights into an actual allocation (whole shares only), and rebalance the allocation such that it sums to 1.

For each non-cash asset, we include the log returns over T. These are represented by the vector $[r_{n,t-1}, \ldots, r_{n,t-T+1}]$ for asset n in the state matrix above. Additionally, in the last row, we include three market volatility indicators at time t: vol_{20}, $\frac{vol_{20}}{vol_{60}}$, *VIX*, which we describe in detail in the Experiments section.

Reward

Rather than maximizing returns, most modern portfolio managers attempt to maximize risk-adjusted returns. Since we wish to utilize DRL for portfolio allocation, we want a reward function that helps optimize for risk-adjust returns. The Sharpe ratio is the most widely-used measure for this, however, it is inappropriate for online learning settings as it is defined over a period of time T. To combat this, we use the Differential Sharpe Ratio D_t (Moody et al. 1998) which represents the risk-adjusted returns at each timestep t and has been found yield more consistent returns than maximizing profit (Moody and Saffell 2001; Dempster and Leemans 2006). Therefore, an agent that aims to maximize its future Differential Sharpe rewards learns how to optimize for risk adjusted returns.

We can define the Sharpe Ratio over a period of t returns R_t, in terms of estimates of the first and second moments of the returns' distributions:

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

with

$$A_t = \frac{1}{t}\sum_{i=1}^{t} R_i \text{ and } B_t = \frac{1}{t}\sum_{i=1}^{t} R_i^2 \text{ , } K_t = (\frac{t}{t-1})^{1/2}$$

where K_t is a normalizing factor.

A and B can be recursively estimated as exponential moving averages of the returns and standard deviation of returns on time scale η^{-1}. We can obtain a *differential* Sharpe ratio D_t by expanding S_t to first order in η:

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

Where Differential Sharpe Ratio D_t:

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

with

$$A_t = A_{t-1} + \eta \Delta A_t$$

$$B_t = B_{t-1} + \eta \Delta B_t$$

$$\Delta A_t = R_t - A_{t-1}$$

$$\Delta B_t = R_t^2 - B_{t-1}$$

initialized with $A_0 = B_0 = 0$. We pick $\eta \approx \frac{1}{252}$ (a year has approximately 252 trading days).

Learning Algorithm

RL algorithms can be mainly divided into two categories, model-based and model-free, depending whether the agent has access to or has to learn a model of the environment. Model-free algorithms seek to learn the outcomes of their actions through collecting experience via algorithms such as Policy Gradient, Q-Learning, etc. Such an algorithm will try an action multiple times and adjust its policy (its strategy) based on the outcomes of its action in order optimize rewards.

Policy Optimization Policy optimization methods are centered around the policy $\pi_\theta(a|s)$ which is the function that maps the agent's state s to the distribution of its next action a. These methods optimize the parameters θ either by gradient ascent on the performance objective $J(\pi_\theta)$ or by maximizing local approximations of $J(\pi_\theta)$. This optimization is almost always performed on-policy since the experiences are collected using the latest learned policy, and then using that experience to improve the policy. Some examples of popular policy optimization methods are A2C/A3C (Mnih et al. 2016) and PPO (Schulman et al. 2017). For our experiments we use PPO.

RL Environment Specifics

The environment serves as a wrapper for the market, sliding over historical data in an approach called market replay. It also serves as a broker and exchange; at every timestep, it processes the agents' actions and rebalances the portfolio using the latest prices and the given allocation. As the day shifts and new prices are received, it communicates these to the agent as observations, along with the Differential Sharpe reward. For the purposes of this study, we assume that there are no transaction costs in the environment, and we allow for immediate rebalancing of the portfolio.

At the beginning of each timestep t, the environment calculates the current portfolio value:

$$\text{port\_val}_t = \sum P_{i,t} * \text{shares}_{i,t-1} + c_{t-1}$$

In the above expression, P_i is the price of index i at day t, $\text{shares}_{i,t-1}$ are the index shares at $t-1$, and c_{t-1} is the amount of cash at $t-1$.

In order to calculate $\text{shares}_{i,t}$ and c_t, the environment allocates port_val_t to the indices and cash according to the new weights w_i. Next, it rebalances the portfolio weights w_i to w_{i_reb} by multiplying w_i with the current portfolio value, rounding down the number of shares and converting the remaining shares into cash.

After rebalancing, the environment creates the next state S_{t+1} and proceeds to the next timestep $t+1$. It calculates the new portfolio value based on P_{t+1} and computes the reward $R_t = D_t$ which it returns to the agent.

Experiments

Data & Features

For our experiments, we use daily adjusted close price data of the S&P500 sector indices as shown in Figure 1, the VIX index and the S&P500 index between 2006 and 2021 (inclusive), extracted from Yahoo Finance. The price data is used to compute log returns, as described in a previous section.

To capture market regime, we compute three volatility metrics from the S&P500 index. The first one, vol_{20}, is the 20-day rolling window standard deviation of the daily S&P500 index returns, the second, vol_{60}, is the 60-day rolling window standard deviation of the daily S&P500 index returns and the third is the ratio of these two $\frac{vol_{20}}{vol_{60}}$. This ratio indicates the short-term versus the long-term volatility trend. If $\frac{vol_{20}}{vol_{60}} > 1$, that indicates that the past 20-day

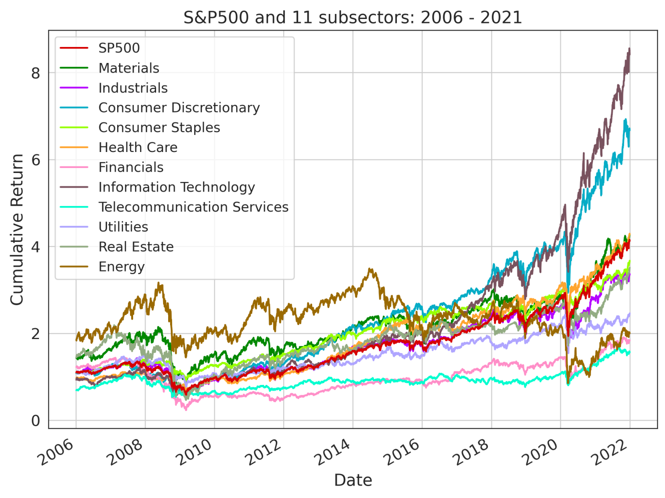

Figure 1: S&P500 and its 11 sector indices between 2006 and 2021.

daily returns of the S&P500 have been more volatile than the past 60-day daily returns, which might indicate a movement from lower volatility to a higher volatility regime (and vice versa). We use the first and third metrics in the observation matrix, along with the value of the VIX index. These values are standardized by subtracting the mean and dividing by the standard deviation, where the mean and standard deviation are estimated using an expanding lookback window to prevent information leakage.

Deep RL Approach

Training Process Although financial data is notoriously scarce (atleast on the daily scale), we want to test the DRL framework across multiple years (backtests). Additionally, financial timeseries exhibit non-stationarity (Cont 2001); this can be tackled by retraining or fine-tuning models by utilizing the most recently available data. In light of these stylized facts, we devise our experiment framework as follows:

The data is split into 10 sliding window groups (shifted by 1-year). Each group contains 7 years worth of data, the first 5 years are used for training, the next 1 year is a burn year used for training validation, and the last year is kept out-of-sample for backtesting.

During the first round of training, we initialize 5 agents (different seeds) with the hyperparameters described in the following section. All five agents start training on data from $[2006 - 2011)$ and their performance is periodically evaluated using the validation period 2011. At the end of the first round of training, we save the best performing agent (based on highest mean episode validation reward). The final year (2012) is kept held-out for backtesting.

This agent is used as a seed policy for the next group of 5 agents in the following training window $[2007 - 2012)$, validation year 2012 and testing year 2013, where this experiment is repeated. This process continues till we reach the final validation period of 2020, generating a total of 50 agents (10 periods x 5 agents), and 10 corresponding backtests (described in a following section).

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

Table 1: Hyperparameters used for PPO.

PPO Implementation & Hyperparameters We use the StableBaselines3 (Raffin et al. 2021) implementation of PPO, and report the hyperparameters used in Table 1. These were picked based on empirical studies (Henderson et al. 2018; Engstrom et al. 2019; Rao et al. 2020), as well as a coarse grid search over held-out validation data.

Additionally, we make use of the Vectorized SubProcVecEnv environment wrappers provided by StableBaselines3 to collect experience rollouts through multiprocessing across independent instances of our environment. Therefore, instead of training the DRL agent on one environment per step, we trained our model on $n\_envs = 10$ environments per step in order to gain more diverse experience and speed up training.

Each round of training lasted a total 7.5M timesteps so as to have approximately 600 episodes per round per environment: (252 trading days per yr $\times$ 5 yrs per round) $\times$ (10 environments) $\times$ (600 episodes) $\approx$ 7.5M timesteps. The rollout buffer size was set to $n\_steps = 252 \times 3 \times n\_envs$ so as to collect sufficient experiences across environments. We set up the learning rate as a decaying function of the current progress remaining, starting from $3e-4$, annealed to a final value of $1e-5$. We used a $batch\_size$ of $1260 = (252 \times 5)$, set the number of epochs when optimizing the surrogate loss to $n\_epochs = 16$, picked the discount factor $\gamma = 0.9$, set the bias-variance trade-off factor for Generalized Advantage Estimator $gae\_lambda = 0.9$ and $clip\_range = 0.25$. Additionally, we use a $[64, 64]$ fully-connected architecture with $tanh$ activations, and intiailize the policy with a log standard deviation $log\_std\_init = -1$.

Mean-Variance Optimization Approach

As we wish to compare the model-free DRL approach with MVO, we equalize the training and operational conditions. For training, the MVO approach uses a 60-day lookback period (same as DRL) to estimate the means and covariances of assets. Asset means are simply the sample means over the lookback period. However, we do not directly use the sample covariance, as this has been shown to be subject to estimation error that is incompatible with MVO. To tackle this, we make use of the Ledoit-Wolf Shrinkage operator (Ledoit and Wolf 2004). Additionally, we enforce non-singular and positive-semi-definite conditions on the covariance matri-

Metric	DRL	MVO
Annual return	0.1211	0.0653
Cumulative returns	0.1195	0.0650
Annual volatility	0.1249	0.1460
Sharpe ratio	1.1662	0.6776
Calmar ratio	2.3133	1.1608
Stability	0.6234	0.4841
Max drawdown	-0.3296	-0.3303
Omega ratio	1.2360	1.1315
Sortino ratio	1.7208	1.0060
Skew	-0.4063	-0.3328
Kurtosis	2.7054	2.6801
Tail ratio	1.0423	0.9448
Daily value at risk	-0.0152	-0.0181

Table 2: Statistics for the DRL and MVO approaches. All metrics are averaged across 10 backtests (backtesting period: $[2012 - 2021]$), except Max Drawdown which is reported as the maximum seen in any period.

ces, setting negative eigenvalues to 0, and then rebuilding the non-compliant matrices.

Given the estimated means and covariances for a lookback period, we then optimize for the Sharpe Maximization problem and obtain the weights at every timestep. We use the implementation in PyPortfolioOpt (Martin 2021) to aid us with this process.

Evaluation & Backtesting

We evaluate the performance of both techniques through 10 independent backtests $[2012 - 2021]$. Both strategies start each backtest period with an all cash portfolio allocation of $\$100,000$. Then, the strategies trade daily using the portfolio weights obtained by each method, enforcing for weight constraints $\sum w = 1, 0 \leq w_i \leq 1$, and ensuring only whole number of shares are purchased. By doing so, we can obtain daily portfolio values (and returns), which we subsequently use to compute the statistics we will discuss in the Results section. These are computed with the aid of the Python library Pyfolio.

DRL Agent: We evaluate the trained PPO agents in deterministic mode. For each backtest, the agent used has a gap burn year between the last day seen in training and the backtest period. For example, a DRL backtest carried out in 2012 would use an agent trained in $[2006-2011)$, with 2011 being the burn year.

MVO: As the MVO approach does not require any training, it simply uses the past 60-day lookback period before any given day to calculate portfolio weights. For example, a MVO backtest starting January 2012 will use data starting October 2011 (this 60-day window shifts with each day).

Results

Figure 2 illustrates the performance metrics obtained by applying the aforementioned backtest process on all testing periods [2012-2021]. The DRL agent outperforms the MVO portfolio by exhibiting higher Sharpe and lower yearly maximum drawdowns in virtually every year throughout the backtest period (see Figure 1). It also outperforms the MVO portfolio in terms of having marginally lower maximum drawdown.

To compare overall performance on the entire backtest period between the two methods, we compute the average performance across all 10 backtest periods. For DRL, we average the performance across the 5 agents (each trained on a different seed) for each year and then average performance across all backtest periods. Similarly, for MVO, we average its performance across all 10 years. By looking at Table 2 we observe that DRL annual returns and Sharpe ratio are 1.85x higher than those of the MVO portfolio. The DRL strategy's Sharpe throughout the whole backtest period is 1.16x compared to 0.66x for MVO.

Figure 3a) and Figure 4a) plot the monthly returns over all backtest periods for the two methods. It is evident that DRL is experiencing more steady returns month-to-month than MVO. On the other hand, MVO swings between periods of high returns to periods of low returns a lot more frequently without a steady positive return trajectory. Similarly, in Figure 3b) and Figure 3b), we plot the annual returns for the two methods. The vertical dashed line indicates the average annual return across the 10 backtests. For DRL we observe positive returns for almost all backtest years which is a lot more consistent than the behavior of MVO's annual returns. Figure 3c) and Figure 3c) plot the distribution of monthly returns averaged across all months. The DRL monthly returns distribution has a lower standard deviation and and spread than MVO and a positive mean.

Further, we compute the daily portfolio change for each strategy by measuring the change in its portfolio weights. Δp_w is the absolute value of the element-wise difference between two allocations (ignoring the cash component). As buying and selling are treated as individual transactions, $\Delta p_w \in [0.0, 2.0]$. For example, take a case where the portfolio at time $t-1$ is concentrated in non-cash asset A, and at time t is entirely concentrated in non-cash asset B. This requires selling all holdings of A, and acquiring the equivalent shares in B, leading to $\Delta p_w = 2.0$.

Using metric Δp_w, we observe that the Reinforcement Learning strategy has less frequent changes to its portfolio. In practice, this would result in lower average transaction costs. In particular, the average change in portfolio composition is nearly double for Mean-Variance portfolio compared to the DRL strategy during market downturn in March 2020, as shown in Figure 2, when trading conditions were particularly challenging (i.e. significantly lower market liquidity and elevated bid/ask spreads). Finally, the DRL strategy's performance is derived from the average of five individual agents initialized with different seeds, providing additional regularization which is likely to result in a more stable out-of-sample strategy compared to the MVO strategy.

Conclusion

We highlight our key contributions as follows:

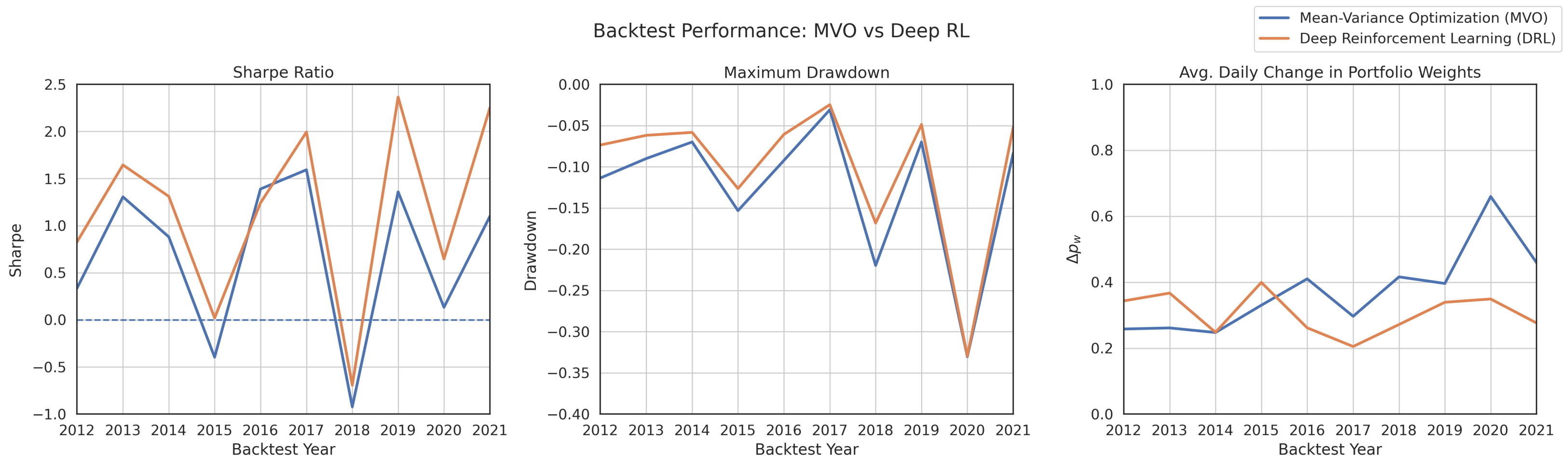

Figure 2: Backtest Results: MVO vs DRL Portfolio Allocation.

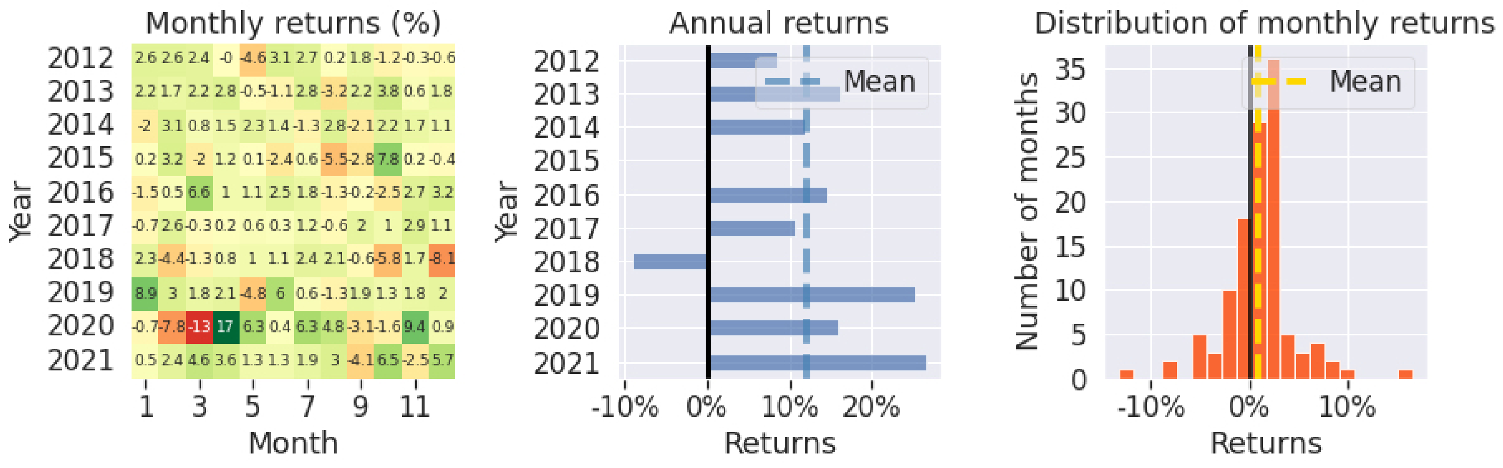

Figure 3: a) DRL Monthly Returns b) DRL Annual Returns c) DRL Monthly Distribution of Returns.

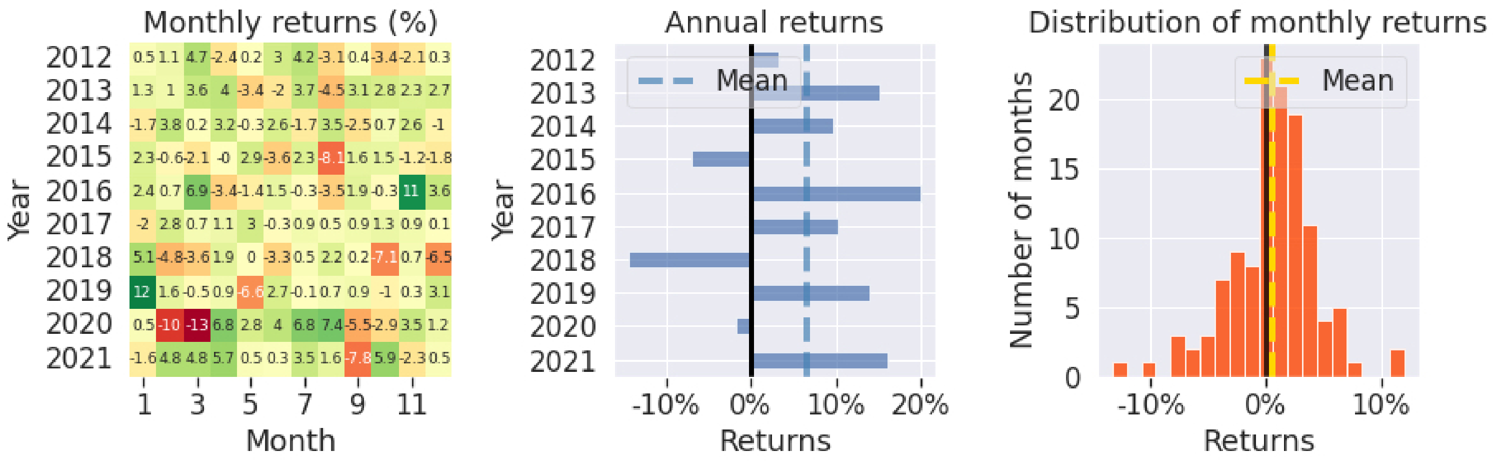

Figure 4: a) MVO Monthly Returns b) MVO Annual Returns c) MVO Monthly Distribution of Returns.

- We have designed a simple environment that serves as a wrapper for the market, sliding over historical data using market replay. The environment can allocate multiple assets and can be easily modified to reflect transaction costs.
- We compare our framework's performance during ten backtest experiments over different periods for the US Equities Market using S&P500 Sector indices. Our experiments demonstrate the improved performance of the deep reinforcement learning framework over Mean-Variance portfolio optimization.
- The profitability of the framework surpasses MVO in terms of Annual Returns, Sharpe ratio and Maximum Drawdown. Additionally, we observe that DRL strategy leads to more consistent returns and more stable portfolios with decreased turnover. This has implications for live-deployment, where transaction costs and slippage affect P&L.

Future Work

In our future work, we would like to model transaction costs and slippage either by explicitly calculating them during asset reallocation or as a penalty term to our reward. Moreover, we would like to explore adding a drawdown minimization component to our reward that will potentially help the agent learn a more consistent trading strategy.

Another area of exploration is training a regime switching model which will balance its funds amongst two agents depending on market volatility (low vs high). One of them will be a low-volatility trained agent and the other a high volatility trained agent. We would like to compare performance between our current implicit regime parametrization and an explicit one. Further exploration of these research directions may produce significant insights into practical trading behavior.

References

Benhamou, E.; Saltiel, D.; Ungari, S.; and Mukhopadhyay, A. 2020a. Bridging the gap between Markowitz planning and deep reinforcement learning. *arXiv preprint arXiv:2010.09108*.

Benhamou, E.; Saltiel, D.; Ungari, S.; and Mukhopadhyay, A. 2020b. Time your hedge with deep reinforcement learning. *arXiv preprint arXiv:2009.14136*.

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; 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.; Chen, J.; Hull, J.; and Poulos, Z. 2021. Deep hedging of derivatives using reinforcement learning. *The Journal of Financial Data Science*, 3(1): 10–27.

Charpentier, A.; Elie, R.; and Remlinger, C. 2021. Reinforcement learning in economics and finance. *Computational Economics*, 1–38.

Chen, L.; He, S.; and Zhang, S. 2011. When all risk-adjusted performance measures are the same: In praise of the Sharpe ratio. *Quantitative Finance*, 11(10): 1439–1447.

Cong, L. W.; Tang, K.; Wang, J.; and Zhang, Y. 2021. AlphaPortfolio: Direct construction through deep reinforcement learning and interpretable AI. *Available at SSRN*, 3554486.

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

Cornuejols, G.; and Tütüncü, R. 2006. *Optimization methods in finance*, volume 5. Cambridge University Press.

Dempster, M. A.; 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.

Du, J.; Jin, M.; Kolm, P. N.; Ritter, G.; Wang, Y.; and Zhang, B. 2020. Deep reinforcement learning for option replication and hedging. *The Journal of Financial Data Science*, 2(4): 44–57.

Du, X.; and Tanaka-Ishii, K. 2020. Stock embeddings acquired from news articles and price history, and an application to portfolio optimization. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, 3353–3363.

Engstrom, L.; Ilyas, A.; Santurkar, S.; Tsipras, D.; Janoos, F.; Rudolph, L.; and Madry, A. 2019. Implementation matters in deep rl: A case study on ppo and trpo. In *International conference on learning representations*.

Ghahtarani, A.; Saif, A.; and Ghasemi, A. 2022. Robust portfolio selection problems: a comprehensive review. *Operational Research*, 1–62.

Hambly, B.; Xu, R.; and Yang, H. 2021. Recent advances in reinforcement learning in finance. *arXiv preprint arXiv:2112.04553*.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

i Alonso, M. N.; Srivastava, S.; et al. 2020. Deep reinforcement learning for asset allocation in us equities. Technical report.

Jiang, Z.; and Liang, J. 2017. Cryptocurrency portfolio management with deep reinforcement learning. In *2017 Intelligent Systems Conference (IntelliSys)*, 905–913. IEEE.

Kalayci, C.; Ertenlice, O.; Akyer, H.; and Aygören, H. 2017. A review on the current applications of genetic algorithms in mean-variance portfolio optimization. *Pamukkale University Journal of Engineering Sciences*, 23: 470–476.

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

Ledoit, O.; and Wolf, M. 2004. Honey, I shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4): 110–119.

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

Li, X.; Li, Y.; Zhan, Y.; and Liu, X.-Y. 2019. Optimistic bull or pessimistic bear: Adaptive deep reinforcement learning for stock portfolio allocation. *arXiv preprint arXiv:1907.01503*.

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

Lima Paiva, F. C.; Felizardo, L. K.; Bianchi, R. A. d. C.; and Costa, A. H. R. 2021. Intelligent trading systems: a sentiment-aware reinforcement learning approach. In *Proceedings of the Second ACM International Conference on AI in Finance*, 1–9.

Liu, X.-Y.; Yang, H.; Chen, Q.; Zhang, R.; Yang, L.; Xiao, B.; and Wang, C. D. 2020. FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*.

Lu, D. W. 2017. Agent inspired trading using recurrent reinforcement learning and lstm neural networks. *arXiv preprint arXiv:1707.07338*.

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

Martin, R. A. 2021. PyPortfolioOpt: portfolio optimization in Python. *Journal of Open Source Software*, 6(61): 3066.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937. PMLR.

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*.

Moody, J.; and Saffell, M. 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4): 875–889.

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

Nguyen, H.; and La, H. 2019. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 590–595. IEEE.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268): 1–8.

Rao, N.; Aljalbout, E.; Sauer, A.; and Haddadin, S. 2020. How to make deep RL work in practice. *arXiv preprint arXiv:2010.13083*.

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. *Streetwise–the Best of the Journal of Portfolio Management*, 169–185.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Soleymani, F.; and Paquet, E. 2020. Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath. *Expert Systems with Applications*, 156: 113456.

Su, P.-H.; Gasic, M.; Mrksic, N.; Rojas-Barahona, L.; Ultes, S.; Vandyke, D.; Wen, T.-H.; and Young, S. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669*.

Sun, S.; Wang, R.; He, X.; Zhu, J.; Li, J.; and An, B. 2021. Deepscalper: A risk-aware deep reinforcement learning framework for intraday trading with micro-level market embedding. *arXiv preprint arXiv:2201.09058*.

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

Théate, T.; and Ernst, D. 2021. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173: 114632.

Wang, J.; Zhang, Y.; Tang, K.; Wu, J.; and Xiong, Z. 2019. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 1900–1908.

Wang, Z.; Huang, B.; Tu, S.; Zhang, K.; and Xu, L. 2021. DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1): 643–650.

Wu, X.; Chen, H.; Wang, J.; Troiano, L.; Loia, V.; and Fujita, H. 2020. Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, 538: 142–158.

Yang, H.; Liu, X.-Y.; Zhong, S.; and Walid, A. 2020. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the First ACM International Conference on AI in Finance*, 1–8.

Ye, Y.; Pei, H.; Wang, B.; Chen, P.-Y.; Zhu, Y.; Xiao, J.; and Li, B. 2020. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 1112–1119.

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