COMILLAS
UNIVERSIDAD PONTIFICIA

Facultad de Ciencias Económicas y Empresariales

ICADE

MULTI-FIDELITY BAYESIAN OPTIMIZATION FOR DEEP REINFORCEMENT LEARNING FOR PORTFOLIO MANAGEMENT

Autor: Beatriz Díaz Nameth

Director: Eduardo César Garrido Merchán

MADRID | Abril 2024

ABSTRACT

The financial landscape is constantly evolving, presenting challenges to traditional methodologies that struggle to adapt to the dynamic nature of markets. Markowitz's Modern Portfolio Theory (MPT) assumes stability in market conditions, overlooking the inherent volatility and rapid changes that characterize real-world financial environments. This inconsistency highlights the need for innovative approaches capable of addressing the non-stationary nature of markets. Deep Reinforcement Learning (DRL) emerges as a promising solution, leveraging its adaptability to changing conditions and its ability to learn from historical data and real-time feedback. By leveraging the power of DRL, this thesis aims to revolutionize portfolio management, offering a more dynamic and responsive framework to navigate today's complex financial landscapes.

This thesis explores the intersection of Multi-Fidelity Bayesian Optimization (MFBO) and Deep Reinforcement Learning (DRL) to improve portfolio management strategies. The research explores how MFBO can optimize hyperparameters within DRL algorithms, particularly focusing on their adaptability to non-stationary market conditions. By challenging the assumptions of traditional portfolio optimization models like Modern Portfolio Theory (MPT), this study aims to utilize the dynamic nature of DRL to overcome the limitations of static approaches. Through experimentation and comparative analysis with established methods such as Bayesian Optimization and Random Search, the study not only seeks to validate the efficacy of MFBO-DRL techniques but also aims to uncover new insights into portfolio optimization.

By combining theoretical principles with empirical evidence, this research contributes to the wider understanding of financial decision-making, offering potential paths for the progress of portfolio management practices.

Key words: Deep Reinforcement Learning, Multi-fidelity Bayesian Optimization, Portfolio Optimization, Modern Portfolio Theory, Bayesian Optimization, Random Search and Hyperparameters.

RESUMEN EJECUTIVO

El mundo financiero está en constante evolución, lo que presenta desafíos para las metodologías tradicionales que luchan por adaptarse a la naturaleza dinámica de los mercados. La Teoría Moderna de Carteras de Markowitz (MPT, por sus siglas en inglés) asume la estabilidad en las condiciones del mercado, no teniendo en cuenta la volatilidad inherente que caracteriza a los entornos financieros. Esta inconsistencia destaca la necesidad de enfoques innovadores capaces de abordar la naturaleza no estacionaria de los mercados. El Deep Reinforcement Learning (DRL) surge como una solución, aprovechando su capacidad de adaptación a condiciones cambiantes y su habilidad para aprender de datos históricos y retroalimentación en tiempo real.

Este TFG profundiza en la intersección de la Optimización Bayesiana de Multi-Fidelidad (MFBO) y el DRL para mejorar las estrategias de gestión de carteras. La investigación explora cómo MFBO puede optimizar hiperparámetros dentro de los algoritmos de DRL, centrándose particularmente en su capacidad de adaptación a condiciones de mercado no estacionarias. Este estudio tiene como objetivo aprovechar la dinámica del DRL para superar las limitaciones de los enfoques estáticos al desafiar los supuestos de los modelos tradicionales de optimización de carteras como la MPT. A través de experimentos y análisis comparativos con métodos como la Optimización Bayesiana y la Búsqueda Aleatoria, el estudio no solo busca validar la eficacia de las técnicas MFBO-DRL, sino que también tiene como objetivo descubrir nuevos conocimientos sobre la optimización de carteras.

Al combinar fundamentos teóricos con evidencia empírica, esta investigación contribuye a la comprensión general de la toma de decisiones financieras, ofreciendo posibles vías para el progreso de las prácticas de gestión de carteras.

Palabras clave: Deep Reinforcement Learning, Optimización Bayesiana de Multi-Fidelidad, Optimización de carteras, Teoría Moderna de Carteras, Optimización Bayesiana, búsqueda aleatoria e hiperparámetros.

INDEX

FIGURE INDEX

EQUATIONS INDEX

1. INTRODUCTION

The financial industry is continuously challenged by the ever-changing landscape of markets, characterized by volatility, emerging trends, and unpredictable shifts in sentiment. Traditional portfolio optimization approaches, particularly Markowitz's Modern Portfolio Theory (MPT) which consists of constructing diversified portfolio in order to optimize the returns of risk averse investors, often struggle to navigate these non-stationary market conditions effectively (Durall, 2022). However, some authors such as Eric Benhamou, David Saltiel, Sandrine Ungari, and Abhishek Mukhopadhyay (2020) have shown the growing potential of Deep Reinforcement Learning (DRL) in finance, offering insights into the future of portfolio management. However, the combination of DRL and multi-fidelity optimization techniques remains a less-explored territory.

The financial sector faces difficulties as market conditions keep changing, and traditional methods struggle to adjust to these changes. The adaptability and learning capability of DRL models directly address the limitations of MPT, which assumes static market conditions. The inherent volatility and constantly changing nature of markets weaken the effectiveness of these traditional methodologies. The challenge lies in finding approaches to deal with the rapid fluctuations and non-stationary nature of financial markets (Benhamou et al., 2020).

This thesis aims to critically assess and address the limitations of conventional portfolio optimization methods by evaluating the applicability and efficiency of utilizing Deep Reinforcement Learning (DRL) in portfolio optimization (Benhamou, 2023). By leveraging DRL's adaptability to non-stationary market conditions, this thesis aims to introduce innovative solutions that exploit DRL's ability to learn from historical data and real-time feedback, opening doors to a more robust and dynamic approach for portfolio management.

The research will focus on investigating how DRL, known for its adaptability and learning capabilities, can be used in optimizing portfolios. It will focus on how DRL can adapt to evolving market trends using historical data and real-time feedback. Additionally, the study will explore the integration of multi-fidelity optimization techniques to fine-tune DRL parameters effectively for portfolio management.

Multi-fidelity optimization is an approach that balances computational cost and optimization accuracy, and it is introduced to fine-tune the parameters of DRL models (Li & Li, 2024). This exploration aims to improve the existing knowledge and offer a promising path for innovative portfolio management practices. It involves using varying levels of accuracy to efficiently search for optimal hyperparameters, balancing computational costs and optimization accuracy, ensuring adaptability to dynamic financial markets needs and boosting both the efficiency and quality of portfolio optimization.

This document is structured as follows: we commence with an introduction which provides an overview of the research objectives and context. Following that, the second section explores the current state of the art in multi-fidelity Bayesian Optimization and Deep Reinforcement Learning (DRL) for portfolio management. The third section explains the scope of this thesis, presenting the specific areas and challenges addressed. Following this, the methodology used to achieve the objectives set is exposed. Subsequently, the theoretical framework is elaborated, providing an understanding of the methodologies and models behind the research. The fifth section covers the application and empirical analysis of multi-fidelity Bayesian Optimization and DRL in portfolio management. The sixth section presents the empirical results obtained from the analysis carried out and, finally, the document concludes with a summary of key findings, implications, and suggestions for future work. The bibliography section offers references and supporting materials for a better understanding of the research.

2. STATE OF ART

In today's financial landscape, multi-factor models, such as Fama-French Three-Factor Model, have emerged as a promising path in delivering superior returns in the stock market (Chen, 2024). However, as research on these models advances, their limitations have arisen. For instance, Hou et al. (2020) discovered that a significant number of identified anomalies failed to meet strict statistical thresholds, questioning the validity of the established factors. Additionally, concerns arise regarding the statistical foundations of these models, such as the misapplication of p-values in significance testing and limitations in traditional econometric analysis (Huang et al., 2020).

To address these limitations, a data-driven approach offers an alternative to traditional econometric methods. While the traditional approach involves predetermining a linear relationship between asset returns and factors, the data-driven concept seeks to derive rules directly from data. Mullainathan and Spiess (2017) highlighted the breakthrough achieved by shifting the focus from deducing rules to letting the data reveal optimal rules.

The concept of data-driven strategies dates back to the 1990s, when academic interest arose in artificial neural networks. However, early neural network technologies faced challenges dealing with gradient-related issues, resulting in a decade of slowness in AI research. The application of AI in finance primarily relies on machine learning, categorized into supervised, unsupervised, and reinforcement learning (Bengio et al. 1994).

There has been a significant transformation in the world of financial decision-making driven by the huge amount of data and the later evolution in data processing and analysis techniques. Traditional methodologies, which rely on model assumptions from stochastic control theory and other analytical approaches, often fall short in capturing the complexities of financial environments. These limitations have driven the emergence of Reinforcement Learning (RL) approaches in finance, utilizing extensive financial data with fewer model assumptions to improve decision-making in complex financial scenarios (Hambly, Xu & Yang, 2023).

Reinforcement Learning stands out for its interaction-based data generation and learning of optimal strategies by agents from this data. While early applications of RL primarily used

Q-learning and Policy Gradient algorithms for asset management, these models faced limitations with increased complexity and local optimality issues (Bengio et al. 1994). The breakthrough came in 2015 when Mnih et al. successfully applied deep reinforcement learning (DRL) in playing computer games, displaying its potential in AI.

Traditionally, financial decision-making problems have been approached through stochastic processes and techniques from stochastic control (Singh et al., 2022). However, these models often face a trade-off between viability and realism. On the one hand, simpler models offer straightforward strategies but oversimplify market behavior, potentially leading to less optimal strategies and financial losses. On the other hand, models trying to replicate real market behavior are exceptionally complex and computationally challenging (Huang et al., 2020).

The finance sector's data explosion has revolutionized data processing and statistical modeling. RL strategies enable agents to learn optimal decision-making through interactions with a system, finding successful applications in order execution, market making, and portfolio optimization, especially in scenarios with limited market information.

In the rapidly changing environment of portfolio management, the advancements in machine learning, particularly Deep Reinforcement Learning (DRL), have demonstrated significant potential in optimizing investment strategies. The integration of DRL into portfolio management aligns with the fundamental objective of maximizing cumulative rewards, specifically returns, while interacting with the dynamic financial market environment (Wang et al., 2021). The adaptability and sequential learning capabilities of DRL make it an ideal candidate for addressing the complexities of financial markets, where it can autonomously learn from past experiences and adapt to new market conditions (Bartram et al., 2021).

Various studies have investigated the application of DRL in portfolio management. Jiang, Xu, and Liang (2016) demonstrated the use of DRL in forming cryptocurrency portfolios, achieving significant returns over short periods, despite the market's inherent volatility (Wu et al., 2021). This highlights DRL's effectiveness in navigating complex and dynamic markets.

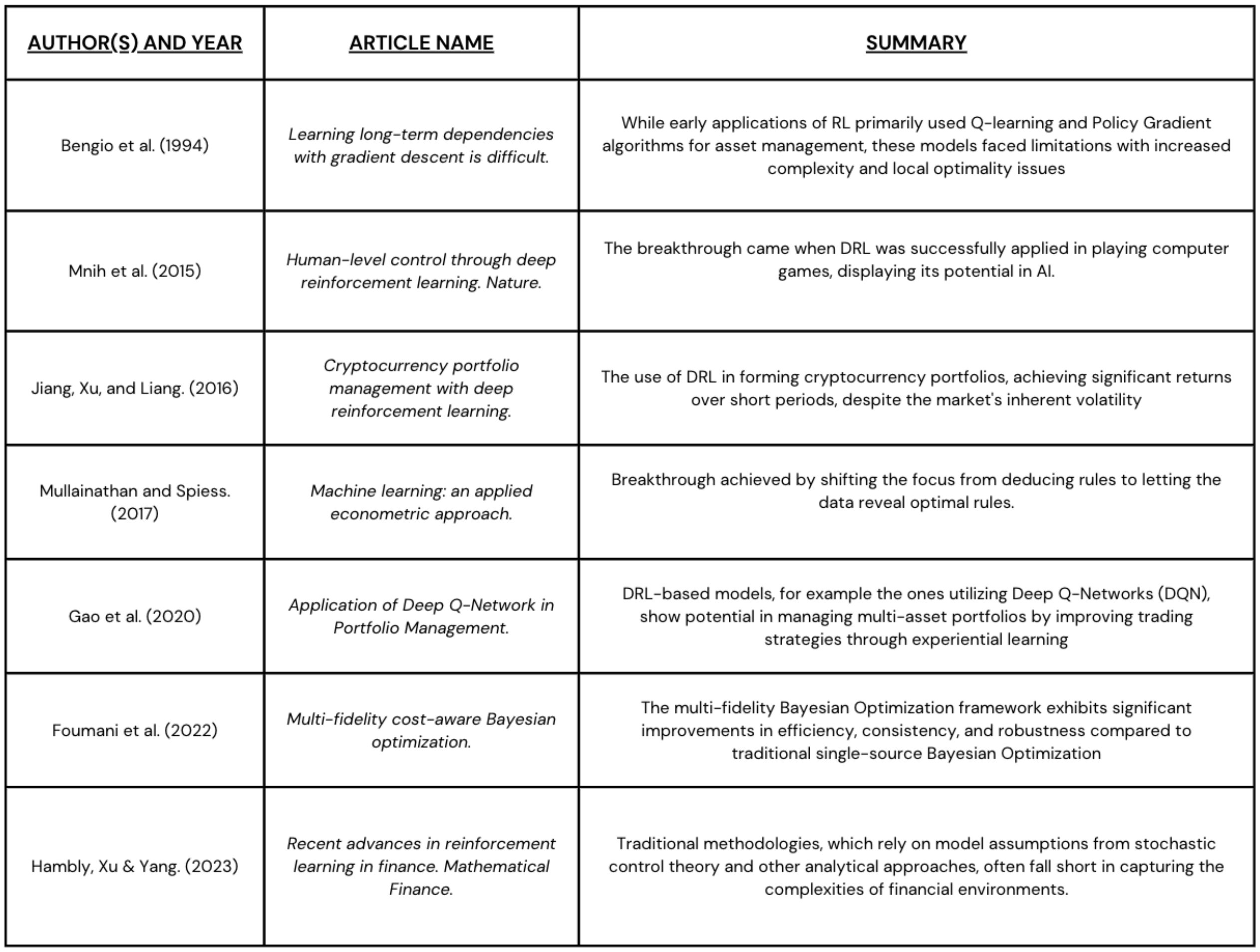
Integrating DRL into portfolio management introduces a set of challenges inherent to the complex nature of financial markets. These markets demand sophisticated models capable of effectively understanding and leveraging market dynamics. Traditional methodologies, although effective in specific scenarios, often fail to adapt to the changing market landscape (Wang et al., 2021). In contrast, DRL-based models, for example the ones utilizing Deep Q-Networks (DQN), show potential in managing multi-asset portfolios by improving trading strategies through experiential learning (Gao et al., 2020). However, these methods face certain limitations, including being susceptible to overfitting and the complexity of managing hyperparameters.

Recent efforts to optimize DRL models in portfolio management focus on leveraging multi-fidelity Bayesian Optimization. This methodology involves utilizing diverse data sources with varying levels of accuracy and costs. By doing so, it minimizes the overall optimization cost, moving beyond an exclusive reliance on costly high-fidelity data (Foumani et al., 2022). The multi-fidelity Bayesian Optimization framework presented by Foumani et al. exhibits significant improvements in efficiency, consistency, and robustness compared to traditional single-source Bayesian Optimization. This shows the promising potential of multi-fidelity Bayesian Optimization in optimizing expensive black-box function within portfolio management contexts.

Despite the advancements achieved, the application of DRL in portfolio management remains an active area of investigation, presenting numerous opportunities for future exploration. Conducting studies on volatility and experimenting across diverse markets could refine DRL frameworks, enhancing their adaptability in various financial landscapes (Lucarelli et al., 2020).

In summary, DRL and multi-fidelity Bayesian Optimization represent revolutionary methodologies within portfolio management, offering advanced tools for navigating uncertain markets. The continuous improvements in these approaches, along with the integration of multi-fidelity data, suggest the potential for creating more robust and effective strategies for optimizing portfolios.

Figure 1. Summary of State of Art.



AUTHOR(S) AND YEAR	**ARTICLE NAME**	**SUMMARY**
Bengio et al. (1994)	*Learning long-term dependencies with gradient descent is difficult.*	While early applications of RL primarily used Q-learning and Policy Gradient algorithms for asset management, these models faced limitations with increased complexity and local optimality issues
Mnih et al. (2015)	*Human-level control through deep reinforcement learning. Nature.*	The breakthrough came when DRL was successfully applied in playing computer games, displaying its potential in AI.
Jiang, Xu, and Liang. (2016)	*Cryptocurrency portfolio management with deep reinforcement learning.*	The use of DRL in forming cryptocurrency portfolios, achieving significant returns over short periods, despite the market's inherent volatility
Mullainathan and Spiess. (2017)	*Machine learning: an applied econometric approach.*	Breakthrough achieved by shifting the focus from deducing rules to letting the data reveal optimal rules.
Gao et al. (2020)	*Application of Deep Q-Network in Portfolio Management.*	DRL-based models, for example the ones utilizing Deep Q-Networks (DQN), show potential in managing multi-asset portfolios by improving trading strategies through experiential learning
Foumani et al. (2022)	*Multi-fidelity cost-aware Bayesian optimization.*	The multi-fidelity Bayesian Optimization framework exhibits significant improvements in efficiency, consistency, and robustness compared to traditional single-source Bayesian Optimization
Hambly, Xu & Yang. (2023)	*Recent advances in reinforcement learning in finance. Mathematical Finance.*	Traditional methodologies, which rely on model assumptions from stochastic control theory and other analytical approaches, often fall short in capturing the complexities of financial environments.

Source: Own elaboration

2.1. Deep Reinforcement Learning in finance

Deep reinforcement learning (DRL) combines reinforcement learning with deep learning's power (Benhamou, 2023). DRL is for great use when dealing with complex tasks in dynamic environments, like financial markets. It works by learning from interactions with the environment, using rewards and sanctions to adjust its strategies. DRL has a wide range of applications in finance, like managing portfolios, handling risks, and trading using algorithms (Hambly, Xu & Yang, 2023). Furthermore, it can optimize various goals, including returns, diversification, and transaction costs and can adapt to different scenarios by processing lots of data, even when it's noisy, like market prices and news. Finally, it can also learn efficiently from limited data using techniques like experience replay and data augmentation (Osterrieder, J., 2023).

There are several components that come into play in the world of Reinforcement Learning (RL). There is the state space, which includes factors like stock market conditions and asset portfolio ratios. In finance, it's tough to describe the entire state space due to its complexity, so we often use simplified versions. The action space involves trading decisions, which can be as simple as buying, selling, or holding assets (Hu & Lin, 2019).

RL also considers transition probabilities, which are the chances of moving from one state to another based on an action (Benhamou et al., 2023). The reward function is essential because it provides a measure of how well things are going. To make RL work, agents learn to choose actions that maximize expected rewards based on feedback from the environment. It involves determining which actions are effective and which are not. This feedback assists in refining their strategies and improving their performance in their tasks (Osterrieder, J., 2023).

2.1.1. Applying DRL to financial problems

DRL has found application in diverse areas of finance, which has enabled the development of intelligent systems to tackle portfolio optimization, risk management, and algorithmic trading challenges. This technology allows these systems to make data-driven decisions, adapt to changing market conditions, and enhance their performance (Osterrieder, J., 2023).

A. <u>Portfolio optimization</u>

In the context of portfolio optimization problems, DRL algorithms have been applied in order to maximize the risk-adjusted return of a portfolio including various assets. This optimization process takes into account budget, risk, and liquidity restrictions. A DRL algorithm can be trained to efficiently select and rebalance a portfolio of stocks by using a neural network to represent the Q-value function. This function quantifies the expected discounted sum of future rewards based on the state and action taken. These algorithms store transitions in an experience replay buffer and learn from them through mini-batch sampling, driven by the loss function and the optimization algorithm (Osterrieder, J., 2023).

In managing portfolios involving multiple assets, it becomes essential to determine how to allocate resources among these assets. An investing strategy plays a crucial role in selecting the most promising assets, and the frequency of portfolio rebalancing stands out as an essential parameter in this process. Portfolio rebalancing involves making decisions about the distribution of resources among the most attractive assets. In the world of trading, especially when employing DRL, a significant focus is placed on optimizing these elements of portfolio management (Khamis & Wang, 2021).

B. Risk management

Risk is an inherent part of financial investments, but using advanced techniques can help minimize it while still yielding profits. Each investor has its risk preferences, with some being cautious and others seeking more risk.

Deep reinforcement learning is used in risk management to minimize losses or portfolio risks while considering objectives like return, liquidity, and diversification. To hedge a portfolio efficiently, the DRL algorithm can take actions to eliminate the risks, like trading derivatives or changing the portfolio's exposure to underlying assets. It gets feedback through rewards and penalties based on how well the hedge performs, allowing the algorithm to adapt its strategies and value functions. This learning process involves saving transitions in a replay buffer and using loss functions and optimization algorithms. Overall, DRL helps manage investment portfolio risks while improving performance and returns (Osterrieder, 2023).

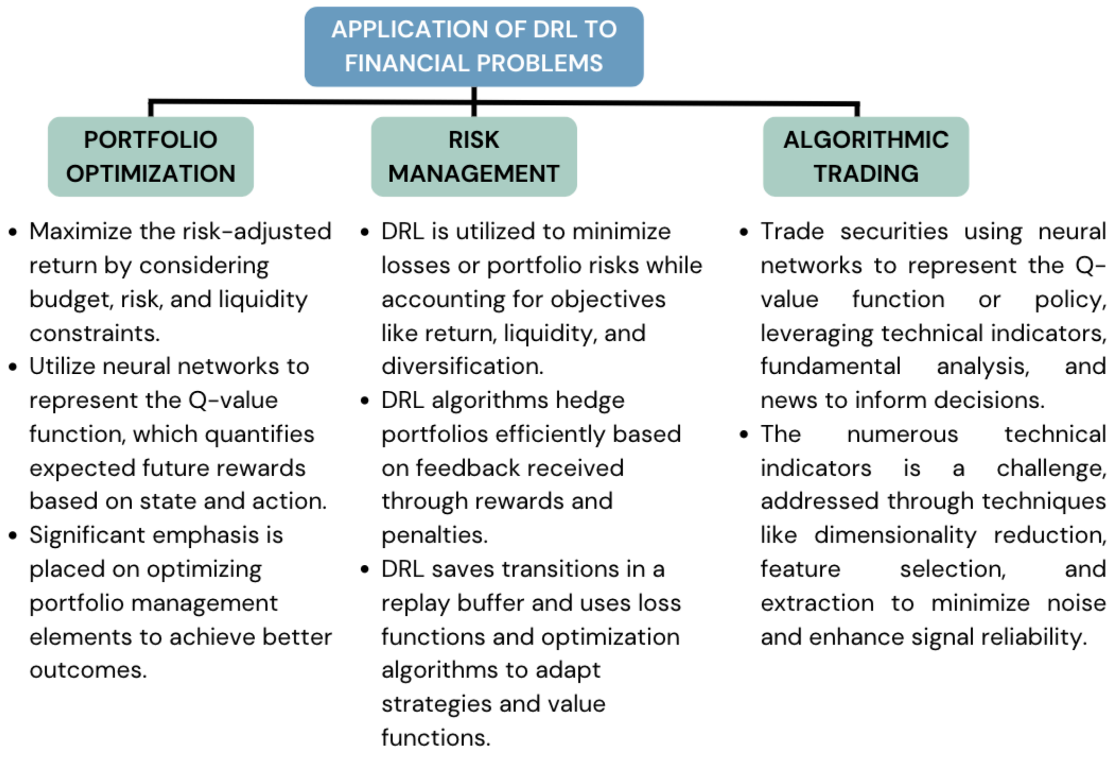
C. Algorithmic trading

Algorithmic trading in stock markets aims to maximize returns by exploiting market volatility through frequent buying and selling of shares. However, this approach faces increasing challenges given the dynamic and complex nature of stock markets. In response, strategies based on DRL have emerged as a possible solution to address these challenges.

Algorithmic trading leverages DRL to automate trade executions based on signals and rules. This involves the use of technical indicators, fundamental analysis, and news to inform trading decisions. DRL algorithms learn to trade securities or baskets of securities using neural networks to represent the Q-value function or policy (Dulac-Arnold et al., 2019). To enhance the learning process, techniques such as experience replay, recurrent neural networks, and proximal policy optimization are utilized to ensure stability and efficiency in decision-making (Pricope, 2021).

One significant challenge in algorithmic trading is the numerous technical indicators, some of which may yield inaccurate signals and show correlations with each other. DRL is used to address this challenge through techniques such as dimensionality reduction, feature selection, and extraction. These methods aim to minimize noise and enhance the reliability of trading signals (Osterrieder, 2023).

Figure 2. Application of DRL in financial problems.



PORTFOLIO OPTIMIZATION

- Maximize the risk-adjusted return by considering budget, risk, and liquidity constraints.
- Utilize neural networks to represent the Q-value function, which quantifies expected future rewards based on state and action.
- Significant emphasis is placed on optimizing portfolio management elements to achieve better outcomes.

RISK MANAGEMENT

- DRL is utilized to minimize losses or portfolio risks while accounting for objectives like return, liquidity, and diversification.
- DRL algorithms hedge portfolios efficiently based on feedback received through rewards and penalties.
- DRL saves transitions in a replay buffer and uses loss functions and optimization algorithms to adapt strategies and value functions.

ALGORITHMIC TRADING

- Trade securities using neural networks to represent the Q-value function or policy, leveraging technical indicators, fundamental analysis, and news to inform decisions.
- The numerous technical indicators is a challenge, addressed through techniques like dimensionality reduction, feature selection, and extraction to minimize noise and enhance signal reliability.

Source: Own elaboration

2.2. Future Opportunities for Deep Reinforcement Learning in Financial Applications

In the changing landscape of financial services and market dynamics, Deep Reinforcement Learning algorithms have made significant advances in automating investment guidance, market making and multi-exchange trading. These advances provide new paths to improve the efficiency and adaptability of financial practices.

A. Robo- Investment guiding

In the world of financial guidance, there is a rising trend regarding automated investment managers, often referred to as robo-advisors. These digital financial advisors have become increasingly popular due to their ability to offer online financial guidance and in management with minimal human intervention. By using the power of mathematical algorithms and incorporating a wide range of data sources, including news, social media insights, sentiment data, and earnings reports, automated investment managers have emerged as a modern alternative to traditional human advisors. This shift was stimulated by the loss of trust in financial services institutions after the 2008 financial crisis (Hambly, Xu & Yang, 2023).

One significant challenge faced by automated investment managers is accurately determining and adapting to clients' risk preferences over time. A client's risk preference can change due to various factors, including market returns and economic conditions. As a result, robo-advisors need to establish a suitable interaction frequency with clients to maintain consistency in risk preference when adjusting portfolio allocations. Additionally, automated investment managers often face the dilemma of balancing client preferences with the pursuit of better investment performance. There is a delicate balance between how often a client is engaged and the accuracy of the information obtained. It is important to consider that collecting real-time data might not always accurately represent the client's true risk aversion due to behavioral biases (Tao et al., 2021).

In this world of robo-advisors, DRL algorithms have shown promise. These algorithms learn from various data sources, including prices, volumes, returns, risk, and sentiment, while optimizing for objectives like return, risk, liquidity, and diversification. They engage with investors to understand their preferences, risk tolerance, and financial goals and suggest portfolios or strategies aligned with their profiles and constraints. DRL algorithms also adapt and update their advice based on investor feedback and behavior, offering a data-driven approach to automated investment management (Osterrieder, 2023).

B. Automated market creation

Market makers act as traders or institutions that provide liquidity by placing buy and sell limit orders for a particular financial instrument. Their primary goal is to profit by earning the bid-ask spread, rather than predicting price movements. They utilize state variables including bid and ask prices, current asset holdings, order-flow imbalance, volatility, and sophisticated market indices. Actions include adjustments in bid/ask prices for posting limit orders and the sizes of limit buy/sell orders. The reward function combines profit maximization, inventory risk minimization, and the improvement of market qualities in a linear combination (Hambly, Xu & Yang, 2023).

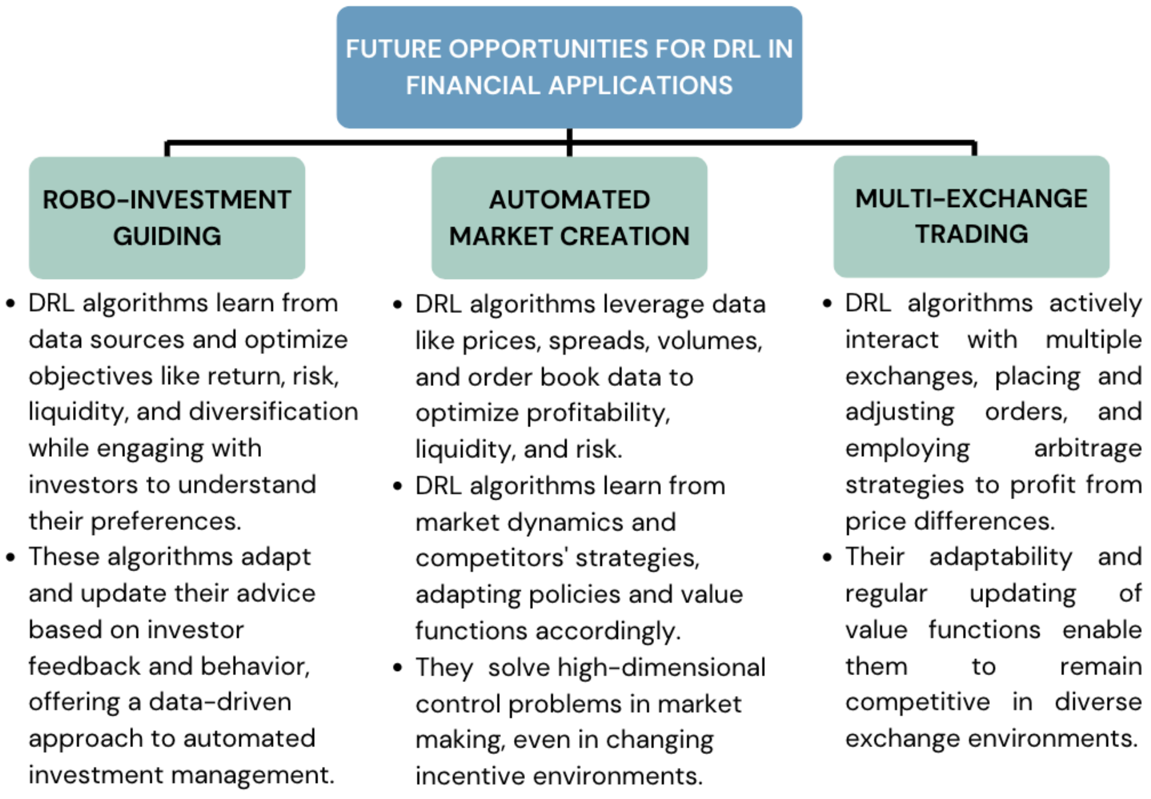
DRL algorithms offer a promising approach for market making activities by leveraging data such as prices, spreads, volumes, and order book data to optimize profitability, liquidity, and risk. These algorithms actively interact with the market, placing and modifying orders, and adapting their inventory and exposure to underlying assets. They also learn from market dynamics and competitors' strategies, allowing them to adapt and update their policies and value functions. Furthermore, DRL algorithms have proven their effectiveness in solving high-dimensional control problems in market making, even in situations with changing incentives in a fully informed environment.

C. Multi-Exchange trading

Deep reinforcement learning algorithms have the potential to revolutionize trading across multiple exchanges. By analyzing a great amount of data, including prices, trading volumes, transaction fees, and order book information, these algorithms can be customized to optimize various objectives. This includes incrementing liquidity, maximizing profitability, and mitigating risk.

These intelligent algorithms actively interact with multiple exchanges, placing and adjusting orders, and even using arbitrage strategies to profit from price differences between exchanges. Their ability to adapt and regularly update their value functions help them remain competitive in diverse exchange environments.

Figure 3. Future Opportunities for DRL in financial applications.


FUTURE OPPORTUNITIES FOR DRL IN FINANCIAL APPLICATIONS
ROBO-INVESTMENT GUIDING
AUTOMATED MARKET CREATION
MULTI-EXCHANGE TRADING

ROBO-INVESTMENT GUIDING

- DRL algorithms learn from data sources and optimize objectives like return, risk, liquidity, and diversification while engaging with investors to understand their preferences.
- These algorithms adapt and update their advice based on investor feedback and behavior, offering a data-driven approach to automated investment management.

AUTOMATED MARKET CREATION

- DRL algorithms leverage data like prices, spreads, volumes, and order book data to optimize profitability, liquidity, and risk.
- DRL algorithms learn from market dynamics and competitors' strategies, adapting policies and value functions accordingly.
- They solve high-dimensional control problems in market making, even in changing incentive environments.

MULTI-EXCHANGE TRADING

- DRL algorithms actively interact with multiple exchanges, placing and adjusting orders, and employing arbitrage strategies to profit from price differences.
- Their adaptability and regular updating of value functions enable them to remain competitive in diverse exchange environments.

Source: Own elaboration

2.3. Benefits in utilizing Deep Reinforcement Learning for Financial Challenges

According to Benhamou et al. (2023), DRL brings several significant advantages over traditional methods. These benefits make it a promising approach for portfolio management and investment decision-making (Benhamou, 2023).

A. Flexibility in Response to Shifting Conditions

The DRL model outperforms in adapting to dynamic market conditions. It can rapidly adjust its strategies and policies in response to shifting investment environments. This adaptability results in higher average Sharpe ratios compared to traditional methods, potentially capitalizing on emerging opportunities.

B. Independence from Conventional Financial Risk Assumptions

Unlike traditional methods that depend on specific risk assumptions and basic statistics about portfolio assets, such as average returns and variance, DRL is not restricted by these norms. It can consider a wider range of data and factors in its decision-making process. This adaptability allows the DRL model to uncover new insights from the data, eventually boosting its performance. By escaping from these traditional risk restrictions, DRL can easily adapt to varying market conditions and make more informed decisions about portfolio allocation.

C. Integration of Additional Data and Multi-Input Capability

One of DRL's distinctive features is its capacity to incorporate additional data that goes beyond the conventional financial metrics. This adaptability allows the model to take into account various inputs, such as macroeconomic indicators, market sentiment, or other pertinent data, which allows the model to effectively capture complex relationships and patterns that influence asset returns. This improved ability to handle multiple inputs improves the model's decision-making capacity and overall performance.

2.4. Obstacles in utilizing Deep Reinforcement Learning for Financial Challenges

According to Dulac-Arnold et al. (2021), there are a serious of obstacles which DRL faces in the world of financial applications. There are three main challenges: high-dimensional state spaces, long-term dependencies, and limited data (Osterrieder, 2023).

A. High-dimensional state spaces

Financial challenges often involve complex scenarios with many variables, such as prices, volumes, and returns of numerous assets which, consequently, create high-dimensional spaces with hundreds or thousands of dimensions. The curse of dimensionality happens when the rapid growth in dimensions exceeds the growth in available data samples. This leads to a decrease in sample efficiency and model generalization.

To deal with this, DRL algorithms use techniques such as *feature engineering*, which helps choose the most important sections of the data; *dimensionality reduction*, and *structured exploration* to extract relevant features and simplify the state space.

B. Long-term dependencies

In finance, difficulties can arise from complex long-term connections. For instance, in risk management, it is crucial for an agent to balance short-term and long-term risks, taking into account how present actions may impact future rewards and costs.

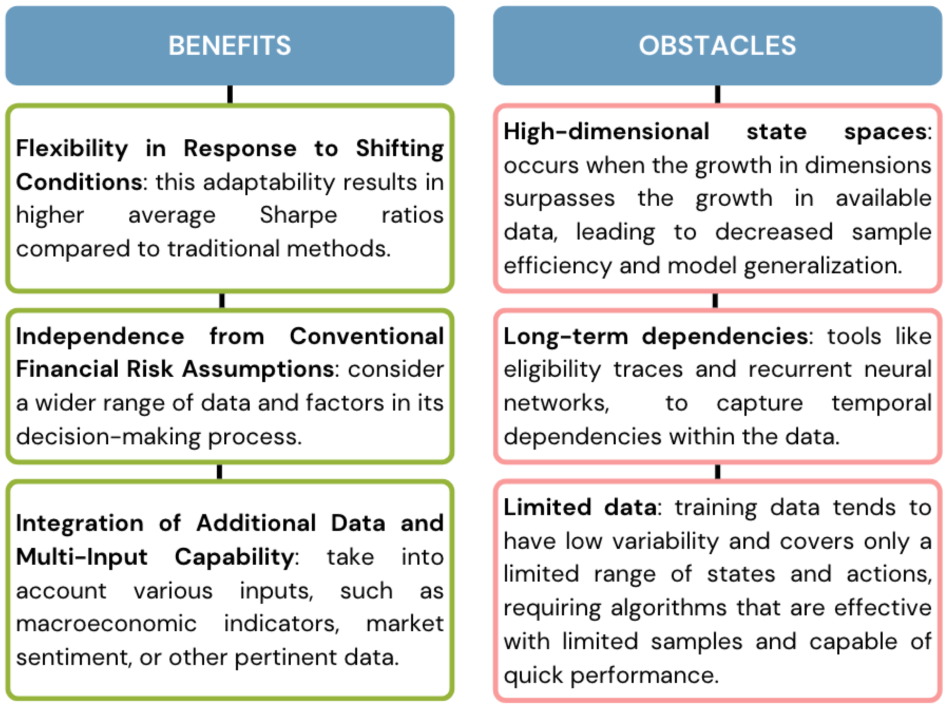
To address these long-term dependencies effectively, DRL algorithms use tools like *eligibility traces* and *recurrent neural networks*, which help capture temporal dependencies within the data.

C. Limited data

In many real-world systems, such as financial settings, collecting data can be difficult due to system limitations like slow movement, fragility, or high operating costs. Consequently, learning algorithms need to be highly efficient in using data. As these agents use actual system data, they cannot take excessively aggressive exploration policies during training since these actions have real-world consequences. Therefore, the training data tends to have low variability and covers only a limited range of states and actions. Consequently, learning from real systems requires algorithms that are both effective with limited samples and capable of quick performance.

Numerous financial problems involve data limitations, making it necessary for agents to learn and generalize from restricted datasets. These challenges include data scarcity, making it challenging to learn from small or biased datasets, and data efficiency, where a substantial number of samples are needed for optimal performance. To overcome these data-related issues, DRL algorithms use different techniques. For instance, *experience replay* stores and replays transitions to improve sample efficiency. *Transfer learning* utilizes knowledge from other tasks to improve performance in the given financial context. Additionally, methods like *data enrichment* and *active learning* are used to generate new data and select the most informative samples to facilitate efficient learning.

Figure 4. Future Opportunities for DRL in financial applications.


BENEFITS
OBSTACLES
Flexibility in Response to Shifting Conditions: this adaptability results in higher average Sharpe ratios compared to traditional methods.
Independence from Conventional Financial Risk Assumptions: consider a wider range of data and factors in its decision-making process.
Integration of Additional Data and Multi-Input Capability: take into account various inputs, such as macroeconomic indicators, market sentiment, or other pertinent data.
High-dimensional state spaces: occurs when the growth in dimensions surpasses the growth in available data, leading to decreased sample efficiency and model generalization.
Long-term dependencies: tools like eligibility traces and recurrent neural networks, to capture temporal dependencies within the data.
Limited data: training data tends to have low variability and covers only a limited range of states and actions, requiring algorithms that are effective with limited samples and capable of quick performance.

Source: Own elaboration

3. SCOPE OF THE THESIS

This thesis undertakes a comprehensive analysis of portfolio management in the current financial landscape, aiming to address emerging challenges and explore new frontiers. The main objectives seek to understand and apply Deep Reinforcement Learning (DRL) techniques in portfolio optimization, critically evaluating a DRL-based robotrading approach and addressing the weaknesses of traditional models such as Markowitz's Modern Portfolio Theory. It is proposed that the integration of Multi-fidelity Bayesian Optimization with DRL will enhance portfolio optimization, increase efficiency, and adaptability to constantly changing market conditions. The fundamental assumptions establish beliefs about market behavior and the utility of artificial intelligence techniques like DRL, while the restrictions validate temporal, technological, and data-related barriers encountered during the research process.

3.1. Objectives

There are several objectives which I am going to pursue in this thesis:

- **O1:** Realization of the document of my thesis.
- **O2:** Investigate how DRL can be used in portfolio optimization, specifically focusing on its ability to adapt to non-stationary market conditions and learn from both historical data and real-time feedback.
- **O3:** Critically assess the applicability and effectiveness of a robotrader employing Deep Reinforcement Learning (DRL).
- **O4:** Address the gap, which exits due to models such as Markowitz's Modern Portfolio Theory (MPT) which assumes stationary market conditions that may not hold in practice, by using DRL within the field of portfolio optimization.
- **O5:** Revolutionize the field of portfolio management and create substantial benefits for investors, financial institutions, and the broader financial markets.
- **O6:** Execute the experiment comparing different methods.

3.2. Hypothesis

The integration of Multi-fidelity Bayesian Optimization with Deep Reinforcement Learning (DRL) presents a promising path to improve portfolio management strategies in the dynamic landscape of financial markets. This section explores the potential advantages and improvements that this innovative approach might offer in optimizing investment strategies and effectively handling the complexities of diverse financial environments.

There is going to be a differentiation between the main hypothesis and the secondary ones:

A. Main Hypothesis:

- **H1:** Multi-fidelity Bayesian Optimization combined with DRL will display increased efficiency, reducing optimization costs and convergence time, consequently, enabling real-time decision-making in financial markets.

  The null hypothesis (H_0) and the alternative hypothesis (H_1) for the main hypothesis is:

Equation 1. Main Hypothesis.

$$H_0: \mu_t\_\text{multifidelity} < \mu_t\_\text{normal}$$
$$H_1: \mu_t\_\text{multifidelity} \geq \mu_t\_\text{normal}$$

B. Secondary Hypothesis:

- **H2:** Multi-fidelity Bayesian Optimization integrated with DRL will show superior performance, demonstrating improved portfolio optimization compared to conventional single-source optimization methods.
- **H3:** The use of Multi-fidelity Bayesian Optimization within DRL-based portfolio optimization will exhibit improved generalization across diverse financial environments, leading to adaptable and robust strategies in changing market conditions.
- **H4:** The integration of multi-fidelity data sources using Multi-fidelity Bayesian Optimization will improve decision-making accuracy in portfolio management, effectively mitigating biases from low-fidelity data sources.

- **H5:** Multi-fidelity Bayesian Optimization combined with DRL will demonstrate increased adaptability to changing market conditions, showing effective learning and adjustment in dynamic financial environments.

3.3. Assumptions

In this section, I am going to outline the key ideas I am starting with. These assumptions help set the boundaries for the research, showing what I am going to explore and how I am going to do it. By introducing these assumptions, this section aims to provide transparency, and the guiding of the study. Furthermore, the identification and comprehension of these assumptions are key to comprehend the study's limitations, strengths, and applicability in the landscape of portfolio management within financial markets.

- **A1:** We assume that financial markets present a stochastic behavior, requiring adaptable strategies for effective portfolio management.
- **A2:** Assumption that market information may be incomplete, creating challenges for accurate decision-making within portfolio management.
- **A3:** We need to assume that while complex models might replicate market behavior more accurately, they can be computationally challenging for practical implementation.
- **A4:** Deep Reinforcement Learning (DRL) methods can improve portfolio optimization through sequential learning and adaptation to changing market conditions.
- **A5:** Integrating Multi-fidelity Bayesian Optimization techniques offers advantages in managing expensive, complex, and black-box functions within portfolio optimization.
- **A6:** Optimization in portfolio management involves a trade-off between computational cost and the accuracy of information used.
- **A7:** Artificial Intelligence (AI) and machine learning techniques, particularly DRL, hold significant potential in addressing complex financial decision-making problems.
- **A8:** DRL models might face limitations such as overfitting, hyperparameter management, and the need for adaptation to volatile market conditions.

- **A9:** The fidelities of the estimated optimal policy are linearly correlated.

3.4. Restrictions

This section indicates the diverse restrictions which I have encountered during this thesis. These limitations come from different factors such as temporal and technological barriers and data availability issues. It is crucial to understand these limitations in order to provide clarity and set the boundaries within which the study operates.

- **R1:** Limited time to conduct research, gather data, analyze findings, and write the thesis.
- **R2:** Access to specific databases and technologies. For instance, being restricted to free versions of tools (google collab) and having limited access to certain libraries or datasets.
- **R3:** Constraints in computational power or infrastructure that limit the complexity and scale of computations and simulations.
- **R4:** Challenges in addressing a broad range of topics due to the need for depth and specificity within a limited time.

4. METHODOLOGY

This study adopts a mixed-methods approach, combining quantitative analysis with qualitative insights, to examine the effectiveness of Deep Reinforcement Learning (DRL) in portfolio optimization. The research framework is designed to offer a comprehensive understanding of the topic, covering both theoretical foundations and real-world implementations.

The research begins with an exploration of the current state of the art, examining existing literature and studies relevant to the topic. Platforms like Google Scholar offer an abundance of scholarly articles and research papers that contribute to this foundational stage. Following this, the theoretical framework is developed, illuminating the fundamental concepts critical to the subject matter. Finally, the research progresses to the experimental phase, where real-world data collected from the Dow Jones is utilized. This empirical approach adds depth and practical relevance to the study, offering insights derived from tangible market data. This methodical approach aims to connect theoretical insights with empirical evidence, facilitating a comprehensive understanding of the subject matter.

The study involves the development of a DRL-based portfolio optimization model using Python programming language. The flexibility and adaptability of the DRL framework allow for dynamic adjustments and iterative improvements throughout the modeling process. An experiment is conducted to evaluate the performance of the DRL-based portfolio optimization model. This experiment involves training the model on historical data and testing its effectiveness in real-time market scenarios. Performance metrics such as Sharpe ratio and the quality of time employed are used to assess the model's performance and compare it with baseline models.

The performance of the DRL-based portfolio optimization model is systematically compared with traditional portfolio optimization techniques, including Mean-Variance Optimization and Markowitz's Modern Portfolio Theory. This comparative analysis provides valuable insights into the relative strengths and weaknesses of DRL in portfolio management, highlighting its potential to outperform conventional approaches in certain contexts.

The study acknowledges certain limitations and delimitations, including constraints in data availability, computational resources, and the inherent complexity of financial markets. These limitations are taken into account when interpreting the research findings, providing a balanced perspective on the scope and applicability of the study.

By following this comprehensive methodology, the study aims to generate valuable insights into the application of Deep Reinforcement Learning in portfolio optimization, contributing to the advancement of knowledge in the field of financial technology and paving the way for innovative approaches to portfolio management.

5. THEORICAL FRAMEWORK

In order to achieve the various objectives, set in the previous section, I am going to make use of Deep Reinforcement Learning (DRL) for portfolio optimization. Furthermore, I am going to explore the use of multi-fidelity in the fine-tuning of parameters.

DRL is a type of machine learning that has achieved great suitability and effectiveness in portfolio optimization. It combines deep neural networks with reinforcement learning, enabling the robotrader to learn and adapt its trading strategy based on interactions with the financial markets (Osterrieder, 2023).

The decision-making process is based on the concept of rewards and feedback. The robotrader is continuously updating its strategies to maximize returns while minimizing risks. The use of DRL in portfolio optimization is driven by its capacity to adapt to dynamic market conditions and learn from historical data and real-time feedback.

Bayesian optimization (BO) is a technique used to optimize black-box functions systematically (Garrido, 2021). It utilizes a sophisticated model, often referred to as a Gaussian Process (GP), to estimate the objective function. BO progressively identifies potential solutions, refining its estimates with each iteration based on newly acquired information (Nguyen, 2019).

In certain scenarios, solving a problem requires information at varying levels of detail. For instance, in simulations, using more detailed data produces more accurate results but demands more computational resources. Multi-fidelity BO approaches manage this trade-off by integrating both quick, approximate evaluations and slower but more precise analyses (Li, Kirby & Zhe, 2021).

Despite their success, many methods often oversimplify the correlations between function outputs at different fidelities. Therefore, these methods may suffer from inefficiencies and inaccuracies, obstructing objective function estimation and, consequently, optimization efficiency.

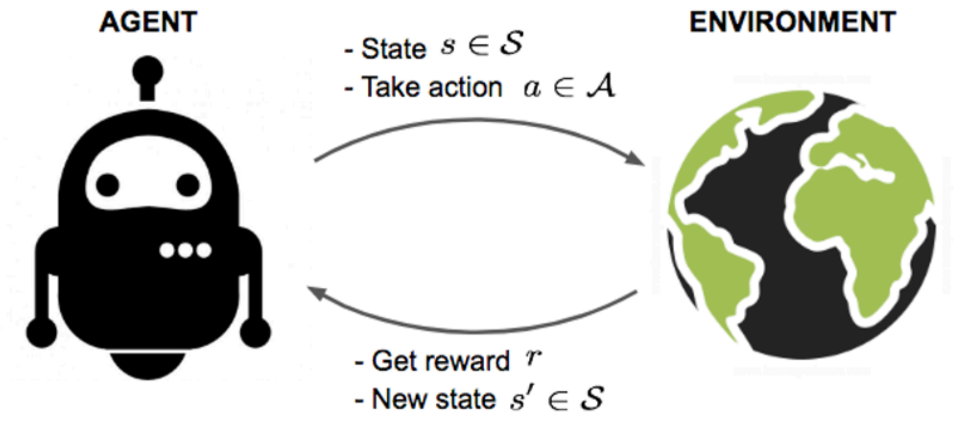
5.1. Introduction to Reinforcement Learning (RL)

Reinforcement Learning consists of a type of Machine Learning where an agent situated in an environment, learns to make decisions by taking actions in response to each state or situation in that environment. The agent learns through trial and error. For each action taken, the agent is going to receive feedback from the rewards. This is going to help the agent improve its actions in order to maximize the reward accumulated over time (Osterrieder, 2023).

The agent is going to explore different alternative actions, which are the moves made by an agent in a precise environment. By doing this, the information of the different outcomes is going to be accumulated. For each action taken, the agent is going to obtain rewards based on feedback, indicating the quality of the action taken. Lastly, the agent is going to make observations regarding what it perceives from the environment, which is going to help decide the next action.

The primary objective of an agent in Reinforcement Learning is to learn a policy, which consists of a function that guides its decision-making in each state it encounters. This involves the agent being in a specific state (s) at each time step and selecting an action (a) based on its policy. As a result of this action, the environment transitions to a new state, and in return, the agent receives a reward (r) (Mukherjee, Deligiannis, Biswas & Lal, 2020).

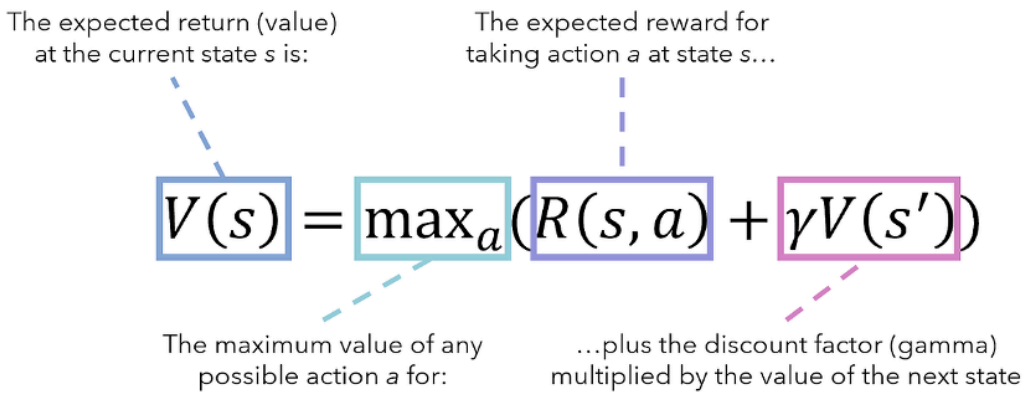
Figure 5. Reinforcement Learning Process.


AGENT
ENVIRONMENT
- State $s \in \mathcal{S}$
- Take action $a \in \mathcal{A}$
- Get reward r
- New state $s' \in \mathcal{S}$

Source: Zürn, 2018

The main goal of the agent is to learn a policy that maximizes the expected return, which is essentially the cumulative sum of rewards while considering the effects of discounting (Mukherjee, Deligiannis, Biswas & Lal, 2020). The discounting factor (γ) aims to penalize the rewards in the future due to the higher uncertainty in the future rewards and the fact that future rewards will not provide immediate benefits (Owen, 2020).

Figure 6. The value function.


The expected return (value) at the current state s is:
The expected reward for taking action a at state s...
$V(s) = \max_a(R(s,a) + \gamma V(s'))$
The maximum value of any possible action a for:
...plus the discount factor (gamma) multiplied by the value of the next state

Source: Ye, 2020

In the world of Reinforcement Learning, there are a variety of algorithms for problem-solving. These include value-based methods, which revolve around the estimation of the value function and represent the anticipated return associated with a particular state or a pair of state and action. On the other hand, policy-based methods take a more direct approach by learning the policy itself, eliminating the need for estimating the value function. Moreover, there are actor-critic methods that combine aspects of both value-based and policy-based techniques for effective problem-solving (Osterrieder, 2023).

Reinforcement learning differences itself from other types of machine learning in that it participates in a dynamic interaction between the agent and the environment. The agent's actions have the power to influence and change the subsequent state of the environment. This unique quality makes it great for situations where the agent needs to make choices over a long time, for instance, managing financial investments.

5.1.1. The Reinforcement Learning process

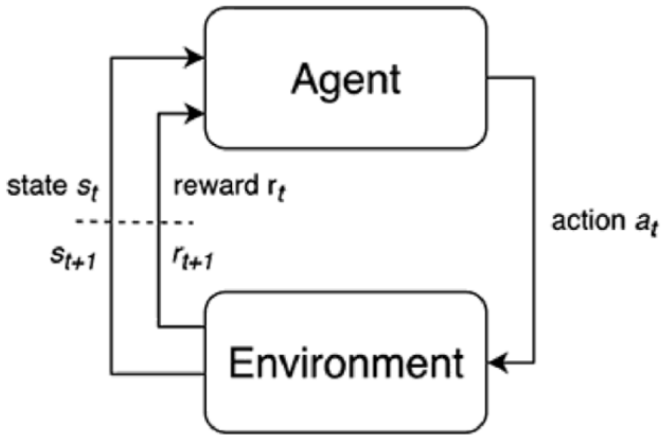
Reinforcement Learning is a dynamic system composed of two main components: an agent and an environment. The environment generates information that characterizes the current condition of the system, referred to as the "state." The agent engages with the environment by observing this state and using the information to make decisions, selecting an appropriate "action." The environment then responds to the action, transitioning into the next state, and providing both the new state and a "reward" back to the agent. Each time this sequence of *state → action → reward* occurs, we consider it a single time step. This cycle continues until the environment concludes its operation (Graesser & Keng, 2019*)*. You can visualize this entire process through the diagram in *Figure 7*.

We refer to the action-generating function of an agent as its "policy", which is a function that maps states to actions. Each action taken by the agent has the power to influence the environment, subsequently shaping what the agent perceives and how it proceeds. This dynamic interaction between the agent and environment continues over time, similar to a sequential decision-making process.

In Reinforcement Learning, problems are centered around a main objective: the cumulative sum of rewards an agent receives. The agent's primary aim is to maximize this objective by making optimal choices. It learns to do so through iterative interaction with the environment, utilizing a trial-and-error approach and reinforcing actions that lead to positive rewards (Rao & Jelvis, 2022).

A RL system is a feedback control loop, where the agent and the environment engage in a dialogue, sharing signals denoted as s_t (state), a_t (action), and r_t (reward), with "t" representing the time step at which these signals occur. This control loop can continue indefinitely or conclude when reaching a terminal state or a maximum time step, denoted as $t = T$. The duration from $t = 0$ until the environment's termination marks an "episode" (Omran et al., 2024). Learning a policy typically requires the agent to undergo numerous episodes, varying from hundreds to millions, depending on the problem's complexity (Graesser & Keng, 2019).

Figure 7. The reinforcement learning control loop.



Source: Graesser & Keng, 2019

5.1.2. Policy: on policy and off policy

In reinforcement learning, a **policy** is a critical element that guides how an agent should act in a given state. This strategy can take two different forms, the first one, a deterministic policy, where a specific action is defined for each state; or a stochastic policy, which involves a probability distribution over actions (Osterrieder, 2023). The main goal for the agent is to learn an optimal policy that maximizes the expected return, which represents the cumulative sum of rewards over time.

Reinforcement Learning also uses the **value functions** to calculate the expected return associated with specific states or state-action pairs. These functions come in two types: the state-value function *(v(s)),* which estimates the expected return for a particular state, and the action-value function *(q(s,a)),* that estimates the expected return for a state-action pair (Osterrieder, 2023).

In the RL control loop described earlier, a policy is crucial as it guides the agent's actions to accomplish its main objective. It is important to remind that policies can be stochastic, indicating that they might probabilistically generate different actions for the same state. Additionally, value functions offer insights into the goal, helping the agent in evaluating the desirability of states and available actions in terms of expected future returns.

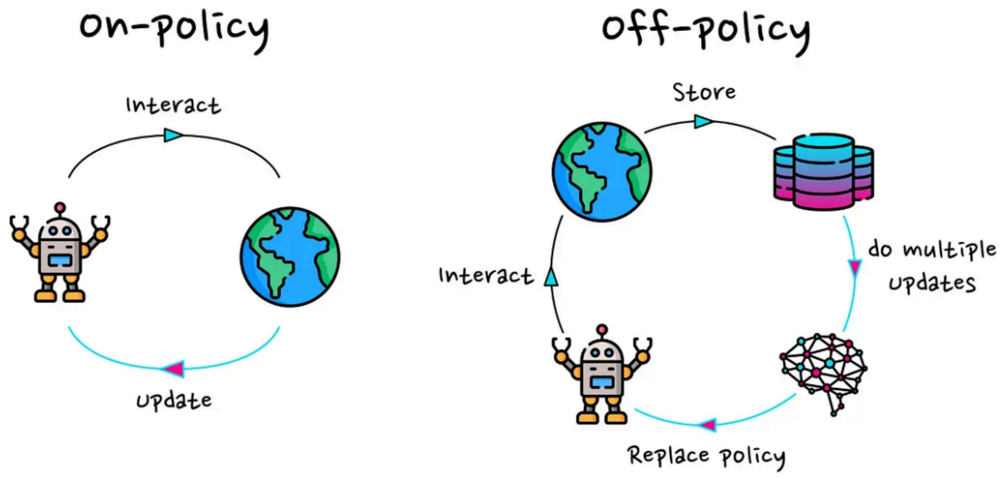
Reinforcement Learning algorithms use feedback from rewards to adjust and improve the policy for future decision-making. Policy updates often depend on the difference between predicted and actual rewards, fine-tuning how the agent makes decisions. These algorithms can be classified as either on-policy or off-policy, and this distinction has a significant impact on how they make use of training data.

<u>On-policy algorithms</u> learn directly from the current policy (π). This means that during training, they only use data generated by the current policy at that particular time. As the training advances through various policy versions, each iteration relies only on the currently active policy to collect training data, they are generally stable and easy to implement. However, this approach has a drawback: all collected data becomes outdated and cannot be reused. Consequently, on-policy methods are sample-inefficient and require more training data (Hammami & Nguyen, 2022).

In contrast, <u>off-policy algorithms</u> do not have this limitation. They can make use of any collected data in their training process. This property makes off-policy methods more sample-efficient since data can be reused (Hammami & Nguyen, 2022). However, it is important to mention that this advantage comes at the cost of potentially needing more memory to store the data. While these models are data-efficient, they may be less stable and more complex to handle.

In the on-policy scenario, the agent consistently interacts with the environment, updating its policy and adjusting its behavior based on newly collected experiences. On the other hand, in the off-policy scenario, an agent or multiple agents act according to a policy called the behavior-policy. The experiences they accumulate are stored in a buffer. Consequently, this collected experience is utilized to train a policy for action selection. Therefore, the buffer may contain experiences from previous versions of the policy or even from a completely different policy (Vandelaer, 2022).

Figure 8. On-policy vs Off-policy representation.



Source: Vandelaer, 2022

Both on-policy and off-policy models have their respective advantages and disadvantages, making it challenging to determine the most appropriate model for a given scenario. The choice between the two depends on the specific requirements and restrictions of the reinforcement learning problem.

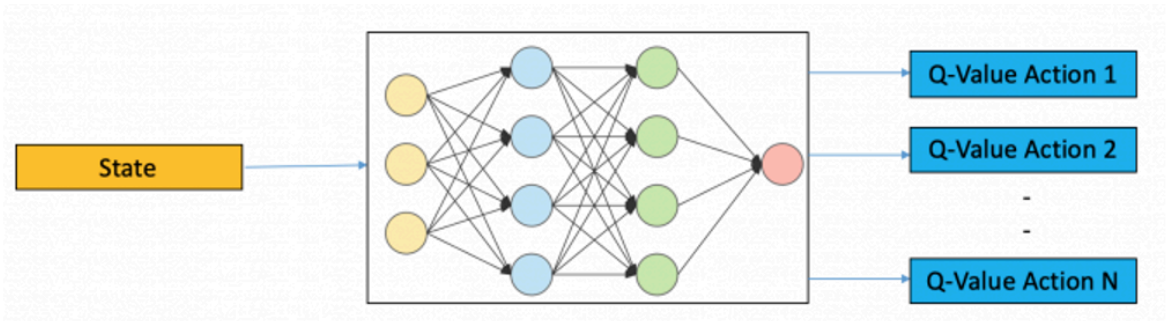
5.2. Introduction to Deep Learning (DL)

Deep Learning is a subfield of machine learning which enables computers to acquire knowledge from experience and comprehend the world through a hierarchical structure of concepts. As the computer acquires knowledge through experience, there is no requirement for a human computer operator to explicitly define all the information necessary for the computer. The hierarchical structure enables the computer to learn complex ideas by constructing them from simpler ones; a graph of this structure would be represented by multiple layers of depth (Kim, K. G., 2016).

The objective of DL is to learn a function that maps inputs to outputs using labeled training examples (Benhamou et al., 2023). This function is represented by a neural network, which computational models designed after the architecture and operations of biological neural networks found in the human brain (Coursesteach, 2023). DL models operate in layers, with a typical model consisting of at least three layers. Each layer receives input from the preceding layer and forwards it to the subsequent layer (Karunakaran, 2018).

Training the neural network involves adjusting the weights and biases of connections between neurons (Osterrieder, J. 2016). Initially, these weights are assigned random values. As the neural network gathers more information about the input data, it adjusts these weights according to any misclassifications or errors attributed to the previous weight settings (Karunakaran, 2018).

Figure 9. Deep Learning Representation.



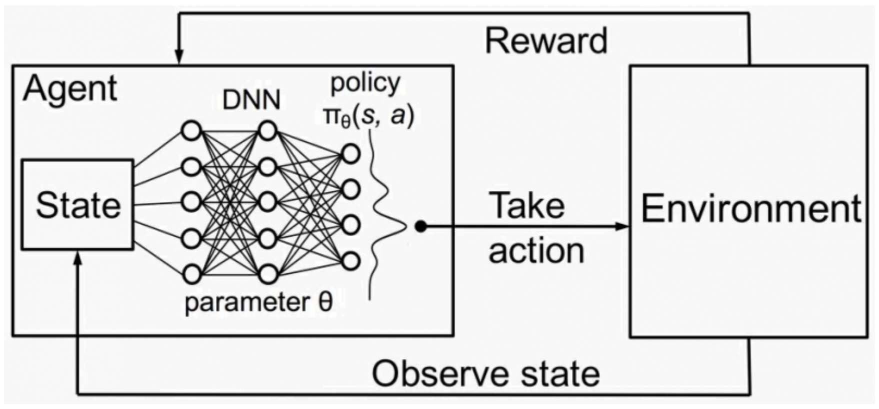
Source: Garrido, 2023

The process of adjusting network parameters through gradient descent becomes essential by utilizing a loss function to evaluate network outputs (Shrestha, A., & Mahmood, A., 2019). **Gradient descent** is an iterative optimization technique used in machine learning to minimize a loss function. The loss function provides insight into the model's performance based on the current parameters (weights and biases). By using gradient descent, we seek to identify the optimal set of parameters that minimizes the loss function, thus enhancing the model's overall performance. The iterative nature of this process involves continuously updating the model parameters through gradient descent to refine its predictive capabilities (Liu, C., 2022).

5.3. Introduction to Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning (DRL) uses the power of Deep Learning to address the challenges of Reinforcement Learning (RL). In DRL, agents use neural networks to represent the policy or value function, which, as previously mentioned, is a measure of expected cumulative rewards at each state (Durall, 2016). Deep neural networks are used to connect what the system senses to the values of actions or the chances of taking certain actions (Dulac-Arnold et al., 2019).

Figure 10. Deep Reinforcement Learning Process.



Source: Vijayan PV, 2020

Deep Reinforcement Learning can be divided into three primary algorithm families: policy-based, value-based, and model-based approaches. These approaches focus on learning policies, value functions, and models, respectively (Hambly, Xu & Yang, 2023). Nonetheless, DRL faces some challenges such as data volume, computational resources, and hyperparameter tuning. These challenges have led to the creation of new methods and techniques like off-policy learning, distributional RL, and model-based RL. These methods are designed to make the learning process more efficient and stable.

The agent's main objective is to learn a policy that maps states to actions, aiming to maximize the cumulative reward over time. This cumulative reward is determined by the Q-value function, which assesses the expected return when taking a specific action, *a,* in a given state, *s*, and following the policy, π, thereafter (Osterrieder, 2023).

To represent the Q-value function or the policy in DRL, a neural network is utilized. The Q-value function or the policy is represented by a neural network with many layers of interconnected nodes, called neurons. This network can understand complex patterns in the data. The neural network is trained using Deep Q-learning, an algorithm that estimates the Q-value for each state-action pair and adjusts the policy accordingly. This method helps the agent learn and adapt to various situations in its environment (Rao & Jelvis, 2022).

Deep Q-learning, a central algorithm in DRL, follows a structured process:

1) Initialize the neural network with random weights and biases.
2) Set up a replay buffer to store and sample transitions from the environment.
3) At each time step, observe the current state, select an action, and execute it using the current policy.
4) Record the reward and next state, storing the transition in the replay buffer.
5) Sample a batch of transitions from the replay buffer.
6) Compute the target Q-value for each transition using the Bellman equation:

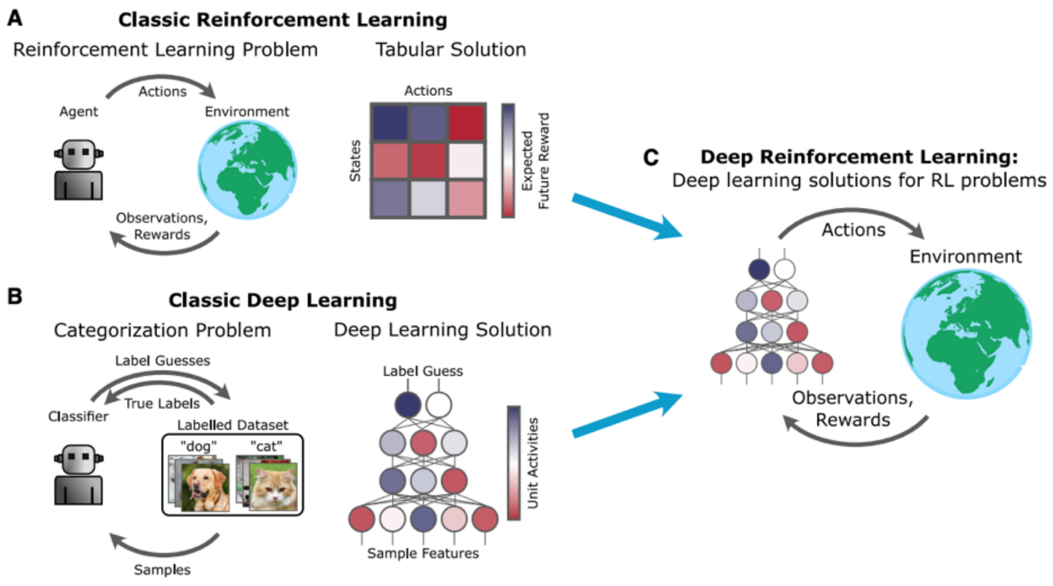
Equation 2. Bellman Equation.

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma * V_\pi(S_{t+1}) | St = s]$$

7) Update the neural network's weights and biases using Stochastic Gradient Descent (SGD) and the mean squared error loss.
8) Repeat steps 2 – 7 until convergence.

Deep Q-learning is an off-policy algorithm, allowing it to learn from transitions generated by different policies, making it more sample-efficient and stable compared to on-policy algorithms that require transitions generated by the same policy (Botvinick et al., 2020*)*.

Figure 11. RL, Deep Learning and Deep RL.



Source: Botvinick, M., et al., 2020

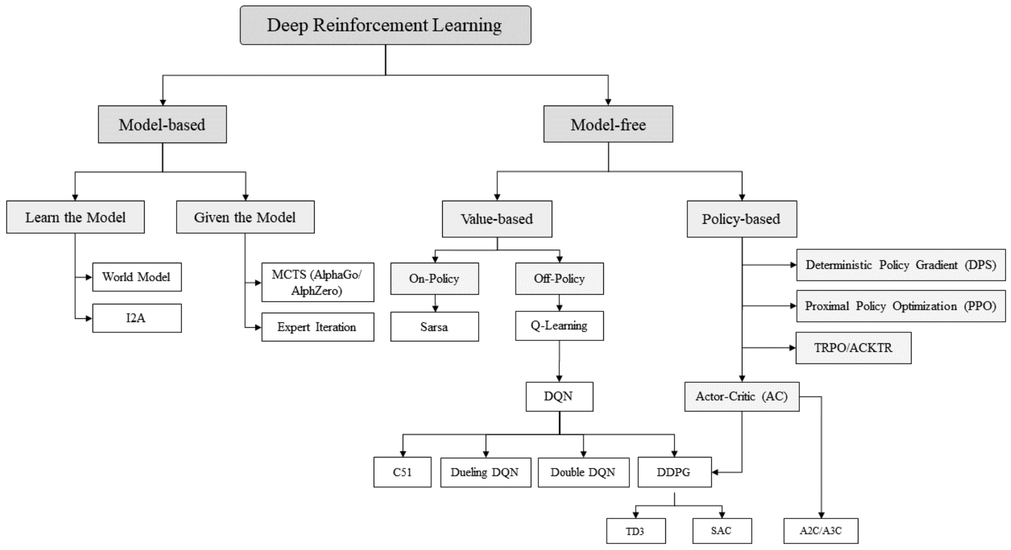
(A) On the left, we observe the Reinforcement Learning problem where the agent is tasked with selecting actions which later need to be transmitted to the environment. In return, the environment is going to provide the agent with observations and rewards. The goal of the agent is to choose the actions which maximize long-term rewards, which might not yield immediate benefits but could lead to a change in the environment's state, which would eventually result in rewards (Botvinick et al., 2020).

(B) On the left, we have the Supervised Learning problem where a sequence of unlabeled data samples, such as images, are received by an agent. Later, the agent makes educated guesses about the correct labels and immediate feedback on the correct label is provided (Botvinick et al., 2020).
On the right, we observe a Deep Leaning solution to the supervised learning problem. The characteristics of a sample go through multiple layers of artificial neurons where each neuron's activity is a weighted sum of its inputs, and its output is a non-linear function of this activity. The network's output corresponds to an estimated label of the sample. During the learning process, the network fine-tunes its weights to approximate the true labels. These solutions have proven effective at generalizing to samples that were not part of the training data (Botvinick et al., 2020).

(C) Deep Reinforcement Learning, where a neural network acts as an agent to address a reinforcement learning challenge. These solutions are really good at learning suitable internal representations which enable effective generalization to new states and actions (Botvinick et al., 2020).

5.3.1. Model-based and Model-free algorithms

Figure 12. Reinforcement Learning.



Source: Khamis & Wang, 2021

Reinforcement Learning methods can be categorized into two main groups: model-based and model-free approaches, where each offers different ways of optimizing policies.

In the **model-based approach**, agents try and understand the environment to later create a model for it by using their experience and interactions with this specific environment. This method exploits experience, as all the information from the environment is saved in a reliable and easy-to-handle way. This approach is effective for goal-oriented actions as it allows for swift adjustments to plans when transition conditions and outcomes change. This means that the model-based RL system can adapt rapidly to new circumstances and optimize its actions (Odemakinde, 2023).

Moreover, these algorithms provide agents with the capability to anticipate and simulate scenarios, which allows them to understand the consequences of their actions without direct interaction with the environment. This is an advantage in situations where acquiring experiences from the environment is either time-consuming or expensive. Additionally, these algorithms generally require fewer data samples for efficient policy learning as they can incorporate simulated experiences at the same time that real ones (Dulac-Arnold et al., 2019).

Nonetheless, there are some challenges when models are hard to obtain, as many environments are stochastic with unknown transition dynamics. In these cases, model-based algorithms must learn the model. This approach is still in its early stages of development and faces several challenges. First, representing environments with many different states and possible actions can be very complex and might even be impossible, especially when transitions are highly complicated. Second, the use of models depends on their capacity to make precise predictions about how the environment will change many steps into the future. Depending on how accurate the model is, errors in predictions can add up at each time step, making the model less reliable (Dulac-Arnold et al., 2019).

In the **model-free approach**, agents learn to optimize policies without creating a world model. Instead, they focus on two key elements: state-action values or policies learned directly from experience, in order to achieve optimal behavior without estimating or using a world model. Model-free methods, like temporal difference (TD) learning, identify prediction errors and use them to refine value estimates, minimizing inconsistencies. These values guide policy improvements by selecting actions leading to higher rewards and more valuable states. These values must satisfy a particular set of consistency conditions, with a state's value being high if the actions guided by the policy lead to favorable immediate outcomes. Some model-free methods even improve policies without acquiring values (Botvinick et al, 2020).

However, model-free methods are statistically less efficient compared to model-based method as they mix information from the environment with previous, and potentially inaccurate, estimates or beliefs about state values rather than utilizing it directly. For this reason, model-free RL are more suitable when working as a model for habitual actions (Dayan & Niv, 2008).

5.3.1.1. Types of model-free algorithms

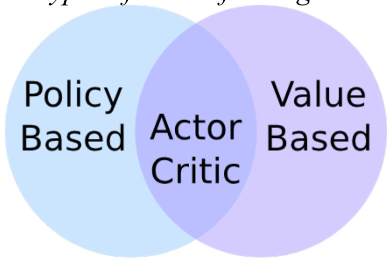
We can find three types of model-free algorithms: value-based, policy-based and actor-critic algorithms. Each algorithm offers a distinct approach to solving RL problems, with unique advantages and limitations.

- **Value-based algorithms** focus on estimating state values or state-action values. Dynamic programming approaches, such as value iteration and policy iteration, break down the problem into subproblems and iteratively compute the value function. These algorithms outperform in sample efficiency and data utilization. However, they do not guarantee convergence to an optimal solution and were historically limited to discrete action choices (Osterrieder, 2023).

- **Policy-based algorithms**, on the other hand, directly optimize the policy without the need for value function estimation. Techniques like policy gradient methods use gradient ascent to improve policy parameters based on expected returns. These methods offer flexibility in handling various types of actions, assuring convergence to locally optimal policies, but they require sensitive hyperparameter tuning (Osterrieder, 2023).

- **Actor-critic algorithms** work as a balance of both previous algorithms as it combines aspects of both value-based and policy-based methods. They simultaneously learn a value function to critique the policy and leverage the value function to enhance the policy itself (Osterrieder, 2023). This approach maintains the strengths of both value-based and policy-based methods, providing stability and sample efficiency (Khdoudi et al., 2024).

Figure 13. Types of model-free algorithms.



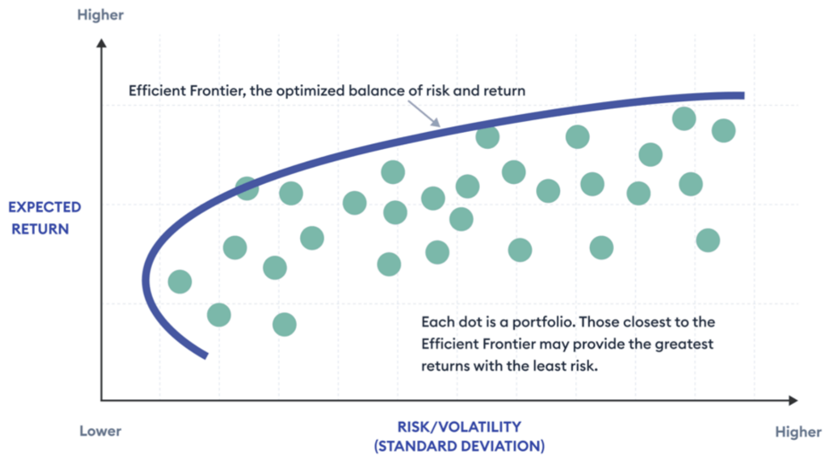
Source: Siegel, 2020

5.4. Markowitz's Modern Portfolio Theory vs. Deep Reinforcement Learning

Portfolio optimization is a strategy used to select and trade assets in order to maximize returns while managing risks. Diversification plays a crucial role as it offers superior returns per unit of risk compared to investing in a single asset.

Modern Portfolio Theory (MPT) is built upon Markowitz's Mean-Variance Optimization model, and it looks to maximize returns while maintaining an acceptable level of risk. Despite its effectiveness, it relies on historical data which introduces uncertainties regarding future accuracy. MPT is structured around a single-period investment model, and it quantifies risk through volatility (Durall, 2022).

Figure 14. Modern Portfolio Theory Efficient Frontier.



Source: Baldridge, 2023

Deep Reinforcement Learning (DRL) introduces a new dimension to portfolio management. While Markowitz relies on mathematical models and statistical techniques to determine an optimal portfolio based on historical returns and variances, DRL introduces a dynamic and learning-based methodology (Benhamou et al., 2023).

In DRL, neural networks are employed to optimize portfolios through a combination of value-based and policy-based methods. The algorithm learns and adapts its trading strategy by interacting with the financial markets, making it more adaptive to changing market conditions. This is a great difference from the static nature of the Markowitz model, which is based on fixed statistical parameters and assumes that historical trends will continue into the future.

DRL's adaptability allows it to respond to stochastic events and market fluctuations, making it potentially more resilient in unpredictable financial environments. While DRL introduces challenges in terms of interpretability due to its learning from data, its stability can be enhanced by incorporating additional data or technical indicators into the model.

There are several assumptions of Markowitz regarding the market which Deep Reinforcement Learning does not include:

- <u>Assumption of normal distribution</u>: the traditional framework assumes that asset returns follow a normal distribution, which is not always the case (FasterCapital, n.d.). Real-world events can lead to extreme negative returns, distorting the distribution. This can underestimate the overall risk of the portfolio (Sam Obeidat, 2018).

- <u>Single-Period Framework</u>: the traditional method assumes that asset allocations are made once at the beginning of a period and cannot be modified until the end. This single-period approach fails to accommodate multi-period objectives. Numerous studies have examined the single-period problem and concluded that, under certain assumptions, the multi-period problem can be addressed by a series of single-period problems. However, this approach would yield a different optimal portfolio compared to what is obtained using the single-period model. (Sam Obeidat, 2018).

- Market efficiency: the MPT assumes that the market is efficienct, implying that all available information is already reflected into the securities' price. Nonetheless, this assumption has been challenged by numerous researchers who argue that the market is not always efficient. For instance, sudden market shifts, like natural disasters or political events, may not be quickly reflected in security prices (Mangram, 2013).

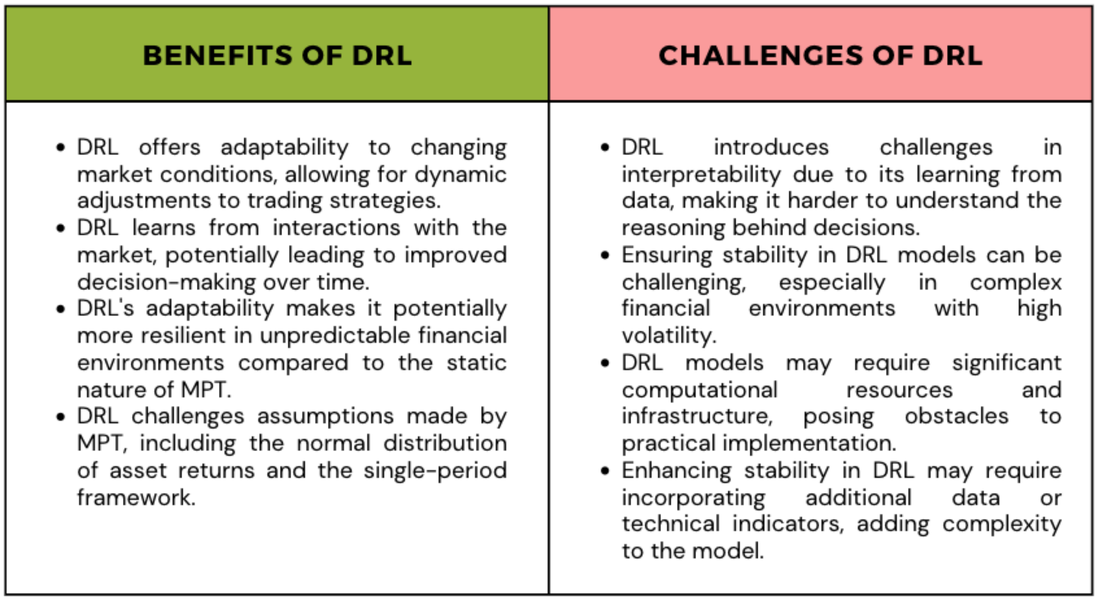
- Absence of transaction costs in markets: the MPT overlooks transaction costs, including brokerage fees and taxes, which can notably diminish an investor's returns. When making investment decisions, it's crucial to consider these costs, as they can substantially impact overall profitability. Therefore, the Markowitz Efficient Set may not accurately predict future security performance, as it fails to incorporate these significant expenses associated with trading (Mangram, 2013).

- Independence of returns: It is assumed that securities can be chosen in a way that their individual performance remains unaffected by the performance of other investments within the portfolio. However, in reality, the interdependence between securities can't always be ignored, especially during times of market volatility or economic uncertainty (Mangram, 2013).

A transition is suggested from the traditional Markowitz method to a more dynamic and learning-based approach using DRL. This shift involves considering returns and variances more accurately and treating the optimization process as a sequential, step-by-step learning challenge that adapts to evolving market conditions (Graesser & Keng, 2019).

Below is a summary figure illustrating the benefits and obstacles of DRL compared to MPT:

Figure 15. Benefits and Obstacles of Deep Reinforcement Learning (DRL) Compared to Modern Portfolio Theory (MPT)



BENEFITS OF DRL	CHALLENGES OF DRL
• DRL offers adaptability to changing market conditions, allowing for dynamic adjustments to trading strategies.	• DRL introduces challenges in interpretability due to its learning from data, making it harder to understand the reasoning behind decisions.
• DRL learns from interactions with the market, potentially leading to improved decision-making over time.	• Ensuring stability in DRL models can be challenging, especially in complex financial environments with high volatility.
• DRL's adaptability makes it potentially more resilient in unpredictable financial environments compared to the static nature of MPT.	• DRL models may require significant computational resources and infrastructure, posing obstacles to practical implementation.
• DRL challenges assumptions made by MPT, including the normal distribution of asset returns and the single-period framework.	• Enhancing stability in DRL may require incorporating additional data or technical indicators, adding complexity to the model.

Source: Own Elaboration

5.5. Bayesian Optimization for Hyperparameter Tuning

When talking about machine learning, fine-tuning learning parameters and model hyperparameters is a common practice, often relying on expert intuition, rule-of-thumb guidelines, or exhaustive trial-and-error methods. Consequently, there's a growing interest in automated techniques capable of optimizing learning algorithms to match specific problem requirements. Here is where Bayesian Optimization comes into play as it is a framework that models a learning algorithm's performance using a Gaussian process (GP). Selecting appropriate GP characteristics, such as kernel type and hyperparameter treatment, significantly impacts the effectiveness of the optimization process (Snoek, Larochelle & Adams, 2012).

Bayesian Optimization is a strategic method used when evaluating functions is expensive, either due to significant computational resources or financial expenses. The goal of this method is to achieve the best possible outcome from these expensive functions while minimizing the number of evaluations, significantly reducing both time and cost (Barsce et al., 2017). It establishes an initial belief, or prior, regarding the optimization function and incorporates information from past samples to refine this belief into a posterior. Later, a utility function is employed to determine the next sample point, aiming to maximize the optimization function (Wu et al., 2019).

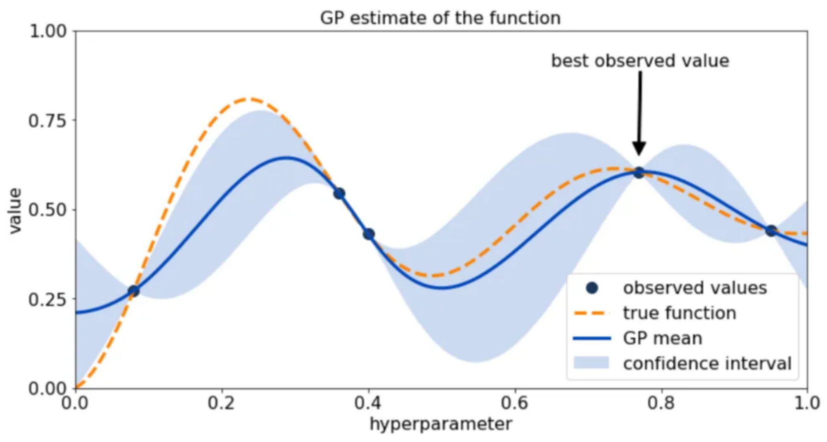
It plays a crucial role in fine-tuning machine learning algorithms by intelligently selecting hyperparameter values. The process begins with the initial random selection of hyperparameters, serving as initial data points. Subsequently, a balance is stablished between two strategic approaches (Run.ia, 2024):

- <u>Active Learning</u> (Exploitation): This strategy focuses on selecting points with the highest uncertainty in each iteration, aiming to exploit regions with the potential for optimal results.

- <u>Best Objective Function</u> (Exploration): This approach prioritizes selecting points from regions currently exhibiting the best results, thus exploring areas with the potential for further optimization.

For instance, in a maximization problem scenario, the Bayesian Optimization method executes the algorithm with various random hyperparameter values in each iteration. It then decides whether to prioritize <u>exploitation</u>, by selecting the point with the maximal result, or <u>exploration</u>, by opting for the point with the highest uncertainty, indicating potential for improved outcomes (Run.ai, 2024).

Figure 16. Bayesian Optimization Process.



Source: Gomede, 2024

This decision-making process is governed by several key functions (Run.ia, 2024):

- Upper Confidence Bound (UCB): This function selects the next point based on the highest upper confidence bound, calculated using the mean (μ), standard deviation (σ), and an exploration parameter (κ).

Equation 3. Upper Confidence Bound.

$$UCB(x)=\mu(x)+\kappa\sigma(x)$$

- Probability of Improvement (PI): Here, the next point is chosen based on its potential for improvement compared to the current maximum objective function (fmax), factoring in a trade-off parameter (ε) to balance exploration and exploitation.

Equation 4. Probability of Improvement.

$$PI(x)=\Phi(\mu(x)-fmax-\varepsilon\sigma(x))$$

- Expected Improvement (EI): Quantifying the expected improvement achieved by new points, this function selects the point with the highest expected value, while also considering the exploration-exploitation trade-off parameter (ε) in its calculation.

Equation 5. Expected Improvement.

$$EI(x)=(\mu(x)-fmax)\Phi(\mu(x)-fmax-\varepsilon\sigma(x))+\sigma(x)\phi(\mu(x)-fmax-\varepsilon\sigma(x))$$

Rather than directly analyzing the costly function, the approach is based on the construction of a simpler model for the unknown objective function, known as the black-box function. An "acquisition function" derived from this model guides the selection of the next area to explore, balancing between exploring new regions and exploiting the promising ones. The acquisition function drives this decision-making process, allowing for a balance between rapid convergence and comprehensive exploration. This strategy facilitates a faster discovery of the best outcome (Nguyen, 2019).

Overall, Bayesian Optimization acts as an intelligent approach to handling costly functions. It involves using a simplified model to determine the next steps, proving to be a pivotal tool in computer learning, greatly optimizing the learning process.

5.6. Multi-Fidelity Optimization in DRL

Multi-fidelity approaches in optimization aim to reduce computational costs by using low-fidelity approximations for certain evaluations. This strategy involves initially using less resource-intensive evaluations to identify promising parameters. Subsequently, more resource-demanding evaluations focus on improving the estimations within this smaller group (Wu et al., 2020).

These methods control fidelity by adjusting training iterations, training data points, or validation data points during validation error approximations. This provides clear advantages for iteratively trained machine learning models. They offer a detailed performance record over training iterations, allowing to significantly reduce computation time. This enables the rapid evaluation of low-fidelity approximations for different hyperparameter settings, followed by more accurate observations for the best options by increasing iterations. Moreover, they offer flexibility in adjusting fidelity through various methods, providing more options compared to adjusting a single control or choosing from limited accuracy options (Fare et al., 2022).

In order for the DRL-based robotrader to be successful, we need effective hyperparameter tuning. There are numerous parameters involved in DRL models, so it is necessary to optimize them in order to achieve an optimal performance. To do so, we introduce multi-fidelity optimization. This method involves using various fidelity levels to efficiently look for the optimal hyperparameters. It balances the trade-off between computational cost and optimization accuracy. Hyperparameter optimization for DRL models is a crucial task as it is critical to find a more efficient way to explore the hyperparameter space while still achieving a high-quality solution.

Multi-fidelity optimization provides an effective solution to this challenge by offering a structured approach to hyperparameter tuning. It starts by using lower fidelity in the early stages of hyperparameter tuning to explore the hyperparameters more efficiently. As the optimization process progresses, we will gradually introduce higher fidelity models to refine the hyperparameters which are going to provide more precise evaluations of the hyperparameters. This approach is going to speed up the optimization process without affecting negatively to the quality of the final solution (Wu et al., 2020). Multi-fidelity optimization helps the robotrader fine-tune its hyperparameters in a more effective way, adapting to the specific needs of the financial markets. This makes the DRL-based robotrader more flexible to the changing financial markets.

By using multi-fidelity optimization, the robustness and adaptability of the portfolio management strategy is increased. This approach helps in the process of optimization of hyperparameters, leading to improved trading strategy performance. Not only is the strategy going to perform better in terms of risk-adjusted returns, but it will also result in a higher degree of flexibility, cost-efficiency, and adaptability to market dynamics. This can ultimately translate to more successful and sustainable portfolio management practices in the dynamic and changing landscape of financial markets.

6. EXPERIMENT

In this chapter, some experiments will be conducted to investigate the hypotheses presented and reach the proposed objectives. However, it is important to note that these experiments will be limited in scope due to various restrictions such as budget limitations, time constraints, and technological resources availability. As a result, the outcomes of the experiments will provide a specific estimation within the limitations of these restrictions. The data collected for the experiment will come from the 30 constituents of the Dow Jones.

6.1. Experiment description

The experiment will consist of optimizing the hyperparameters of the Proximal Policy Optimization (PPO), which is a reinforcement learning algorithm designed to train a computer agent's decision-making function to address challenging tasks. The optimization is carried out by utilizing different fidelities of the estimated optimal policy (maximization of the Sharpe ratio in the trading period) by these algorithms. The fidelities are going to be a function of the timesteps and, according to A9, it is assumed that these fidelities are linearly correlated. We conduct a thorough comparison between different methods: Random Search and traditional Bayesian Optimization against multi-fidelity Bayesian Optimization.

The key idea is that hyper-parameter tuning of DRL algorithms is a very costly process, however, it is critical. Bad hyper-parameter values will not deliver a good policy in practice. Moreover, the financial context differs from the robotics and videogames context where these algorithms are usually trained, and their default values are set. Consequently, we expect that their hyper-parameter values will change significantly. Hence, we need to design a cheaper process to obtain those hyper-parameter values, and this process is multi-fidelity Bayesian Optimization.

In order to do so, we are going to use FinRL-StableBaselines3 for DRL and for multi-fidelity Bayesian Optimization we will use the package Dragonfly, which contains examples of multi-fidelity optimization.

We establish the null hypothesis (H_0) that multi-fidelity should yield better results than random search and take less time than all other methods. The alternative hypothesis (H_1)

suggests otherwise. The main objective is to maximize the Sharpe ratio. However, the experiments could also be made with the accumulated return or the Sortino ratio. While traditional Bayesian Optimization prioritizes high fidelity, multi-fidelity seeks to optimize the process by initially exploring options with lower fidelity and gradually eliminating poor-performing ones. This approach saves computational resources by focusing on promising options and eventually refining them with higher fidelity. Moreover, the training period which is going to be used for the experiments is from 2008 to 2022; and the test period is going to be 2023.

Our ideal strategy involves allocating the same amount of time to traditional Bayesian Optimization, multi-fidelity Bayesian Optimization and Random Search methods. Performance and time are considered as key parameters. The goal is not necessarily to outperform the traditional Bayesian optimization method but to achieve good results in less time.

Initially, we train all methods with a small number of timesteps. The best-performing models from each method are then further trained with a higher number of timesteps, while the remaining models are trained entirely with high timesteps. Each timestep represents a trading day, and to ensure convergence, the market needs to be navigated millions of times. A comparison of the performance of multi-fidelity Bayesian Optimization with traditional Bayesian Optimization and with Random Search is going to be carried out. 10 iterations are going to be considered to refine the hyperparameters and a low (relative) number of timesteps to make the experiment possible.

However, the experiment faces constraints such as limited funding and time availability. These constraints may impact the comprehensiveness and duration of the experiment, potentially affecting the accuracy and reliability of the results.

Further experiments would require the comparison of these methods with respect to genetic algorithms, more indexes (for instance, NASDAQ) more DRL algorithms (for example, A3C) and more repetitions.

6.1.1. Problem Definition

This problem involves designing an automated trading system for portfolio allocation. The approach consists of modeling the stock trading process as a Markov Decision Process (MDP) and framing the trading objective as a maximization problem.

The algorithm is trained using Deep Reinforcement Learning (DRL) algorithms and the key components of the reinforcement learning environment include:

1. Action: The action space defines the allowed actions the agent can take in the environment. Typically, denoted as '$a \in A$', represents the weight of a stock in the portfolio, with values ranging from -1 to 1: $a \in (-1,1)$. Assuming our stock pool includes N stocks, we can use a list $[a_1, a_2, \ldots, a_N]$ to determine the weight for each stock in the portfolio, where $a_i \in (-1,1)$, $a_1 + a_2 + \ldots + a_N = 1$. For instance, "The weight of AAPL in the portfolio is 10%." is [0.1 , ...] (FinRL, 2021).

2. Reward function: it is denoted as $r(s, a, s')$, and it is the incentive mechanism for an agent to learn a better action. It calculates the change of the portfolio value when action a is taken at state s and arriving at new state s', i.e., $r(s, a, s') = v' - v$, where v' and v represent the portfolio values at state s' and s, respectively (FinRL, 2021).

3. State: The state space describes the observations received by the agent from the environment. Similar to a human trader analyzing various information before making a trade, the trading agent observes many different features to better learn in an interactive environment (FinRL, 2021).

4. Environment: Dow Jones 30 constituents serve as the environment for this trading system (FinRL, 2021).

The data of the single stock used in this case study is obtained from Yahoo Finance API, including Open-High-Low-Close price and volume.

6.1.1. Python Packages Load

Before carrying out the analysis, the initial step involves installing the necessary libraries and packages to facilitate the development and implementation of an automated trading system.

Following the library installations, the next phase involves ensuring that all essential packages are present for the project. These include the Yahoo Finance API, pandas, numpy, matplotlib, stockstats, OpenAI gym, stable-baselines, tensorflow, and pyfolio. If any of these packages are missing, we install them to make sure our development environment is complete and fully functional.

Once the packages are in place, the next step involves importing the required modules such as pandas, numpy, and matplotlib for data manipulation and visualization. The 'YahooDownloader' and 'FeatureEngineer' modules from the 'FinRL' library are also imported, along with modules related to environment setup, agent modeling, and result visualization.

To organize the project and manage data, folders are created using the 'os' module. The folders include directories for data storage, trained model output, and result summaries. This organizational step helps maintain a structured approach, ensuring we manage data and track results in a structured way throughout the thesis project.

6.1.2. Data Download

The data has been retrieved from Yahoo Finance, which is a website that provides stock data, financial news, financial reports, etc. All the data provided by Yahoo Finance is free.

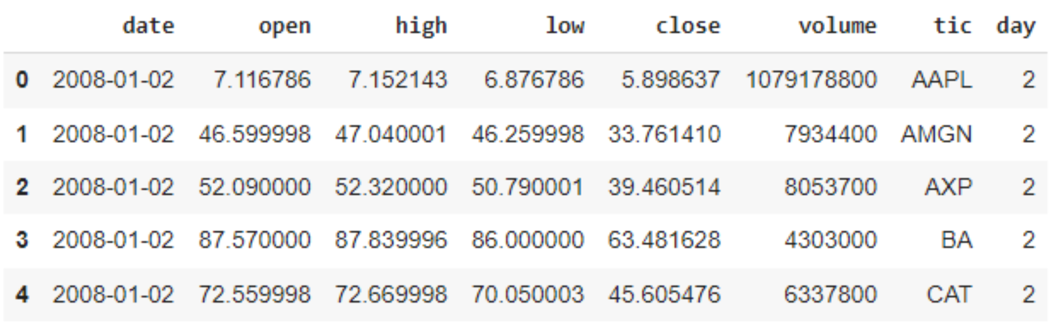
- FinRL uses a class YahooDownloader to fetch data from Yahoo Finance API (FinRL, 2021).
- Call Limit: by using the Public API (without authentication), you are limited to 2,000 requests per hour per IP (or up to a total of 48,000 requests a day) (FinRL, 2021).

From the downloaded data, for both the training and the test period, we obtain several variables:

- Date: this column represents the date of the recorded data. For each date recorded, there is data for every company.
- Open: it refers to the "opening" price which is the price of the stock from the first transaction made in a business day when the market opens.
- High: the "high" price represents the highest price observed during the business hours of trading on a specific date.
- Low: the "low" price represents the lowest price reached during the business hours of trading on a specific date.
- Close: it refers to the "closing" price which is the last price anyone paid for a stock trade during a business day where that stock trades, in this case the Dow Jones.
- Volume: it is the total number of shares traded during a specific trading session. It represents the liquidity and activity in the market.
- Ticker Symbol (Tic): each company is associated with a unique ticker symbol.

Here is an example of the variables exposed:

Figure 17. Variables used for the experiment.



	date	open	high	low	close	volume	tic	day
0	2008-01-02	7.116786	7.152143	6.876786	5.898637	1079178800	AAPL	2
1	2008-01-02	46.599998	47.040001	46.259998	33.761410	7934400	AMGN	2
2	2008-01-02	52.090000	52.320000	50.790001	39.460514	8053700	AXP	2
3	2008-01-02	87.570000	87.839996	86.000000	63.481628	4303000	BA	2
4	2008-01-02	72.559998	72.669998	70.050003	45.605476	6337800	CAT	2

Source: Own elaboration

6.1.3. Preprocess Data

Data preprocessing is a crucial step for training a high-quality machine learning model. It involves addressing missing data and performing feature engineering in order to transform the data into a state suitable for modeling.

Technical indicators are added to improve the dataset, where factors such as historical stock prices, current holding shares, and specific indicators are considered. Additionally, a turbulence index is introduced. Risk aversion plays a key role in an investor's decision-making process as it influences their choice to preserve the capital or not. Furthermore, it also influences one's trading strategy in response to varying market volatility. To manage risk in highly volatile markets, such as the financial crisis of 2007-2008, FinRL utilizes a financial turbulence index which measures extreme asset price fluctuation in order to effectively assess the level of market turbulence (Liu et al., 2024).

Once the technical indicators are included, there are some new variables to analyze:

- Macd (Moving Average Convergence Divergence): A trend-following momentum indicator that shows the relationship between two moving averages of a security's price.

Equation 6. Moving Average Convergence Divergence.

$$\text{MACD} = \text{12-Period EMA} - \text{26-Period EMA} \quad (1)$$

- Boll_ub and Boll_lb: Upper and lower bands of the Bollinger Bands, respectively, representing volatility around a moving average.

Equation 7. Upper and Lower Bollinger Band.

$$\text{BOLU} = \text{MA}(\text{TP}, n) + m * \sigma[\text{TP}, n] \quad (1)$$

$$\text{BOLD} = \text{Lower Bollinger Band} \quad (2)$$

Where:

$\text{TP (typical price)} = (\text{High} + \text{Low} + \text{Close}) \div 3$ (3)

BOLU = Upper Bollinger Band

BOLD = Lower Bollinger Band

MA = Moving average

n = Number of days in smoothing period (typically 20)

m = Number of standard deviations (typically 2)

$\sigma[\text{TP},n]$ = Standard Deviation over last n periods of TP

- <u>Rsi_300 (Relative Strength Index over 300 days):</u> A momentum oscillator measuring the speed and change of price movements. The calculation of this indicator is separated in two formulas:

Equation 8. Relative Strength Index.

$$RSI_{\text{step one}} = 100 - \left[\frac{100}{1+\frac{\text{Average gain}}{\text{Average loss}}}\right] \quad (1)$$

$$RSI_{\text{step two}} = 100 - \left[\frac{100}{1+\frac{(\text{Previous Average Gain}\times 13) + \text{Current Gain}}{((\text{Previous Average Loss}\times 13) + \text{Current Loss})}}\right] \quad (2)$$

- <u>Cci_30 (Commodity Channel Index over 30 days):</u> An oscillator used to identify cyclical trends in a security's price.

Equation 9. Commodity Channel Index.

$$\text{CCI} = \frac{\text{Typical Price} - \text{MA}}{.015 \times \text{Mean Deviation}} \quad (1)$$

Where:

$\text{Typical Price} = \sum_{i=1}^{P}((\text{High} + \text{Low} + \text{Close}) \div 3)$ (2)

P = Number of periods

MA = Moving Average

$\text{Moving Average} = (\sum_{i=1}^{P} \text{Typical Price}) \div P$ (3)

$\text{Mean Deviation} = (\sum_{i=1}^{P} \mid \text{Typical Price} - \text{MA} \mid) \div P$ (4)

- <u>Dx_30 (Directional Movement Index over 30 days):</u> A trend strength indicator.

Equation 10. Directional Movement Index.

$$+\text{DI} = \left(\frac{\text{Smoothed +DM}}{\text{ATR}}\right) \times 100 \quad (1)$$

$$\text{-DI} = \left(\frac{\text{Smoothed -DM}}{\text{ATR}}\right) \times 100 \quad (2)$$

$$\mathrm{DX} = \left(\frac{|\,+\mathrm{DI} - \text{-DI}\,|}{|\,+\mathrm{DI} + \text{-DI}\,|}\right) \times 100 \quad (3)$$

Where:

$$+\mathrm{DM}\ (\text{Directional Movement}) = \text{Current High} - \mathrm{PH} \quad (4)$$

PH=Previous high

$$\text{-DM} = \text{Previous Low} - \text{Current Low} \quad (5)$$

$$\text{Smoothed } +/\text{-DM} = \textstyle\sum_{t=1}^{14} \mathrm{DM} - \left(\frac{\sum_{t=1}^{14} \mathrm{DM}}{14}\right) + \mathrm{CDM} \quad (6)$$

CDM=Current DM

ATR=Average True Range

- Close_30_sma (30-day Simple Moving Average of closing prices): A smoothed average of the last 30 closing prices.

Equation 11. Simple Moving Average.

$$\mathrm{SMA} = \frac{A_1 + A_2 + \ldots + A_n}{n} \quad (1)$$

Where:

A_n = the price of an asset at period n, in this case n=30

n = the number of total periods

- Close_6_sma (6-day Simple Moving Average of closing prices): A smoothed average of the last 6 closing prices.

Equation 12. Simple Moving Average.

$$\mathrm{SMA} = \frac{A_1 + A_2 + \ldots + A_n}{n} \quad (1)$$

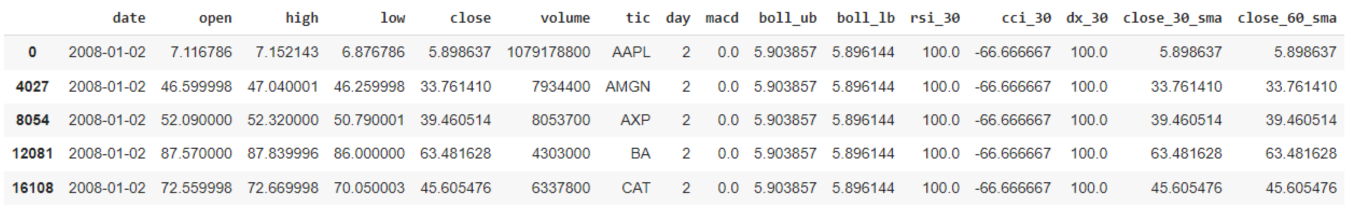
Where:

A_n = the price of an asset at period n, in this case n=6

n = the number of total periods

Here is an example of the exposed variables:

Figure 18. All variables used for the experiment.



	date	open	high	low	close	volume	tic	day	macd	boll_ub	boll_lb	rsi_30	cci_30	dx_30	close_30_sma	close_60_sma
0	2008-01-02	7.116786	7.152143	6.876786	5.898637	1079178800	AAPL	2	0.0	5.903857	5.896144	100.0	-66.666667	100.0	5.898637	5.898637
4027	2008-01-02	46.599998	47.040001	46.259998	33.761410	7934400	AMGN	2	0.0	5.903857	5.896144	100.0	-66.666667	100.0	33.761410	33.761410
8054	2008-01-02	52.090000	52.320000	50.790001	39.460514	8053700	AXP	2	0.0	5.903857	5.896144	100.0	-66.666667	100.0	39.460514	39.460514
12081	2008-01-02	87.570000	87.839996	86.000000	63.481628	4303000	BA	2	0.0	5.903857	5.896144	100.0	-66.666667	100.0	63.481628	63.481628
16108	2008-01-02	72.559998	72.669998	70.050003	45.605476	6337800	CAT	2	0.0	5.903857	5.896144	100.0	-66.666667	100.0	45.605476	45.605476

Source: Own elaboration

Furthermore, in order to better understand the stock behavior and market dynamics, two new variables are introduced:

- <u>Cov_list</u>: it represents a series of covariance matrices, which is a quantitative measure of how the returns of different assets or stocks co-move with each other. A positive covariance suggests that the variables tend to move in the same direction, while a negative covariance indicates that they tend to move in opposite directions. A covariance close to zero suggests little to no linear relationship between the variables (FasterCapital, n.d).

- <u>Return_list</u>: it captures the percentage returns of each stock over the same one-year lookback period. Percentage return is a measure of how much a stock's price has changed relative to its previous value. This information is crucial for assessing the volatility and trends in individual stock prices (FasterCapital, n.d).

6.1.4. Environment Building

We approach the financial challenges of automated stock trading tasks by treating them as Markov Decision Process (MDP) problems due to their stochastic and interactive nature. This modeling framework allows us to formulate the task in a way that considers the dynamic nature of stock price changes. During the training process, the agent observes these changes, takes actions, and calculates rewards, enabling the adjustment of its strategy. The goal is for the agent to iteratively refine its trading strategy by maximizing rewards through interactions with the environment over time.

To create a realistic simulation of stock markets, we implement our trading environments using the OpenAI Gym framework. This framework allows us to simulate livestock markets with authentic market data. This approach ensures that the trading agent operates in an environment that closely mirrors the conditions of real-world stock markets.

Moving forward, we divide the obtained dataset into two distinct sets. The first set, serving as the training data, includes all information recorded between January 1, 2008, and December 31, 2022. This period is crucial for teaching the trading agent and developing its strategy. The second set, designated as the test data, covers the period from January 1, 2023, to December 31, 2023. This data is reserved for evaluating the agent's performance on unseen market conditions, providing insights into the model's generalization capabilities.

The reinforcement signal serves as feedback for the automated trading system, providing insights into the change in portfolio value between the current day and the previous day. This feedback mechanism enables the system to evaluate the effectiveness of its trading decisions and make adjustments to its strategies accordingly. As market conditions evolve throughout the trading day, the system's portfolio performance is influenced by various shifts and fluctuations in the market.

Through continuous learning, the automated trading system analyzes the impact of different trading actions on the overall value of its portfolio. By closely monitoring market trends and assessing the outcomes of its trades, the system gradually develops a deeper understanding of market dynamics. Over time, it learns to identify patterns and correlations among different assets, which improves its ability to make informed trading decisions in future scenarios.

This iterative process of receiving reinforcement signals, adapting to changing market conditions, and learning from past experiences enables the automated trading system to navigate the complexities of financial markets more effectively. As a result, it evolves into a more intelligent and adaptive trader, capable of optimizing portfolio performance and maximizing returns over time.

6.2. Results

Once the experiment has been conducted, the obtained results are going to be analyzed. The Sharpe Ratio obtained for each model will be compared, as well as the quality of their respective times.

The Sharpe ratio evaluates the relationship between an investment's return and its risk. It quantifies the idea that higher returns over a specific period might indicate increased volatility and risk, rather than investment expertise, through the next mathematical formula (Fernando, 2024):

Equation 13. Sharpe Ratio.

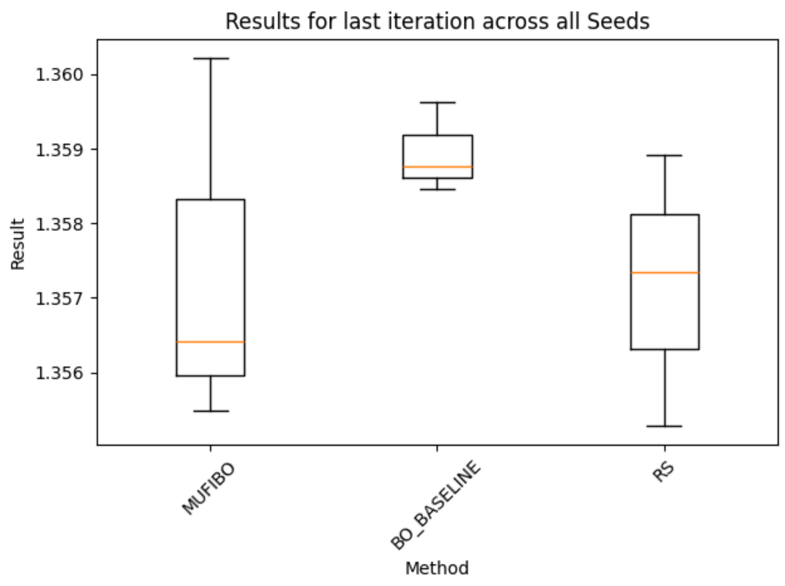
$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

Where:
R_p = portfolio's return
R_f = risk-free rate
σ_p = standard deviation of the portfolio's excess return

The Sharpe ratio is a widely used metric in finance that evaluates a portfolio's risk-adjusted performance by dividing its excess returns by a measure of volatility. Excess returns refer to profits exceeding those of an industry benchmark or the risk-free rate of return. Calculations for the Sharpe ratio can be based on either historical returns or forecasts. Typically, a higher Sharpe ratio is indicative of superior performance when comparing portfolios with similar characteristics. On the other hand, a negative Sharpe ratio suggests that either the risk-free rate or benchmark return exceeds the portfolio's historical or projected return, or that the portfolio's return is anticipated to be negative (Fernando, 2024).

Below is a Boxplot illustrating the results obtained from the experiment. It compares the outcomes achieved by the Multi-fidelity Bayesian Optimization (MUFIBO) method with those of Random Search (RS) and Bayesian Optimization (BO_BASELINE). This visual representation allows for a clear comparison of the performance of each method in the experiment, providing insights into their respective effectiveness in optimizing the given parameters.

Figure 19. Sharpe ratio's result for each method.

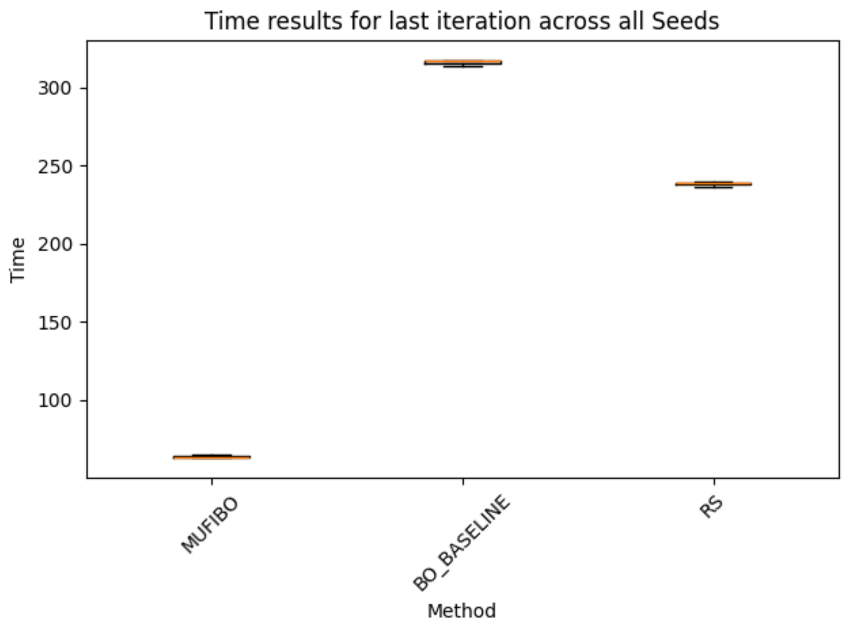


Source: Own elaboration

We observe how, although the Sharpe ratio mean appears to be lower compared to the other methods, a closer examination reveals that the best result is achieved with MUFIBO. This apparently contradictory outcome can be attributed to the correlation in fidelity levels. While fidelity may not consistently yield favorable results, there are instances where it proves to be highly effective. These sporadic successes, although exceptions, demonstrate the potential of fidelity-driven strategies like MUFIBO.

Below is a Boxplot illustrating the time results obtained from the experiment. Despite the occasional shortcomings found when analyzing the Sharpe ratio's results obtaind, when fidelity does lead to success, it does so significantly faster than alternative methods. This highlights the importance of considering both the average performance and the potential outliers when evaluating the effectiveness of different optimization approaches.

Figure 20. Time results for each method.



Source: Own elaboration

In conclusion, the experiment's findings analyze the comparative effectiveness of Multi-fidelity Bayesian Optimization (MUFIBO) against Random Search (RS) and Bayesian Optimization (BO_BASELINE) in portfolio management. While initial observations suggested a lower Sharpe ratio mean for MUFIBO, a closer examination revealed that it achieved the best overall result. This discrepancy highlights the variable effectiveness of fidelity levels, where occasional failures are offset by significant successes. Moreover, the time results demonstrated MUFIBO's efficiency, particularly when fidelity led to favorable outcomes, highlighting its potential for faster optimization. Overall, these insights emphasize the promising role of fidelity-driven strategies like MUFIBO in improving portfolio management strategies, offering valuable implications for future research and practical application in financial markets.

7. CONCLUSIONS AND FURTHER WORK

Each chapter has contributed from the fundamentals to the results and the connection with the objectives presented. In this final chapter, the key ideas, emerging perspectives, and fundamental reflections will be highlighted, thus offering a comprehensive view of the journey we have undertaken during this thesis.

O1: Realization of the document of my thesis.

The finalization of the thesis document represents a significant achievement in this research journey. Through systematic planning, drafting, and revision, the document offers a comprehensive overview of the study's objectives, methodologies, findings, and conclusions. The process of composing the thesis has not only facilitated the organization and synthesis of research but has also refined critical writing and analytical skills, contributing to personal and academic development.

O2: Investigate how DRL can be used in portfolio optimization, specifically focusing on its ability to adapt to non-stationary market conditions and learn from both historical data and real-time feedback.
O3: Critically assess the applicability and effectiveness of a robotrader employing Deep Reinforcement Learning (DRL).

The investigation into the application of Deep Reinforcement Learning (DRL) in portfolio optimization has uncovered significant insights into its adaptability and learning capabilities. Through empirical analysis and experimentation, it was observed that DRL demonstrates promising potential in adapting to non-stationary market conditions and leveraging both historical data and real-time feedback to inform decision-making. These findings emphasize the relevance of DRL as a dynamic and effective tool in portfolio management strategies.

Utilizing Deep Reinforcement Learning (DRL) for portfolio optimization involves a systematic process. Initially, the problem of portfolio optimization is formulated, specifying objectives, constraints, and risk measures. Afterwards, an environment is set up to simulate the financial market, defining state and action spaces, as well as the reward function. A neural network-based agent is then designed to interact with this environment, receiving market data as input and generating portfolio allocations as output. Through training on historical market data, the DRL agent learns to maximize a reward signal, typically based on performance metrics like the Sharpe ratio or cumulative returns. Throughout the training process, the agent explores various portfolio strategies to learn optimal behavior while exploiting successful ones. Following training, the agent's performance is evaluated using validation data or backtesting. Fine-tuning may be conducted based on evaluation results, involving adjustments to hyperparameters, reward functions, or neural network architecture. Once optimized, the trained DRL agent can be deployed in real-world portfolio management scenarios, where its performance is monitored, and adjustments are made as needed to adapt to evolving market conditions. Through this systematic approach, DRL offers a dynamic and adaptive solution to portfolio optimization in today's complex financial landscapes.

O4: Address the gap, which exits due to models such as Markowitz's Modern Portfolio Theory (MPT) which assumes stationary market conditions that may not hold in practice, by using DRL within the field of portfolio optimization.
O5: Revolutionize the field of portfolio management and create substantial benefits for investors, financial institutions, and the broader financial markets.

The utilization of Deep Reinforcement Learning (DRL) within the field of portfolio optimization has effectively addressed the gap presented by traditional models like Markowitz's Modern Portfolio Theory (MPT). By leveraging DRL's adaptability to non-stationary market conditions, the study has demonstrated a more robust and flexible approach to portfolio management.

Deep Reinforcement Learning (DRL) represents a paradigm shift in portfolio management, offering a dynamic and adaptive approach compared to the traditional Markowitz model. Unlike Markowitz's reliance on historical data and static statistical parameters, DRL leverages neural networks to optimize portfolios through iterative learning from market interactions. This learning-based methodology enables DRL to adapt to changing market conditions and respond to stochastic events, potentially enhancing adaptability in unpredictable financial environments. While DRL introduces challenges in interpretability, its stability can be improved through the incorporation of additional data or technical indicators. The transition from the Markowitz method to DRL suggests a more accurate consideration of returns and variances, treating optimization as a sequential learning challenge that evolves with market dynamics.

By demonstrating the efficacy and adaptability of DRL-based strategies, this study lays the foundation for substantial benefits for investors, financial institutions, and the broader financial markets. The insights gained from this research pave the way for improved decision-making processes, improved risk management, and ultimately, greater returns on investment.

O6: Execute the experiment comparing different methods.

The execution of an experiment comparing different methods has provided valuable empirical evidence and insights into the effectiveness of various portfolio optimization approaches. The experiment focuses on optimizing the hyperparameters of the Proximal Policy Optimization (PPO), utilizing different fidelities of the estimated optimal policy. By comparing multi-fidelity Bayesian Optimization with Random Search and traditional Bayesian Optimization, we aim to optimize the hyperparameter tuning process for Deep Reinforcement Learning (DRL) algorithms. Our hypothesis suggests that multi-fidelity should outperform random search and traditional methods, achieving better results in less time.

Through thorough analysis, we observed that while MUFIBO initially presented a lower Sharpe ratio mean, it ultimately achieved the best overall result. This emphasizes the variable efficacy of fidelity levels, where occasional challenges are outweighed by significant successes. Furthermore, MUFIBO demonstrated efficiency in time results, particularly when fidelity led to favorable outcomes, highlighting its potential for faster optimization. These findings emphasize the promising role of fidelity-driven strategies like MUFIBO in improving portfolio management, offering valuable insights for future research and practical implementation in financial markets.

In terms of future work, conducting additional experiments is crucial. To conduct them properly, a significantly larger number of timesteps, iterations, and seeds would be required. This will allow to explore a wider range of scenarios and capture more complex insights into the performance of multi-fidelity Bayesian Optimization. Additionally, it's crucial to expand our comparative analysis by benchmarking our method against alternative approaches. By performing thorough comparisons, deeper insights can be gained into the relative strengths and weaknesses of different optimization strategies in the context of investment management. This comprehensive approach to future work will not only improve the understanding of multi-fidelity Bayesian Optimization but also contribute to the wider knowledge base in financial decision-making.

8. GENERATIVE ARTIFICIAL INTELLIGENCE TOOLS STATEMENT

I, Beatriz Díaz Nameth, a student of ADE and Business Analytics at Universidad Pontificia Comillas, hereby submit my Bachelor's Thesis titled "Multi-fidelity Bayesian Optimization for Deep Reinforcement Learning for portfolio management". I declare that I have used the Generative Artificial Intelligence Tool ChatGPT or similar IAG tools only in the context of the activities described below:

1. Code interpreter: To perform preliminary data analysis.
2. Literary style and language corrector: To improve the linguistic and stylistic quality of the text.
3. Complicated book synthesizer and disseminator: To summarize and understand complex literature.
4. Reviewer: To receive suggestions on how to improve and refine the work with different levels of scrutiny.
5. Translator: To translate texts from one language to another.

I affirm that all the information and content presented in this work are the product of my individual research and effort, except where otherwise indicated and proper credits have been given (I have included appropriate references in the Bachelor's Thesis and have explicitly stated when ChatGPT or similar tools have been used). I am aware of the academic and ethical implications of submitting non-original work and accept the consequences of any violation of this declaration.

Date: 21/04/2024

Signature: Beatriz Díaz Nameth

9. BIBLIOGRAPHY

Baldridge, R. (2023). *Understanding Modern Portfolio Theory*. Forbes. https://www.forbes.com/advisor/investing/modern-portfolio-theory/

Barsce, J. C., Palombarini, J. A., & Martínez, E. C. (2017). *Towards autonomous reinforcement learning: Automatic setting of hyper-parameters using Bayesian optimization.* IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8226439

Bartram et al. (2021). *Machine Learning for Active Portfolio Management*. https://www.semanticscholar.org/paper/36c63fcbf433dd5f2c7b0e522833f8ce8a4fcff6

Bayesian hyperparameter optimization. (2024). Run.ai. https://www.run.ai/guides/hyperparameter-tuning/bayesian-hyperparameter-optimization#optimization

Benhamou, E., Saltiel, D., Ungari, S., & Mukhopadhyay, A. (2020). *Bridging the gap between Markowitz planning and deep reinforcement learning.* https://arxiv.org/abs/2010.09108

Benhamou, E., Ohana, J. J., Guez, B., Saltiel, D., Laraki, R., & Atif, J. (2023). *Comparing Deep RL and Traditional Financial Portfolio Methods.* SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4557792

Bengio, Y., Simard, P, & Frasconi, P. (1994). *Learning long-term dependencies with gradient descent is difficult.* IEEE Xplore. https://ieeexplore.ieee.org/document/279181

Botvinick, M., et al. (2020). *Deep reinforcement learning and its neuroscientific implications.* https://www.cell.com/neuron/pdf/S0896-6273(20)30468-2.pdf

Chen, J. (2024). *Multi-factor model: Definition and formula for comparing factors*. Investopedia. https://www.investopedia.com/terms/m/multifactor-model.asp

Courseteach (2023). *Deep Learning (Part 1): Understanding Basic Neural Networks*. Medium. https://medium.com/@Coursesteach/deep-learning-part-1-86757cf5a0c3

Dayan, P. & Niv, Y. (2008). *Reinforcement learning: The good, the bad and the ugly.* Current Opinion in Neurobiology. https://www.sciencedirect.com/science/article/abs/pii/S0959438808000767

Dulac-Arnold, G., Mankowitz, D., & Hester, T. (2019). *Challenges of real-world reinforcement learning*. https://arxiv.org/abs/1904.12901

Dulac-Arnold, G., et al (2021). *Challenges of real-world reinforcement learning: definitions, benchmarks and analysis.* SpringerLink. https://link.springer.com/article/10.1007/s10994-021-05961-4

Durall, R. (2022). *Asset Allocation: From Markowitz to Deep Reinforcement Learning.* https://arxiv.org/abs/2208.07158

Fare, C. et al. (2022). *A multi-fidelity machine learning approach to high throughput materials screening.* Nature News. https://www.nature.com/articles/s41524-022-00947-9

Gomede, E. (2024). *Bayesian Optimization: Revolutionizing Efficient Search in Complex Spaces*. Medium. https://medium.com/aimonks/bayesian-optimization-revolutionizing-efficient-search-in-complex-spaces-3e2cc476d2cd

Importance of covariance matrix in portfolio construction. FasterCapital. (n.d.). https://fastercapital.com/topics/importance-of-covariance-matrix-in-portfolio-construction.html

Single stock trading. FinRL. (2021). https://finrl.readthedocs.io/en/latest/tutorial/Introduction/SingleStockTrading.html

Fernando, J. (2024). *Sharpe Ratio: Definition, Formula, and Examples*. Investopedia. https://www.investopedia.com/terms/s/sharperatio.asp

Foumani et al. (2022). *Multi-fidelity cost-aware Bayesian optimization*. https://www.semanticscholar.org/paper/bc0228120a1a04bc7ce2a15f698c9b12bf56c4e6

Gao et al. (2020). *Application of Deep Q-Network in Portfolio Management*. https://www.semanticscholar.org/paper/4b53c24062edd98a7d94aabf2b427927b90e64ff

Garrido, E. (2021). Adv*anced Methods for Bayesian Optimization in Complex Scenarios.*

Graesser, L., & Keng, W. L. (2019). *Foundations of deep reinforcement learning. Addison-Wesley Professional.* Google Libros. https://books.google.es/books?hl=es&lr=&id=0HW7DwAAQBAJ&oi=fnd&pg=PT16&dq=main+definitions+of+reinforcement+learning&ots=1LiHCBpVjs&sig=2XSurHQRoXYq9JOeruM0ltLg7Hg#v=onepage&q=main%20definitions%20of%20reinforcement%20learning&f=false

Hambly, B., Xu, R., & Yang, H. (2023). *Recent advances in reinforcement learning in finance. Mathematical Finance.* https://onlinelibrary.wiley.com/doi/epdf/10.1111/mafi.12382

Hammami, N., & Nguyen, K. K. (2022). *On-Policy vs. Off-Policy Deep Reinforcement Learning for Resource Allocation in Open Radio Access Network*. IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9771605

Hou, K., Xue, C., & Zhang, L. (2020). *Replicating anomalies*. The Review of financial studies. OUP Academic. https://academic.oup.com/rfs/article-abstract/33/5/2019/5236964?redirectedFrom=PDF&login=false

Hu, Y. J., & Lin, S. J. (2019). *Deep reinforcement learning for optimizing finance portfolio management.* IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8701368

Huang, G., Zhou, X., & Song, Q. (2020). *Deep reinforcement learning for portfolio management.* https://arxiv.org/abs/2012.13773

Jiang et al. (2016). *Cryptocurrency portfolio management with deep reinforcement learning*. https://www.semanticscholar.org/paper/632225349977400a68825405e18ae3ed927f4f1e

Karunakaran, D. (2018). *Deep learning series 1: Intro to deep learning*. Medium. https://medium.com/intro-to-artificial-intelligence/deep-learning-series-1-intro-to-deep-learning-abb1780ee20

Khamis, A., &Wang, Y. (2021). *Reinforcement Learning*. Reinforcement Learning - AI Search Algorithms for Smart Mobility. https://smartmobilityalgorithms.github.io/book/content/LearnToSearch/ReinforcementLearning.html

Khdoudi, A., Masrour, T., El Hassani, I., & El Mazgualdi, C. (2024). *A deep-reinforcement-learning-based digital twin for manufacturing process optimization*. MDPI. https://www.mdpi.com/2079-8954/12/2/38

Kim, K. G. (2016). *Book review: Deep learning. Healthcare informatics research.* Healthcare Informatics Research. https://synapse.koreamed.org/articles/1075818

Li, S., Xing, W., Kirby, R., & Zhe, S. (2020). *Multi-fidelity Bayesian optimization via deep neural networks*. Advances in Neural Information Processing Systems. https://proceedings.neurips.cc/paper/2020/hash/60e1deb043af37db5ea4ce9ae8d2c9ea-Abstract.html

Li, S., Kirby, R., & Zhe, S. (2021). *Batch Multi-Fidelity Bayesian Optimization with Deep Auto-Regressive Networks.* Advances in Neural Information Processing Systems. https://proceedings.neurips.cc/paper/2021/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html

Li, K., & Li, F. (2024). *Multi-Fidelity Methods for Optimization: A Survey.*

https://arxiv.org/abs/2402.09638#:~:text=Multi%2Dfidelity%20optimization%20(MFO),a%20pre%2Dtrained%20language%20model.

Liu, C. (2022). *5 Concepts You Should Know About Gradient Descent and Cost Function.* KDnuggets. https://www.kdnuggets.com/2020/05/5-concepts-gradient-descent-cost-function.html#:~:text=Gradient%20descent%20is%20an%20iterative,the%20best%20set%20of%20parameters

Liu, Xiao Yang., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., & Dan Wang, C. (2024). *FINRL: A deep reinforcement learning library for automated stock trading in quantitative finance.* ar5iv. https://ar5iv.labs.arxiv.org/html/2011.09607

Lucarelli et al. (2020). *A deep Q-learning portfolio management framework for the cryptocurrency market*. https://www.semanticscholar.org/paper/255e762a3cb9f9a2d5eb31bdcf4499997313cb0b

Mangram, M.E. (2013) A simplified perspective of the Markowitz portfolio theory, SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2147880

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). *Human-level control through deep reinforcement learning. Nature*. https://www.nature.com/articles/nature14236

Mukherjee, S., Deligiannis, P., Biswas, A., & Lal, A. (2020). *Learning-based controlled concurrency testing. Proceedings of the ACM on Programming Languages*. https://dl.acm.org/doi/10.1145/3428298

Mullainathan, S., & Spiess, J. (2017). *Machine learning: an applied econometric approach.* American Economic Association. https://www.aeaweb.org/articles?id=10.1257%2Fjep.31.2.87&ref=ds-econ

Nguyen, V. (2019). *Bayesian optimization for accelerating hyper-parameter tuning. In 2019 IEEE second international conference on artificial intelligence and knowledge engineering (AIKE)*. IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8791696

Odemakinde, E. (2023). *Model-Based and Model-Free Reinforcement Learning: Pytennis Case Study.* Neptune.ai. https://neptune.ai/blog/model-based-and-model-free-reinforcement-learning-pytennis-case-study

Omran, I. et al., (2024). *Deep reinforcement learning implementation on IC Engine Idle Speed Control*. Ain Shams Engineering Journal. https://www.sciencedirect.com/science/article/pii/S2090447924000455?via%3Dihub

Osterrieder, J. (2023). *A Primer on Deep Reinforcement Learning for Finance*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4316650

Owen, L. (2020). *Bird's-Eye View of Reinforcement Learning Algorithms Taxonomy*. Medium. https://towardsdatascience.com/birds-eye-view-of-reinforcement-learning-algorithms-landscape-2aba7840211c#:~:text=Value%2Dbased%20RL%20aims%20to,value%20function%20and%20the%20policy

Pricope, T. V. (2021). *Deep reinforcement learning in quantitative algorithmic trading: A review.* https://arxiv.org/abs/2106.00123

Rao, A., & Jelvis, T. (2022). *Foundations of Reinforcement Learning with Applications in Finance.* https://stanford.edu/~ashlearn/RLForFinanceBook/book.pdf

Sam Obeidat, M. (2018) *The practical limitations of the modern portfolio theory*, LinkedIn. https://www.linkedin.com/pulse/practical-limitations-modern-portfolio-theory-samer-obeidat-mgm/

Shrestha, A., & Mahmood, A. (2019). *Review of deep learning algorithms and architectures*. IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8694781

Siegel, E. (2020). *Introduction to Reinforcement Learning*. Siegel.Work. https://siegel.work/blog/PolicyGradient/

Singh, V., Chen, S.-S., Singhania, M., Nanavati, B., kumar kar, A., & Gupta, A. (2022). *How are reinforcement learning and deep learning algorithms used for big data based decision making in Financial Industries–a review and Research Agenda*. International Journal of Information Management Data Insights. https://www.sciencedirect.com/science/article/pii/S2667096822000374?via%3Dihub

Snoek, J., Larochelle, H. and Adams, R.P. (2012) *Practical Bayesian optimization of Machine Learning Algorithms*, *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html

Sun, S., Wang, R., & An, B. (2021). *Reinforcement learning for quantitative trading.* https://arxiv.org/abs/2109.13851

Tao, R. et al. (2021). *Robo advisors, algorithmic trading and investment management: wonders of fourth industrial revolution in financial markets.* Technological Forecasting and Social Change. https://www.sciencedirect.com/science/article/abs/pii/S0040162520312476

Vandelaer, C. (2022). *Reinforcement Learning: An introduction (Part 3/4).* Medium. https://medium.com/@cedric.vandelaer/reinforcement-learning-an-introduction-part-3-4-e7d883dcbba2

Vijayan PV, V. (2020). *Deep reinforcement learning: Value functions, DQN, actor-critic method, backpropagation through stochastic functions*. Medium. https://medium.com/@vishnuvijayanpv/deep-reinforcement-learning-value-functions-dqn-actor-critic-method-backpropagation-through-83a277d8c38d

Wang et al. (2021). *DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding.*

https://www.semanticscholar.org/paper/0b26cb46675c0b00483c254488c9a14d89eec8e5.

Wu, J., Toscano-Palmerin, S., Frazier, P. I., & Wilson, A. G. (2020). *Practical multi-fidelity bayesian optimization for hyperparameter tuning. In Uncertainty in Artificial Intelligence.* PMLR. https://proceedings.mlr.press/v115/wu20a.html

Wu et al. (2021). *Portfolio management system in equity market neutral using reinforcement learning*. https://www.semanticscholar.org/paper/f24ceb1d7154934883df25411ebd830270cee947

Ye, A. (2020). *A Crash Course in Markov Decision Processes, the Bellman Equation, and Dynamic Programming*. Medium. https://andre-ye.medium.com/a-crash-course-in-markov-decision-processes-the-bellman-equation-and-dynamic-programming-e80182207e85

Zürn, J. (2018). *Reinforcement Learning — An introduction*. Medium. https://jannik-zuern.medium.com/reinforcement-learning-to-survive-in-a-hostile-environment-3658624a5d83

10. APPENDIX

The 30 constituents from the Dow Jones are:

Figure 21. Dow Jones' constituents.

TICKER	NAME
AXP	American Express Co.
AMGN	Amgen Inc.
AAPL	Apple Inc.
BA	Boeing Co.
CAT	Caterpillar Inc.
CSCO	Cisco Systems Inc.
CVX	Chevron Corp.
GS	Goldman Sachs Group Inc.
HD	The Home Depot Inc.
HON	Honeywell International Inc.
IBM	International Business Machines Corp.
INTC	Intel Corp.
JNJ	Johnson & Johnson
KO	Coca-Cola Co.
JPM	JPMorgan Chase & Co.
MCD	McDonald's Corp.
MMM	3M Co.
MRK	Merck & Co. Inc.
MSFT	Microsoft Corp.
NKE	Nike Inc.
PG	Procter & Gamble Co.
TRV	Travelers Companies Inc.
UNH	UnitedHealth Group Inc.
CRM	Salesforce Inc.
VZ	Verizon Communications Inc.
V	Visa Inc.
WBA	Walgreens Boots Alliance Inc.
WMT	Walmart Inc.

DIS	Walt Disney Co.
DOW	Dow Inc.

Source: Own elaboration