Reinforcement Learning applied to dynamic portfolio management: An article review

Mattia Mastrogiovanni*

September 2024

Abstract

This literature review explores trends in applying Reinforcement Learning (RL) for dynamic portfolio management, highlighting the evolution from traditional Modern Portfolio Theory (MPT) to machine learning-driven methods. By framing portfolio optimization as a Markov Decision Process (MDP), RL techniques demonstrate adaptability to market uncertainties. The study examines prominent RL models, including Deep Deterministic Policy Gradient (DDPG), Policy Gradient (PG), and Proximal Policy Optimization (PPO), focusing on their effectiveness in maximizing risk-adjusted returns. Challenges such as data quality, model evaluation, and market frictions are also discussed. The analysis aims to provide insights into RL's potential and limitations in enhancing financial decision-making.

*for any additional information and/or details please write to mattia.mastrogiovann@libero.it

1 Introduction

Portfolio management has evolved significantly with the integration of machine learning, particularly through Reinforcement Learning (RL). Traditional approaches like Modern Portfolio Theory (MPT) rely on mean-variance optimization to balance risk and return. However, they often lack the adaptability required in volatile markets. RL, on the other hand, reframes portfolio management as a dynamic decision-making process using frameworks like Markov Decision Processes (MDPs) or Partially Observed Markov Decision Processes (POMDPs).

This approach allows an RL agent to continuously interact with the market, making decisions to maximize long-term returns based on evolving data. Essential components of RL models, such as state and action spaces, reward functions, and policy optimization techniques, play a crucial role in this adaptability. Models like DDPG, PG, and PPO stand out for their application in continuous action spaces and their potential in handling the complexity of financial markets.

Furthermore, the review addresses real-world challenges, including transaction costs, liquidity issues, and the need for robust backtesting. By exploring these aspects, this analysis aims to clarify RL's advantages over traditional methods and outline the practical considerations required for its implementation in live trading scenarios.

2 Problem setup

2.1 Modern Portfolio Theory

The problem of portfolio optimization is a classic maximization or minimization problem. 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. It has been proved that investing in a set of assets, instead in only one, allows the investors to reduce the expected portfolio's volatility.

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.

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).

2.2 Reinforcement Learning theory

All of the authors interpret the portfolio optimization's problem as a Markov Decision Process (MDP) or more broadly as a Partially Observed Markov Decision Process (POMDP).

An **MDP** is defined by the tuple (S, A, P, R, γ):

- S: Set of possible states.
- A: Set of possible actions.
- P: Transition probability function $P(s'|s, a)$ that defines the probability of transitioning to state s' when action a is taken in state s.
- R: Reward function $R(s, a, s')$ that defines the immediate reward obtained after transitioning from state s to state s' by taking action a.
- γ: Discount factor, which determines the importance of future rewards.

A **POMDP** is an extension of the MDP, defined by the tuple (S, A, O, P, R, γ):

- S, A, P, R, and γ have the same meaning as in the MDP.
- O: Set of possible observations. It represents the set of observations that the agent can make in each state.

In an MDP, the portfolio manager has full knowledge of the state of the environment at each time step, including factors such as current asset prices and portfolio composition, allowing for direct observation of the environment's state. Conversely, in a POMDP, the portfolio manager does not have complete knowledge of the environment's state but receives observations that are probabilistically

related to the true state. This lack of complete information adds complexity to decision-making, as the portfolio manager must infer the true state of the environment based on the received observations.

The first two fundamental elements of an RL algorithm are the **agent** and the **environment**. A learning agent must be able to sense the state of its environment to some extent and must be able to take actions that act the state. The agent also must have a goal or goals relating to the state of the environment. Markov decision processes are intended to include just these three aspects—sensation, action, and goal—in their simplest possible forms without trivializing any of them. Any method that is well suited to solving such problems we consider to be a reinforcement learning method. Reinforcement learning is different from supervised learning, the kind of learning studied in most current research in the field of machine learning. Supervised learning is learning from a training set of labeled examples provided by a knowledgeable external supervisor. In uncharted territory—where one would expect learning to be most beneficial—an agent must be able to learn from its own experience. Reinforcement learning is also different from what machine learning researchers call unsupervised learning, which is typically about finding structure hidden in collections of unlabeled data. The terms supervised learning and unsupervised learning would seem to exhaustively classify machine learning paradigms, but they do not. there are other features that deserve to be described:

1. **policy** is a mapping from perceived states of the environment to actions to be taken when in those states;

2. **reward signal** defines the goal of a reinforcement learning problem. On each time step, the environment sends to the reinforcement learning agent a single number called the reward. The agent's sole objective is to maximize the total reward it receives over the long run.

3. **value function** specifies what is good in the long run. Roughly speaking, the value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. Rewards are in a sense primary, whereas values, as predictions of rewards, are secondary.

4. and optionally a **model of the environment** is something that mimics the behavior of the environment, or more generally, that allows inferences to be made about how the environment will behave. (Sutton and Barto, 2012)

Hence, after we have defined the state space (e.g. portfolio weights, historical prices of assets, financial indicators, market trends...), the action space (e.g. adjusting portfolio weights...) and the reward function (e.g. maximizing sharpe ratio), the agent needs to solve the **Bellman** equation.

In reinforcement learning, the Bellman equation helps to recursively decompose the value function $V(s)$, which estimates the optimal future rewards:

$$V(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right)$$

where:

- $R(s,a)$ is the reward received after taking action a in state s.
- γ is the discount factor, determining the importance of future rewards.
- $P(s'|s,a)$ is the probability of reaching state s' from state s by taking action a.

The optimal policy π^* dictates the best action to take in any given state to maximize the cumulative reward. Using methods like Q-learning, it is possible to approximate the action-value function $Q(s,a)$, which estimates the expected utility of taking action a in state s, and thereafter following the optimal policy:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

An example to implement these concepts could be a deep reinforcement learning model where deep neural networks approximate the Q function (Deep Q-Networks, DQN) or policy (Policy Gradient methods, Actor-Critic models). These models can handle the high dimensionality and complexity of the state and action spaces in financial markets. Anyway, pros and cons of the different models for RL are discussed in Section 3.2.

The last step in training the model using historical market data, iteratively updating policies based on received rewards and revising the value or action-value estimates. Evaluation should be done in both simulated environments and, cautiously, in live markets with small stakes to ensure that the model generalizes well across different market conditions.

Portfolio management environments are inherently stochastic due to the unpredictable nature of market movements. Robust RL solutions account for this by considering various possible future states in the value function estimation, enabling the model to learn policies that are not only optimal on average but also risk-aware.

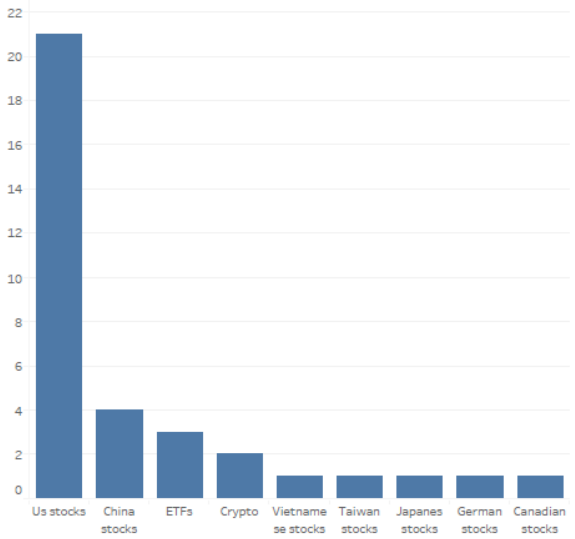

Figure 1: Dataset used for each paper

3 Model's architecture

In this section is presented a detailed analysis concerning all the main features of an RL model based on the papers' research.

3.1 Dataset

This complex model needs high quality data and a properly feature engineering, in order to perform well. To train and validate this specific kind of algorithm, there is a solid trend to use the U.S. securities market prices as a dataset. A reason can be found in the fact that the U.S. market is the most developed in the world, according to the famous statement: *"Prices are originated in the United States"*. The more efficient a market is, the more difficult it is to profit from temporary misalignment. As it is showed in the Figure 1, the preference to use the U.S. stocks as dataset is evident. There is also relevant interest in the Chinese market; thanks to its rapid growth it is increasingly becoming the focus of the research. ETFs and cryptocurrencies are also of considerable interest, because while the former allow for high diversification without the need for further techniques, the latter represent a high volatile market.

There are some small differences between the market used as dataset. However almost all of the authors extract the same information from the market. The most common process is to download **OHLCV** (Open, High, Low, Close, Volume) data. These data represent all the features that guide the price's movement in the market. In this way the agent is able to discover hidden pattern between the stocks and the variables presented. Anyway, in order to have a better representation of the market, prices are not enough. For instance some authors consider 12 U.S. stocks as a good representation of market's universe, this is an arguably consideration. That's why many studies add to OHLCV data other variables (technical indicators and fundamental variables).

Technical indicators describe the current state of the market price and incorporate information about its past trends. An indicator can be seen as a snapshot of the current situation that accounts for past behavior over a certain period. To make technical indicators, it is pivotal to choose which indicators to use and how far back in the past the indicators impact better predictions of future price movements. We use three leading technical indicators based on the literature: moving average (MA), relative strength index (RSI), and moving average convergence divergence (MACD) (Kara, Boyacioglu, & Baykan, 2011). MA is a trend-following technical indicator that computes the average price over a window. RSI is an oscillator that shows the strength or pressure of an asset price movement by comparing the upward or downward movement of an asset's close price, where EMA is an exponential moving average that computes MA with exponential decay weight. MACD is a trend-following technical indicator that shows the relationship between the two MAs (Jang and Seong, 2023).

Economic indicators are financial time series concerning employment, inflation, growth, consumer, productivity, and sentiment indicators. These indicators can improve the quality of predictability's power of the agent, because they are indispensable to understand the market's sentiment. One relevant implementation is found in a published of Sang, Wei and Yang in 2023. Furthermore, incorporating accounting indicators and ratios into the dataset can greatly enhance its decision-making process for investing. For example, incorporating Earnings Per Share (EPS) and Price-to-Earnings (P/E) ratios could guide the RL agent in identifying potentially undervalued stocks that may offer growth opportunities, as supported by Lev and Thiagarajan (1993) in their study on fundamental information analysis. Similarly, integrating the Debt-to-Equity (D/E) ratio and Return on Equity (ROE) helps evaluate a company's financial health and profitability, aligning with the findings of Fama and French (2002) regarding the predictive power of financial ratios on future returns. Adding liquidity measures like the Current Ratio and performance metrics such as Dividend Yield can further refine investment strategies by balancing risk and reward, as noted by O'Hara (2003) in her analysis on market microstructure and liquidity. By enriching an RL agent's dataset with these comprehensive financial metrics, the agent can be better equipped to execute more nuanced and informed investment strategies that are adaptive to changes in market and company fundamentals.

Once created the dataset, it should be cleaned in order to have high quality data. There are many ways to do so. In an interesting paper, the authors use a lot of data, such as economic indicators and stocks indices over prices and technical indicators (Sun et al., 2024). Since the relevant dimension of dataset can decrease the robustness of the model they used SHAP model (Lundberg & Lee, 2017) to underscore the feature importance. Using this method, from raw 93 time series in daily frequency, the number deceased to only 32 important features as node features.

The last step is to divide the dataset in sub-dataset. Dividing a dataset into training, validation, and testing subsets is crucial in developing robust machine learning models. The training set is used to teach the model, allowing it to learn and adapt its parameters to the data's underlying patterns. The validation set is then used to fine-tune model parameters and prevent over fitting, which occurs when a model learns not just the underlying patterns but also the noise in the training data, leading to poor generalization on new, unseen data. Lastly, the testing set provides an unbiased evaluation of a final model, offering insights into how the model will perform in real-world scenarios. This separation is vital because it helps ensure that the model can generalize well beyond the specific examples in the dataset, maintaining its effectiveness and reliability when making investment decisions in the dynamic and often unpredictable financial markets.

3.2 Model used

Choosing the appropriate model is the most important task in building an RL algorithm. Based on my own research, the most frequent models implemented are DDPG (Deep Deterministic Policy Gradient), PG (Policy Gradient) and PPO (Proximal Policy Optimization), as shown in the Figure 2. Here a brief description of these 3 models:

- DDPG is a reinforcement learning algorithm designed for continuous action spaces. It combines elements of deep learning with deterministic policy gradients to learn complex policies efficiently. DDPG employs an actor-critic architecture where it simultaneously learns a deterministic policy (the actor) and a value function (the critic). The actor network directly maps states to actions, providing a policy that is deterministic and continuous, while the critic network evaluates the quality of the actions chosen by the actor by estimating their expected return. DDPG leverages a replay buffer to store past experiences, enabling it to learn from a diverse set of data samples and improve sample efficiency. Additionally, it employs target networks to stabilize training by periodically updating target values. Through iterative updates using the actor and critic networks, DDPG learns an optimal policy that maximizes the expected cumulative reward in the environment.

- PG in contrast to value-based methods, directly learn the policy function without explicitly estimating the value function. PG algorithms operate by updating the parameters of the policy in the direction that increases the expected return. The basic idea behind PG is to estimate the gradient of the expected return with respect to the policy parameters and then use this gradient to update the policy in a way that improves its performance. PG methods typically use stochastic policies, allowing for exploration of the action space. One common approach is to

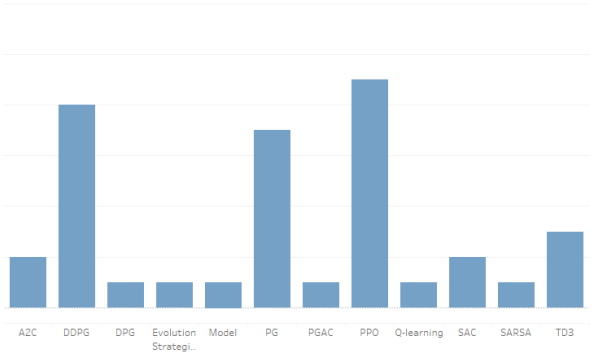

Figure 2: Model used

use the REINFORCE algorithm, where trajectories are sampled from the current policy, and the gradient of the expected return is estimated based on these trajectories using techniques such as the likelihood ratio or score function. By iteratively updating the policy parameters using these gradients, PG methods aim to find a policy that maximizes the expected cumulative reward in the environment.

- PPO is a reinforcement learning algorithm that addresses some limitations of traditional policy gradient methods like instability and inefficiency in utilization of samples. PPO operates by iteratively collecting data from the environment, computing policy updates based on this data, and then applying these updates to the policy. The core idea behind PPO is to constrain the policy update to be close to the previous policy, preventing large policy changes that may lead to instability. This is achieved through a clipped objective function that limits the policy update to a certain range, ensuring that the new policy does not deviate too far from the old one. PPO also utilizes multiple epochs of gradient descent on minibatches of data to further stabilize training. By combining these techniques, PPO effectively balances exploration and exploitation, leading to more stable and efficient learning.

However, other powerful models used in the literature are A2C, TD3 and SAC. The use of model really depends on several factors...

These are basic RL models. Often authors use combinations of them. For instance it is common to build an architecture compose by more then one agent. A relevant example is the student-teacher architecture. The "teacher" agents, each representing different market roles and strategies, interact with the market to gather data and refine their trading strategies. These teachers distill their knowledge and experiences into a simpler form that encapsulates crucial market insights and strategies. The "student" agent learns from this distilled knowledge rather than directly from the raw market data. This allows the student to develop a generalized and effective trading strategy by mimicking the successful behaviors and strategies of the teachers. The student agent takes decisions based on this learned knowledge, applying it to manage a portfolio in a way that is adaptive to new and changing market conditions, aiming to maximize profitability while managing risk effectively (Chen et al., 2023).

3.2.1 Continuous versus discrete actions

As I will describe deeply in the Benchmarks section, the EIIE model computed by Jiang is a cornerstone of RL literature. He used the DQN (Deep Q-Networks) model to optimize a portfolio of cryptocurrencies. Even if the present model is widely cited and implement in various publications, it describes the portfolio optimization problem as a **discrete** problem. Discrete action spaces, exemplified by models like Deep Q-Networks (DQN), involve selecting from a finite set of predefined portfolio allocations. This approach simplifies implementation and reduces computational demands but lacks flexibility, potentially limiting the model's ability to identify optimal allocations beyond the predefined choices. Conversely, **continuous** action spaces, utilized by algorithms such as Deep DDPG and PPO, allow for an infinite range of actions, providing finer granularity in decision-making. This increased flexibility mirrors real-world investment scenarios more closely, enabling more precise and potentially more optimal portfolio weight adjustments. The continuous approach facilitates a smoother learning

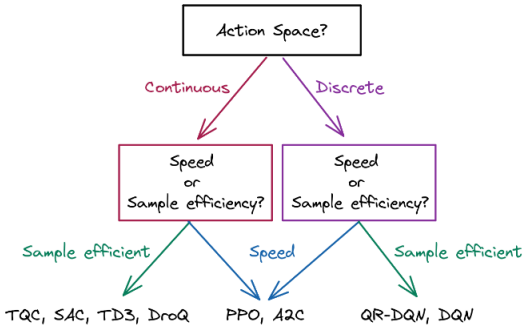

Figure 3: Discrete vs continuos actions

process and a broader exploration of potential solutions, making it a superior choice for developing sophisticated and effective portfolio management strategies. These characteristics make continuous action models better suited for capturing the complex dynamics of financial markets and optimizing portfolio performance effectively. It can be useful a schme visualized in the Figure 3, provided by *Stable Baselines 3 Resources*[1].

3.2.2 Policy

Policy is a fundamental task of the Reinforcement Learning structure. Models can be on policy or off policy. According to the present research, the number of on and off policy models is balanced (almost *fifty/fifty*).

An on-policy algorithm directly optimizes the policy used to collect training data. Popular algorithms include Proximal Policy Optimization (PPO) and Actor-Critic models like A3C. These models aim to improve the policy iteratively based on the agent's current experiences. On-policy algorithms can directly refine the policy through stable updates, which helps maintain consistent behavior. They're particularly effective in environments with relatively stable and predictable states. However, they tend to have a higher sample complexity because they only learn from recent actions under the current policy.

In contrast, off-policy algorithms (like Deep Q-Learning and Deep Deterministic Policy Gradient - DDPG) can learn from experiences generated by a different policy or agent. This allows them to learn from historical data as well as new experiences, improving sample efficiency. Off-policy algorithms can handle more complex, high-dimensional environments because they efficiently leverage past data to optimize policies. But, the greater flexibility comes at the cost of stability, as they are prone to high variance and overestimation errors.

Furthermore, over the choice of on or off-policy, there advanced policy mechanism. Advanced policy mechanisms in reinforcement learning have significantly improved adaptability and efficiency in dynamic portfolio management. Entropy regularization, as utilized in Soft Actor-Critic (SAC) (Haarnoja et al., 2018), incentives exploration by penalizing deterministic actions through an entropy term in the objective function. This ensures policies remain flexible and robust. Policy networks, including Deep Q-Networks (DQN) (Mnih et al., 2015) and Actor-Critic architectures like A3C (Mnih et al., 2016), optimize the policy via neural networks to handle high-dimensional state spaces. Generalized Advantage Estimation (GAE) (Schulman et al., 2016) balances bias and variance by computing a weighted sum of temporal difference errors, enhancing policy gradient stability and making algorithms like Proximal Policy Optimization (PPO) more efficient. Hierarchical Reinforcement Learning (HRL) (Kulkarni et al., 2016) introduces nested sub-policies to handle complex, multi-level decisions, allowing an agent to respond to market trends at both macro and micro levels. Finally, Transfer Learning and Meta-Learning accelerate learning and adaptation across different environments (Taylor & Stone, 2009; Finn et al., 2017). They enable agents to reuse learned policies from one financial market to another or to swiftly adjust strategies for new market conditions. Together, these advanced mechanisms empower

[1]Stable Baselines 3 is a set of improved implementations of Reinforcement Learning (RL) algorithms based on OpenAI Baselines, built in Python

reinforcement learning models to navigate the intricate landscape of portfolio management with greater precision and flexibility.

According to the research (Chung,2023), the off-policy algorithms DDPG, TD3 and SAC are unable to learn the right Q function due to the noisy rewards and therefore perform poorly. The on-policy algorithms PPO and A2C, with the use of generalised advantage estimation (GAE), are able to deal with the noise and derive a close to optimal policy. The clipping variant of PPO is found to be important in preventing the policy from deviating from the optimal once converged.

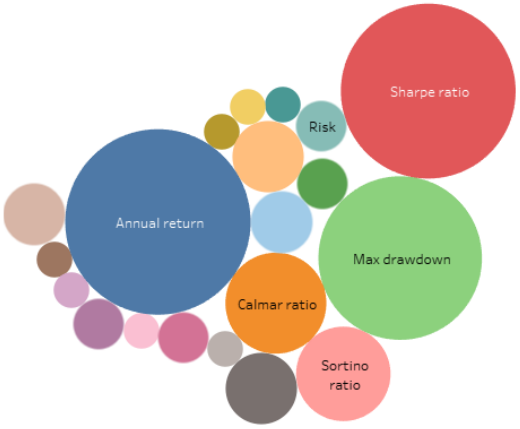

Figure 4: Use of performance measures by dimension

4 Model's evaluation

Usually model's evaluation is biased because some authors compare the results of own model with *"convenient"* indicators and/or benchmarks.

4.1 Performance indicators

As visualized in the Figure 4, the most common performance indicators, implemented in the evaluation, are Annual return (AN), Maximum drawdown (MD) and Sharpe ratio (SR). While AR and MD are indicators respectively related to return and risk, the SR combines both aspects into a single metric.

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, it would be more correct to use the Differential Sharpe Ratio Dt (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. The expression for the differential Sharpe Ratio D_t can be written as[2]:

$$D_t \equiv \frac{B_{t-1}\Delta A_{t-1} - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}.$$

Over these measures there are some specific risk measures. For instance the VaR (Value-at-Risk) to manage the market risk (see Section 6 for more details).

Information Ratio is also a key performance's measure. It is a financial metric that describes an investment's performance beyond the market's returns. It is considered a measure of active management. The Calmar Ratio and Sortino Ratio are two important performance metrics used to evaluate the risk-adjusted returns of an investment. The Calmar Ratio is calculated by dividing the annualized return of an investment by its maximum drawdown, thereby assessing the return per unit of drawdown risk over a specified period. The Sortino Ratio, on the other hand, refines the Sharpe Ratio by considering only downside volatility, which is deemed more relevant for investors concerned about negative returns. It is computed by dividing the excess return of an investment over the risk-free rate by the standard deviation of negative asset returns (downside risk). While both ratios offer valuable insights into an investment's performance under different risk scenarios, they are often considered less comprehensive than metrics mentioned above. This is because the Calmar Ratio and Sortino Ratio focus on specific aspects of risk, potentially overlooking the broader context of market fluctuations

[2]For more details concerning the implementation see the paper published in 2023 by Sood and coauthors.

and overall risk-adjusted performance. Anyway all of these measures can be included in the reward function, building an ad-hoc portfolio for the investor.

Evaluation in the work referenced above typically selects different sets of financial time series and a set of return measures, some risk-adjusted, and does a relative comparison against baselines. As financial markets are recognised to be highly volatile and non-stationary in nature, it is difficult to assess whether an algorithm is truly performing well and able to generalise consistently. In other words, it is difficult to gauge the whether the performance is due to luck or ability. The choice of simulator in this project allows us to derive the optimal policy analytically when assuming zero market impact. When including market impact, the derived optimal policy without market impact serves as an upper bound to measure performance.

4.2 Benchmarks

Since evaluations of existing works often rely on limited metrics, as analyzed in the previous section, many works cannot be reproduced due to data unavailability or lack of experimental information. Then, there is also the difficulty in interpreting the actions taken by the model, caused by the *black box* problem of machine learning. *It's disconcerting how common this phenomenon is.* The financial market, used as the RL agent's environment, has a disproportionate impact on the proposed reward functions and lack of predictability. Furthermore, RL algorithms are very sensitive to *hyperparameter* selection and initialization, which require additional evaluations and consideration. Many published results often contain single-initialization results, which may misrepresent the capabilities of an approach and yield poorer performance if deployed. Furthermore, traditional metrics only indicate the performance of an algorithm during its conception phase, close to its training data. I have found that agents tend to over fit, picking the same assets regardless of market variations, which reflect favorably in traditional metrics. However, when the market evolves, the static behavior degrades. Evaluating the robustness of algorithms and their capacity to adapt to uncertainty and out-of-distribution data gives greater insight into their training quality and generalization capabilities.

So, what is the best way to compare an RL model against a benchmark? Generally there are **classic and machine learning benchmarks.**

Classic benchmarks, as mentioned before, are base on the Markovitz's intuition. Based on the investment's goal there are many optimizations. The most common in the literature are the maximization of the Sharpe ratio or the minimization of risk with a return's target. Other less common ways, but still interesting, are the maximization of the maximum diversification portfolio[3], the maximum decorrelation portfolio and the risk parity portfolio (Maillard, Roncalli, and Teıletche 2010; Roncalli and Weisang 2016). Models that follow classical theory are excellent for comparing one's reinforcement learning model, but misleading benchmarks are also used. For example, a portfolio that is often implemented as a benchmark is the equally weighted portfolio in which each asset has the same weight over time. This approach does not follow any kind of optimisation or even diversification logic, thus enhancing great performance of the model created.

Reinforcement learning models are widely computed. The main state-of-art strategies are listed in the following table.

The most important RL algorithm used as a benchmark is the EIIE model, computed by Jiang (Jiang, 2017). This model is highly significant due to its innovative approach to dealing with the complexities of financial markets, particularly in asset allocation. This model is called the Ensemble of Identical Independent Evaluators (EIIE). EIIE employs a neural network architecture designed to assess the potential of different assets independently but in parallel, which then influence the decision on portfolio weights dynamically. This approach allows the model to evaluate multiple assets simultaneously and update portfolio allocations based on real-time data. The model operates without a prior financial model, relying instead on the data-driven capabilities of machine learning to optimize asset allocation. It integrates various techniques, including Portfolio-Vector Memory and Online Stochastic Batch Learning. Since its excellent performances is used in many papers analyzed.

Finally, the most *"fair"* approach is to compare the own RL model with others, using same input data, same training, same goal in the reward function. Better approach would be to align investor's preferences to evaluate the model; for example if the goal of an investment is to beat the market, the

[3]the concept of diversification is simply the ratio of the weighted average of volatilities divided by the portfolio volatility, but it can be developed

Agent action	**Brief description**
Anticor	Portfolio construction based on stock correlations and anti-correlations in consecutive windows.
BAH	Buy and hold the asset selected, retaining the investment ignoring short-term ups and downs in market price.
BCRP	Redistribute the investment wealth each trading day based on hindsight.
BNN	A nearest neighbor-based strategy exploited by histograms from the nonparametric statistics method.
CRP	On a daily basis, maintain the same wealth distribution among a particular group of assets.
CWMR	Create a Gaussian distribution to represent the portfolio vector, and then update it in accordance with the mean reversion principle.
DDPG	Baseline DRL strategy for constructing portfolios.
ONS	Track the best CRP to date and adopt a L2-norm regularization to limit the variability of the portfolio.
PAMR	Adopt the mean reversion model of financial time series based on online passive aggressive learning.
RMR	Construct portfolios based on the median reversion property of financial time series using a robust L1-estimator and passive aggressive online learning.

Table 1: State-of-art algorithms

benchmark will be target market. Anyway, the best benchmark doesn't exist, but there are benchmarks more appropriate than others, based on the investment's objective.

5 Real market environment

5.1 Market's frictions

All of the authors consider two main assumptions:

- **Market Liquidity**: It is assumed that it is always possible to buy or sell an asset at any time. This assumption relies on the market being sufficiently liquid, meaning there are always enough buyers and sellers to complete transactions without significant delays.
- **Market Impact**: The second assumption is that the transactions do not affect the values of the financial instruments involved. This is based on the idea that the size of the transactions made by an individual or entity is small relative to the total market liquidity. Therefore, these trades do not significantly move market prices.

However in real-world scenarios, both assumptions can sometimes be violated, particularly in markets that are illiquid or during periods of high volatility. It could be interesting to remove these assumptions or one of them, in order to create an environment that is as real as possible for the agent. Only one paper addressed the market impact, using the Bertsimas-Lo model (Bertsimas and Lo, 1998). It is a quantitative framework that describes how large trades can impact market prices. It distinguishes between *temporary* and *permanent* impacts. Temporary impact reflects the immediate effect of a trade on asset prices. According to the model, when a large volume of an asset is bought, it temporarily drives the price up; conversely, selling a large volume drives the price down. This price change is temporary and typically reverses once the trade is completed. The temporary impact is modeled as a function of the rate of trading (i.e., how quickly the assets are being bought or sold). Permanent impact models the longer-lasting effect of trades on asset prices. For instance, a significant buy order might lead to a higher price level even after the trade has been completed, as the market adjusts its valuation of the asset. This type of impact does not reverse and becomes a new baseline for future price movements (Lu, 2023).

An easier way to address market impact might be the implementation of a VWAP (Volume Weighted Average Price) strategy, widely used in trading. This strategy allows the agent to execute trades based on the average price weighted by trading volume over a specific time frame. By executing trades at the volume-weighted average price over a given time period, investors can reduce their impact on the market and minimize price slippage. For example, if an investor wants to buy $10,000$ shares of a stock, they could execute the trade over a period of one hour at the VWAP price (Zarattini, 2023).

A usual real-market feature taken into account is **transaction costs**. Over 60% of the papers analyzed take transaction costs into account in the model. Assuming a reference market transaction cost, this component can be approximated by multiplying the commission rate by the amount of the order. More complex model provide a different commission rate for both buying and selling a stock; others create complex model to predict the transaction costs in a more precise way. Usually the way such market friction is included is in the reward function, with a minus sign. Doing this, the agent has not only maximizing returns as goal, but also minimizing transaction costs. An interesting approach is given in the ESG environment (Garrido et al., 2023). The authors computed two reward functions, corresponding at two scenarios. If the ESG weighted value of the actual portfolio is higher than the ESG target value (it could be decided by the investors or portfolio managers), the agent will receive a grant as a prize. If, however, the peer portfolio strategy performs worse in ESG terms, the agent will pay additional taxes. It is an innovative way to include investors' preferences that care more about the environment. The *grant-taxes framework* could represent the future regulation framework, where are incentivized more ethic investments trough public policies.

Finally in portfolio management there might be other constraints, such as **cardinality** and/or **bounding**. The cardinality constraint restricts the number of assets to be included in the portfolio to exactly K. This constraint is particularly useful in managing the size and complexity of a portfolio, ensuring diversification without overcomplication. It ensures that exactly K assets are chosen, no more, no less. While, the bounding constraint, *bounds* the investment in any asset chosen to be included in the portfolio within a specific range defined by lower and upper limits (ϵ_i and δ_i respectively). If an asset a_i is selected ($s_i = 1$), then its corresponding weight w_i in the portfolio must lie between ϵ_i (minimum buy-in threshold) and δ_i (maximum buy-in threshold), where $0 \leq \epsilon_i \leq \delta_i \leq 1$. It controls

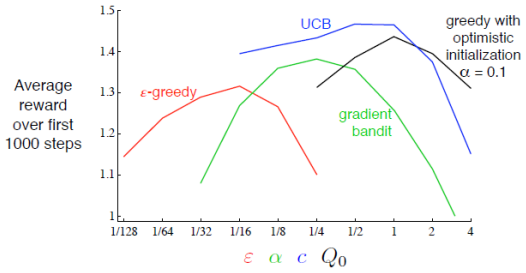

Figure 5: Parameter study, Sutton and Barto, 2018

the investment exposure to individual assets by ensuring that if an asset is chosen, a minimum amount ϵ_i must be invested in it, but no more than δ_i, thus avoiding over-exposure. It is also useful by providing flexibility in how much to invest in each asset while maintaining precise control over the portfolio's composition (Cui, 2022). It could be aligned to investors' preferences, setting the weight for the sector preferred (e.g. $w = 50\%$ in tech companies).

5.2 Sensitive analysis and stress testing

Sensitive analysis is a key to show the performance of the RL model. A classical sensitive analysis is tuning the *hyperparameters* to see how the agent performs with different objectives. Hyperparameter optimization is a critical component of reinforcement learning, significantly influencing the training process and model performance. Key hyperparameters like the learning rate, which controls the magnitude of weight updates, require fine-tuning because overly high rates cause divergence while low rates slow convergence. Batch size, determining the number of samples processed per update, must strike a balance between accurate gradient estimates and memory efficiency. The discount factor (gamma) governs the weighting of future rewards, with high values prioritizing long-term gains and lower values focusing on immediate returns. Replay buffer size controls the diversity of experiences used in training, where larger buffers prevent overfitting to recent trends but may incorporate less relevant older experiences. The exploration-exploitation trade-off, crucial for balancing between discovering new actions and leveraging known profitable ones, often uses epsilon-greedy strategies with carefully tuned decay rates to shift exploration over time. Regarding to the trade-off between exploration and exploitation is interesting to see how changes the model's performance, under different hyperparameters. Figure 5 shows this measure for the various bandit algorithms, each as a function of its own parameter shown on a single scale on the x-axis. This kind of graph is called a parameter study. magnitude. Overall, on this problem, Upper Confidence Bound seems to perform best.[4]

Optimizing these parameters involves methods like grid search, which exhaustively evaluates predefined sets but is computationally intensive, and random search, which samples the hyperparameter space more efficiently. Bayesian optimization refines the search space using probabilistic models, progressively improving predictions but requiring significant computation. Hyperband cleverly combines random sampling with early stopping to rapidly prune underperforming configurations, while genetic algorithms simulate evolution to evolve optimal hyperparameters but risk local optima.

Stress testing and scenario analysis are critical tools in reinforcement learning (RL) applied to dynamic portfolio management, ensuring strategies can withstand adverse market conditions and navigate varied economic environments. Stress testing simulates extreme scenarios like the GFC in 2008,the COVID-19 or Russia and Ukraine conflict, to identify vulnerabilities in the RL strategy. Historical scenarios replay past crises, while hypothetical scenarios create artificial but plausible market conditions, such as liquidity shocks or rapid currency depreciation. Sensitivity analysis applies shocks to key market factors like correlations or volatility to test how changes impact the portfolio's performance and determine whether the RL model can rebalance allocations effectively. Liquidity management scenarios evaluate the effects of increased transaction costs or market illiquidity on portfolio returns.

Scenario analysis, on the other hand, explores a broader range of potential future market outcomes. Market regimes are simulated to understand how an RL strategy adapts to bull, bear, or sideways

[4]see Sutton and Barto 2018 for further details

markets. Macro-economic indicators like inflation rates, GDP growth, or geopolitical tensions are incorporated to examine adaptability to broader economic shifts. Probabilistic models such as Monte Carlo simulations generate a spectrum of market trajectories, providing insights into potential return distributions. Scenario analysis ensures that RL strategies can optimize asset allocation and risk management in the face of uncertain market environments. Together, these approaches rigorously assess how adaptable and resilient the RL model is, ultimately increasing the robustness and reliability of RL-based portfolio management.

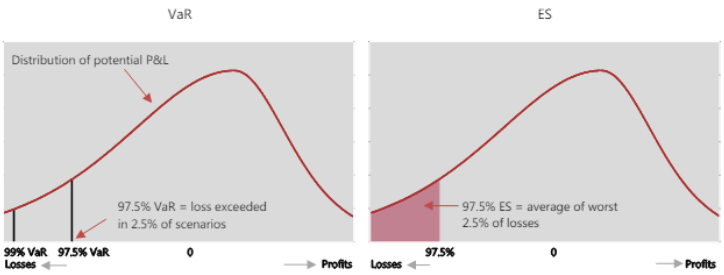

Figure 6: Bank for International Settlement

6 Conclusions

In conclusion, the exploration of reinforcement learning (RL) within the domain of dynamic portfolio management reveals significant potential for revolutionizing investment strategies. As the financial landscape continually evolves, the adaptability and predictive power of RL offer a promising avenue for optimizing portfolios in real-time, responding adeptly to market changes. However, the application of these sophisticated models also necessitates a rigorous evaluation of their performance, ethical considerations, and regulatory compliance. As this technology progresses, collaborative efforts between academia, industry, and regulatory bodies will be crucial in harnessing the benefits of RL while ensuring its transparency, fairness, and alignment with investor needs and market stability. This synthesis of innovation and regulation will ultimately dictate the successful integration of RL into broader financial practices.

Finally I want to conclude with some considerations on model's construction.

1. It is not usual to add the option to invest in data defensive stocks. When there is turmoil of financial markets, causing an increase in uncertainty and volatility, investing in safer assets (*flight to quality)*. That is why, combining risky and risk free assets, can be a pivotal strategy. In this scenario will be interesting to create a model with two agents, each one with a different scope. The first agent is responsible to analyze market trends, in order to understand what is the phase of the financial cycle. The second one, after have received signal from the first agent, will invest in the market, deciding the efficient allocation in the portfolio between different type of assets' risk.

2. VaR is widely used to address market risk in portfolio management, especially in RL algorithms. The VaR is a quantile function of the Profit and Loss distribution, so given a specific confidence level (e.g. 99%), it doesn't capture the tail risks with probability 1%. It might be more correct to use the Expected Shortfall (or CVaR) to manage market risk. It is the average of losses *beyond* the VaR and it is a coherent risk measure. Given the same confidence level, Expected Shortfall is greater or equal than the VaR, so it is a more cautious market risk measure. Note that a VaR with a confidence level at 99% is roughly equal to the ES at 97.5% (see Figure 6). Furthermore, most authors take into account only market risk, when it might be more appropriate including in the model other type of risks: credit risk, liquidity risk, climate risk...

3. By integrating investor's preferences such as risk appetite, investment horizon (long-term vs. short-term), desired level of diversification, and specific financial objectives like inflation protection or retirement planning into the reward function, an RL model can be tailored to adapt its trading behavior accordingly. For instance, younger investors might prefer a reward structure that emphasizes capital growth through riskier investments due to a longer time horizon and higher risk tolerance. In contrast, older investors nearing retirement might prioritize stability and income, focusing the RL model on less volatile assets or those that offer regular dividends. Additionally, incorporating factors like the amount of capital invested can help the model adjust its risk exposure based on the investor's financial capacity, ensuring that the portfolio's risk level is always aligned with the investor's overall financial situation and life stage. This personalized approach not only makes the model more practical for individual use but also enhances its ability to meet diverse investor needs in a dynamic market environment.

4. Since the clear trend to train RL models in the U.S. environment, testing other financial markets can offer intriguing possibilities due to unique market dynamics and characteristics. Markets like Italy's feature different levels of liquidity, volatility, and regulatory environments compared to the U.S. market. These distinctions could challenge existing RL models and potentially uncover unique insights into asset behavior under different economic conditions. For instance, the Italian market is typically less liquid and smaller in scale, which may affect the model's ability to learn from and react to price movements effectively. Additionally, Italy's distinct economic cycles, influenced by regional political and economic events, provide a diverse testing ground to enhance the robustness and adaptability of RL algorithms. Exploring these differences can lead to more generalized and resilient trading strategies that are not solely tailored to the characteristics of large, highly liquid markets like the U.S., thus broadening the applicability and effectiveness of RL in global financial contexts.

5. Including beta can be an interesting approach to address the risk and volatility of individual stocks relative to market movements. This is beneficial for constructing a balanced portfolio that aims to optimize the risk-return profile. Stocks with different beta values respond differently to market changes, and understanding this behavior helps in making informed decisions about buying or selling stocks based on expected market volatility. Developing an intuition for including beta in portfolio management involves understanding the fundamental principle that beta measures a stock's sensitivity to market movements. By using beta, investors can predict how changes in the market will affect their investments and manage potential risks more effectively. For instance, in volatile markets, investors might prefer low-beta stocks as they are likely to experience smaller fluctuations compared to the overall market. Conversely, in a bullish market, high-beta stocks might be preferred as they offer the potential for higher returns, albeit at a higher risk. This strategic use of beta helps in achieving a more tailored investment approach that aligns with the investor's risk tolerance and market outlook.

References

1. Markowitz, Harry. "Portfolio Selection." The Journal of Finance, 1952.
2. Sharpe, William F. "The Sharpe Ratio." The Journal of Portfolio Management, 1994
3. WaveCorr: Deep Reinforcement Learning with Permutation Invariant Policy Networks for Portfolio Management." Proceedings of the International Conference on Learning Representations (ICLR), 2022.
4. Huang, Bo; Yu, Yu. "On Optimal Tracking Portfolio in Incomplete Markets: The Classical Control and the Reinforcement Learning Approaches." University of Science and Technology of China, The Hong Kong Polytechnic University, 2023.
5. "Portfolio Constructions in Cryptocurrency Market: A CVaR Based Approach." Economic Management, 2023.
6. Velay, Marc; Doan, Bich-Liên; Rimmel, Arpad; Popineau, Fabrice; Daniel, Fabrice. "Benchmarking Robustness of Deep Reinforcement Learning Approaches to Online Portfolio Management." Université Paris-Saclay, CNRS, CentraleSupélec, LUSIS, 2023.
7. Jiang, Zhengyao; Xu, Dixing; Liang, Jinjun. "A Deep Reinforcement Learning Framework for Financial Portfolio Management." 2017.
8. Ngo, Nguyen; Nguyen, Nguyen. "Does Reinforcement Learning Outperform Deep Learning and Traditional Portfolio Optimization Models in Frontier and Developed Financial Markets?" Proceedings of the RIBF, 2023.
9. Wang, Huang; Tu, Zhang; Xu, Xu. "DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding." Proceedings of the AAAI Conference on Artificial Intelligence, 2021.
10. Zhao, Ma; Li, Zhang. "Asset Correlation Based Deep Reinforcement Learning for the Portfolio Selection." Expert Systems With Applications, 2023.
11. Sun, Wei; Yang, Yang. "GraphSAGE with Deep Reinforcement Learning for Financial Portfolio Optimization." Expert Systems With Applications, 2024.
12. Qiu, Liu; Lee. "The Design and Implementation of a Deep Reinforcement Learning and Quantum Finance Theory Inspired Portfolio Investment Management System." Expert Systems With Applications, 2024.
13. Dong, Zheng; "Soft Imitation Reinforcement Learning with Value Decomposition for Portfolio Management." Applied Soft Computing, 2024.
14. Chen, Chen; Chen, Huang. "Knowledge Distillation for Portfolio Management Using Multi-Agent Reinforcement Learning." Advanced Engineering Informatics, 2023.
15. Soleymani, Paquet. "Financial Portfolio Optimization with Online Deep Reinforcement Learning and Restricted Stacked Autoencoder." Expert Systems With Applications, 2020.
16. Jin, El-Saawy. "Portfolio Management using Reinforcement Learning." Stanford University.
17. Van Staden, Forsyth; Li, Li. "A Parsimonious Neural Network Approach to Solve Portfolio Optimization Problems." National Australia Bank, University of Waterloo, 2023.
18. Zhang, "Relation-aware Transformer." [No further details provided, 2020 presumed from the filename].
19. Cui, Du; Yang, Ding. "Multi-period Portfolio Optimization Using a Deep Reinforcement Learning Hyper-heuristic Approach." Technological Forecasting and Social Change, 2024.
20. Fu, Krishnamurthy. "A Deep Neural Network Algorithm for Linear-Quadratic." ArXiv, 2023.
21. Garrido-Merchan, Mora-Figueroa; Cruz-Guzman, Coronado. "Deep Reinforcement Learning for ESG Financial Portfolio Management." Universidad Pontificia Comillas, 2023.