Domain Randomization for Deep Reinforcement Learning Agents Trained for Financial Portfolio Management

Max Vilarasau-Serra[1]

Eduardo Garrido-Merchán[2,3]

[1]Department of Quantitative Methods, ICADE, Comillas Pontifical University
https://orcid.org/0009-0001-2790-3598

[2]Department of Quantitative Methods, ICADE, Comillas Pontifical University
[3]Institute of Research in Technology, ICADE, Comillas Pontifical University
https://orcid.org/0000-0002-2695-5484

Acknowledgments

I want to express my heartfelt gratitude to the many people who have played a meaningful role in my academic and personal journey over the past years.

First and foremost, I would like to thank **Álex Escolà Gascón** for introducing me to the world of science and statistics. His passion and curiosity sparked in me the desire to explore, question, and learn, setting the foundation for everything that followed until today.

I am deeply grateful to **José Luis Arroyo Barrigüete**, whose guidance as a teacher and mentor has had a lasting impact, both academically and professionally. His support and advice have been instrumental for me.

A special thanks to **Carlos Halpern Serra** for introducing me to the field of Finance. His enthusiasm for the subject helped me discover a new area of interest and develop a deeper appreciation for it.

To **Alejandro Cadenas González**, thank you for your inspiring vision and for encouraging me to think beyond the obvious, to ask better questions, and to stay ambitious in my goals.

I would also like to thank **Alejandro Rodríguez Gallego**, whose programming classes were foundational for my coding journey. His way of teaching helped me build strong technical skills that I've carried through my work ever since.

To **Alexandra Cifuentes Quintero**, thank you for your ability to make complex ideas simple, and for being such a clear and engaging teacher. Your teaching style made a real difference.

My sincere thanks go to **Manuel Guzmán Caba** for being an outstanding professor in quantitative finance and for offering both academic and professional guidance along the way.

I also want to acknowledge **Carlos Martínez de Ibarreta Zorita**, whose leadership as program director provided clarity, structure, and a sense of direction that helped shape this academic experience.

To **María Coronado Vaca**, thank you for your support during the development of this thesis. Your help was genuinely appreciated.

I owe a special thank you to **Eduardo Garrido Merchán**, not only for supervising this thesis, but also for being an exceptional teacher and a truly insightful guide throughout the entirety of my master's program.

Lastly, on a more personal note, I would like to thank my grandmother, **Rosa Serrat Caballeria**, for always loving me and standing by my side. She is a true example of goodwill, honesty, and kindness. Her innumerable virtues make her a pillar of strength in my life.

Contents

List of Figures

List of Tables

List of Equations

Abstract

This study explores the use of Domain Randomization (DR) to enhance the generalization capabilities of Deep Reinforcement Learning (DRL) agents in financial portfolio management. DRL agents trained solely on historical market data often overfit to specific regimes, limiting their effectiveness in dynamic real-world environments. To address this, we implement DR techniques by perturbing simulated market parameters (such as asset volatility, price dynamics, and noise levels) during training, encouraging agents to develop robust, generalizable strategies. Using the Deep Deterministic Policy Gradient (DDPG) algorithm, we conduct extensive experiments on Dow Jones data and compare the performance of DR-trained agents with conventionally trained ones. The results demonstrate statistically significant improvements in Sharpe ratios across multiple metrics (maximum, quartile, and mean). These findings suggest that DR can serve as an effective regularizer, reducing overfitting and improving the resilience of AI-driven trading strategies in volatile markets.

Keywords: Reinforcement Learning (RL); Deep Reinforcement Learning (DRL); Deep Deterministic Policy Gradient (DDPG); Domain Randomization (DR); Financial Portfolio Management; Quantitative Finance.

1. Introduction

Financial portfolio management is a complex and dynamic problem that requires sophisticated strategies to maximize returns while mitigating risks. Traditional approaches to portfolio management have relied heavily on statistical models, econometric techniques, and heuristic methods (Fama, 1970; Markowitz, 1952). However, the advent of machine learning, particularly DRL, has introduced new possibilities for automating and optimizing financial decision-making (Jiang et al., 2017; Yang et al., 2020). DRL-based agents have demonstrated promising results in sequential decision-making tasks, making them a viable alternative for portfolio allocation strategies (Ye et al., 2020; Z. Zhang et al., 2020).

Despite their potential, DRL agents trained for financial portfolio management face several significant challenges. One of the most critical issues is generalization—the ability of an agent trained in a specific environment to perform well in real-world financial markets, which are inherently non-stationary, stochastic, and characterized by high levels of uncertainty (Charpentier et al., 2021; Cobbe et al., 2019). Training on historical market data alone often leads to overfitting, where an agent learns strategies that perform well on past data but fail when exposed to new or unseen market conditions (Sutton & Barto, 2018).

To address this challenge, DR has emerged as a powerful technique to improve the robustness and adaptability of DRL agents. DR involves systematically perturbing the training environment by introducing variations in key parameters, such as asset price dynamics, volatility structures, transaction costs, and market microstructure (Peng et al., 2018; Pinto et al., 2017). By exposing the DRL agent to a wide range of market conditions during training, DR encourages the development of strategies that generalize better to real-world scenarios (Sadeghi & Levine, 2017; Tobin et al., 2017).

This technique has been successfully applied in robotics and computer vision to bridge the gap between simulation and reality (Tobin et al., 2017), but its potential for financial applications remains underexplored (Benhamou et al., 2021; Spooner & Savani, 2020). The objective of this thesis is to investigate the effectiveness of DR in enhancing the generalization capabilities of DRL agents trained for financial portfolio management. Specifically, this research aims to:

a) Develop a DRL framework for portfolio management that integrates DR techniques (Yu et al., 2019).

b) Evaluate the impact of different randomization strategies on the performance and robustness of DRL agents (Wang et al., 2021).

c) compare DR agents with conventionally trained agents using various performance metrics, including cumulative returns, Sharpe ratio (Sharpe, 1966), and maximum drawdown (H. Park et al., 2020).

d) analyze the adaptability of DR agents to out-of-sample market conditions and extreme events (Benhamou et al., 2021).

By addressing these research objectives, this study seeks to contribute to the growing body of literature on DRL for financial applications and provide insights into improving the practical deployment of AI-driven trading strategies. The findings could offer significant implications for both academic research and industry practitioners, highlighting the role of DR in enhancing the robustness of AI-driven portfolio management (Z. Zhang et al., 2020).

The primary motivation behind this study is to address a critical challenge in the application of DRL to financial portfolio management: the susceptibility of DRL agents to overfitting spurious patterns in historical market data (Sutton & Barto, 2018). Financial markets are inherently noisy and non-

stationary, and DRL agents, when trained on limited or unrepresentative samples, may learn patterns that do not generalize well to unseen data. This overfitting can lead to poor out-of-sample performance and ultimately undermine the reliability of automated trading strategies. We hypothesize that by introducing DR techniques during training (e.g. injecting stochasticity into the environment or perturbing the data) we can increase the agent's robustness to market variability. In particular, DR has the potential to reduce the variance of predictions by forcing the agent to focus on stable, generalizable signals rather than noise. Through this lens, our study investigates the efficacy of DR as a regularization strategy to enhance the generalization capabilities of DRL agents in dynamic financial environments.

The structure of the paper is as follows. State-of-the-Art (Section 2) presents a review of the state-of-the-art literature on generalization challenges in DRL, with an emphasis on DR. Scope of the Study *&* Reach (Section 3) defines the scope, objectives, and limitations of our study. Theoretical Framework *&* Methodology (Section 4) outlines the theoretical background and methodological framework, including the DRL agent architecture, the use of the DDPG algorithm, and the implementation of DR. Experimental Design *&* Results (Section 5) details the experimental design and reports the results, both quantitative and qualitative. Finally, Conclusions *&* Future Work (Section 6) provides conclusions and proposes directions for future work.

2. State-of-the-Art

Understanding the current landscape of research is essential for identifying both the limitations and the opportunities within DRL applied to financial portfolio management. This section reviews the most relevant developments in the field, with a particular focus on how DR techniques have been leveraged to improve generalization in dynamic and uncertain environments, such as the one discussed.

2.1. Domain Randomization as a means of Generalization

Generalization to unseen conditions is a core challenge in DRL, especially in non-stationary environments where the underlying data distributions shift over time. Standard DRL agents often exhibit excellent performance on the training environment yet struggle when faced with novel situations, a phenomenon observed even in controlled benchmarks (Benhamou et al., 2021). In the context of financial markets, this issue is acute: an agent trained on historical market data can easily overfit to past patterns and fail in new market regimes (Charpentier et al., 2021; Cobbe et al., 2019). Traditional machine learning techniques offer limited help here, as they typically assume independent and identically distributed data (Sutton & Barto, 2018). Thus, developing DRL agents that can generalize beyond their training experience is of paramount importance for reliable real-world deployment.

DR has emerged as a powerful technique to improve generalization by training agents on a diverse ensemble of environments instead of a single fixed environment. The key idea is to systematically perturb the training environment by randomizing key parameters within plausible ranges so that the agent learns to handle a wide variety of situations (Sadeghi & Levine, 2017; Tobin et al., 2017). Introduced in the context of sim-to-real transfer for robotics, DR was initially used to bridge the "reality gap" between simulated training and the real world (Tobin et al., 2017). For example, Tobin et al. (2017) trained vision-based object detectors and manipulation policies entirely in simulation by randomizing rendering properties (textures, lighting, object positions, etc.), enabling the learned models to perform robustly when exposed to real images. Similarly, Sadeghi & Levine (2017) showed that a drone navigation policy trained with random variations in a simulated indoor environment (e.g., floor and wall textures, lighting conditions, etc.) could generalize to real-world flight using only a single camera input, despite never having seen real images during training.

These early successes demonstrated that exposing a DRL agent to sufficient variability in training can make it invariant to irrelevant specifics of any single training instance, thereby improving its ability to generalize. In essence, DR can be viewed as a form of implicit data augmentation for RL: instead of augmenting input data as done in supervised learning, DR augments the environment's dynamics or observations. This approach has since been widely adopted in robotics and beyond. For instance, randomizing physical parameters such as masses, friction coefficients, and joint tolerances in robotic simulators forces control policies to become robust against modeling errors (Peng et al., 2018). Policies trained with such dynamics randomization have been successfully transferred to real robots with minimal fine-tuning, underscoring the efficacy of DR in handling discrepancies between training simulations and complex reality (Akkaya et al., 2019; Peng et al., 2018). In computer vision, analogous ideas have also been explored: generating synthetic training images with randomized backgrounds, object appearances, and lighting has improved the robustness of vision models to real-world image variations (Tremblay et al., 2018). Across these domains, the common thread is that diversity in training leads to more generalizable learned representations. By preventing the agent from latching onto spurious details of any single environment, DR encourages the learning of fundamental strategies that apply across varied

conditions (Tobin et al., 2017). However, DR is only one approach to improving generalization. The following section discusses several alternative techniques that researchers have explored to train more robust DRL agents.

2.2. Alternative approaches to Domain Randomization

DR is one approach among several that have been proposed to enhance generalization in DRL. In this section we briefly review other notable strategies like Ensemble Methods, Meta-Learning, Adversarial Training, Regularization and Data Augmentation, and contrast them with DR:

2.2.1. *Ensemble Methods*

Ensemble learning improves robustness by training a collection of models and combining their decisions. In DRL, this can mean learning an ensemble of policies or value functions that vote or average their outputs (Lee et al., 2021; Osband et al., 2016). The diversity among ensemble members helps the overall agent avoid overfitting to peculiarities of any single training run. For example, the SUNRISE framework integrates an ensemble of Q-learning agents to achieve better generalization and higher performance on continuous control tasks (Lee et al., 2021). Ensembles can also reduce estimation biases and stabilize learning, which indirectly benefits generalization (Anschel et al., 2017).

Compared to DR (which varies environments for a single agent), ensembles increase robustness by averaging over multiple agents trained on the same environment (or even on different randomized environments). In practice, ensemble methods can be combined with DR by training each ensemble member on different environment variations, thus combining their benefits.

2.2.2. *Meta-Learning*

Meta-learning (or "learning to learn") trains agents that can rapidly adapt to new tasks or changes in the environment by leveraging knowledge gained across many training tasks (Finn et al., 2017). In the context of RL, meta-learning algorithms like Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) or context-based approaches like PEARL (Rakelly et al., 2019) expose the agent to a distribution of tasks/environments during training so that it acquires an internal mechanism to adapt to new, unseen environments with minimal additional experience. For example, an agent might train on multiple simulated market scenarios with different dynamics; a meta-RL algorithm would encourage the agent to infer the current scenario (perhaps through a latent context variable) and adjust its policy accordingly (Benhamou et al., 2021; Rakelly et al., 2019).

Unlike DR, which learns a single policy that aims to be robust across all variations, meta-learning explicitly trains the agent to adapt to variations. This can be advantageous in non-stationary settings: rather than deploying one static policy to handle everything, the agent learns how to quickly tune its policy to the prevailing conditions. Benhamou et al. (2021), for instance, utilize a context-based DRL approach for portfolio management, where the agent's policy includes an auxiliary network to detect market regime changes (such as the onset of a crisis) and adjust its actions accordingly. Meta-learning approaches can complement DR: an agent might be trained on randomized domains and also trained to adapt its behavior based on clues (or context) from the environment about which domain is currently in effect.

2.2.3. *Adversarial Training*

Adversarial approaches improve generalization by preparing the agent for worst-case perturbations. In robust adversarial RL, an auxiliary "adversary" is introduced during training to challenge the agent (Pinto et al., 2017). The adversary may directly perturb the agent's observations or actions, or manipulate the environment within some bounds, with the goal of inducing agent failures. The RL agent, in turn, learns to handle these adversarial interventions, resulting in a policy that is resilient to a range of difficult or unforeseen conditions. Pinto et al. (2017) demonstrated this concept by training a robot control policy in the presence of an adversarial force that pushed the robot; the resulting policy was significantly more robust to disturbances than one trained under normal conditions.

In the financial domain, Spooner & Savani (2020) applied adversarial training by introducing an adversary that perturbs market dynamics in a market-making simulation. The adversarial agent in their setup would, for example, generate challenging price movement patterns or adversarial order flow, forcing the trading agent to learn strategies that remain profitable under stress. This approach is akin to DR in that it broadens the set of conditions the agent trains on; however, rather than random sampling of conditions, it focuses on targeted worst-case scenarios. Adversarial training often yields very robust policies, though potentially at the cost of some performance in nominal conditions, whereas plain DR aims for a broad but not necessarily worst-case distribution of environments.

2.2.4. *Regularization & Data Augmentation*

Another line of defense against overfitting in DRL borrows techniques from supervised learning: adding appropriate regularizers or noise during training to encourage the agent to learn more general features. For example, weight decay (L2 regularization) and dropout have been applied to the neural networks of DRL agents to prevent overly complex co-adaptations of neurons (Farebrother et al., 2018). Farebrother et al. (2018) found that such regularization can modestly improve an RL agent's ability to generalize to new instances of Atari games by reducing overfitting to the training levels.

Data augmentation on the agent's observations is another effective strategy, especially in vision-based RL. Techniques like random image crops, flips, or adding Gaussian noise to observations have shown improvements in generalization for video-game agents by training the policy on perturbed versions of the original inputs (Laskin et al., 2020; Raileanu et al., 2021). In effect, these methods increase the diversity of experience without altering the environment's underlying dynamics.

Regularization and augmentation are typically easier to implement than DR since they do not require designing a family of environment variations. However, they address only certain facets of generalization. For instance, adding observation noise might make a trading agent's policy robust to noisy price signals, but it will not prepare it for a structural break in market dynamics. In contrast, DR (or adversarial training) could simulate such structural breaks during training. In practice, a combination of these techniques may be used: one could randomize market parameters (a form of environment augmentation) while also applying, say, dropout in the policy network to tackle generalization from multiple angles.

2.2.5. *Conclusion*

Each of these generalization strategies has its merits, and they are not mutually exclusive. Ensemble and Meta-Learning methods tend to require substantially more training (training multiple models or an adaptive model), while Adversarial Training and DR demand careful design of perturbations to be

effective yet realistic. Regularization-based methods are lightweight but might yield smaller gains.

DR stands out for explicitly tackling environment-driven uncertainty. By training on a broad distribution of environments rather than a single one, it directly addresses the problem of environment shift. Nevertheless, it is often beneficial to combine approaches (e.g., using dropout or ensembles in conjunction with DR) to cover different sources of overfitting, potentially increasing performance.

2.3. Domain Randomization Applied to Financial Portfolio Management

Applying DR to financial portfolio management is a novel and promising idea, but it comes with unique challenges. Financial markets can be viewed as complex, partially observable environments with evolving dynamics, which makes them an ideal setting to benefit from DRL generalization techniques. To date, however, the use of DR in finance remains relatively limited (Benhamou et al., 2021; Spooner & Savani, 2020). Most DRL-based trading systems are trained and tested on historical market data from a fixed period or a particular market, which risks producing agents that perform well under those specific conditions but fail to adapt when market characteristics change. There is a growing recognition that robustness to regime changes and unexpected events is crucial for AI-driven portfolio management (Charpentier et al., 2021). This has led researchers to explore methods for training agents that can handle different market scenarios. For example, Z. Zhang et al. (2020) highlight the importance of training trading agents on diverse market conditions (e.g., bull and bear markets) to enhance robustness. Likewise, Jiang et al. (2017), Yang et al. (2020), and Ye et al. (2020) have demonstrated DRL approaches for portfolio allocation, but they primarily evaluate these methods in static historical backtests without explicit mechanisms to ensure generalization across varying future market regimes.

DR offers a systematic way to inject variety into the training phase of a trading agent. In a financial context, this could involve generating multiple market scenarios by perturbing key market parameters and processes in a simulated environment. For instance, key elements for randomization include volatility, jump frequency, asset correlation, and slippage patterns (Pinto et al., 2017). By training on an ensemble of such simulated markets, the agent would learn investment strategies that are not tailored to one specific history but are resilient across many plausible futures. This approach is conceptually similar to the stress testing or scenario analysis used in quantitative finance, where models are evaluated on hypothetical extreme conditions; however, with DRL one can incorporate these scenarios directly into the training loop.

Recent work by Wang et al. (2021) has begun to investigate different randomization strategies in market environments, showing that the way randomness is introduced (e.g., perturbing historical data with noise vs. drawing synthetic samples from a generative model) can significantly affect the learned policy's performance and stability. Although comprehensive studies are still lacking, these early explorations suggest that DR could help trading agents learn regime-agnostic policies that maintain performance even as market dynamics evolve. Despite its promise, adapting DR to finance faces several limitations and open research gaps. First, unlike in robotics or games, we do not have a perfect simulator of "market physics"—financial simulators are at best approximate. Randomizing a flawed simulator could produce training data that diverges too much from reality, potentially confusing the agent.

Designing realistic yet diverse market randomizations is an art in itself: one must ensure the random scenarios are plausible representations of how real markets might behave. If the randomization is too mild, the agent may still overfit to narrow conditions; if it is too wild, the agent might learn strategies that work in simulation but not in actual markets. Identifying which parameters to randomize (e.g., the sequence of market news events, the degree of mean reversion, the frequency of market crashes) is a

non-trivial task requiring financial expertise. Moreover, financial time series have memory and heavy-tail risks that are harder to capture with simple parametric randomization. Current research is beginning to explore the use of generative models (e.g., market scenario generators using GANs) to produce more realistic random market environments, but integrating these into DRL training is still nascent.

Another challenge is evaluation: in robotics, a policy can be tested on a real robot to judge sim-to-real success. In finance, deploying an unproven policy on live markets is risky and costly. Thus, we rely on historical out-of-sample testing, but truly unseen market conditions may not exist in past data. This makes it difficult to objectively quantify how well DR improves generalization for trading agents. Some researchers address this by using split-by-time evaluations—training on one period and testing on a later period that includes different market regimes (Benhamou et al., 2021). If an agent trained with DR can adapt to the later period's regime shifts better than a conventionally trained agent, that provides evidence of improved generalization.

So far, approaches to achieve generalization in financial DRL have more often relied on alternative strategies like robust training and meta-learning rather than on explicit DR. Benhamou et al. (2021), as mentioned earlier, incorporate context detection to help the agent handle crises, effectively adding an adaptive component to the policy. Spooner & Savani (2020) introduce an adversarial agent to stress-test a trading strategy during training, yielding a more robust market-making agent. These studies confirm the value of training-time interventions to improve out-of-sample performance. They also underscore a gap: DR remains relatively underexplored in the context of financial applications. No widely cited study yet provides a systematic assessment of DR for portfolio management, which is precisely the gap this work aims to fill.

In summary, DR holds significant potential for advancing the state of the art in DRL for portfolio management. By learning from a breadth of simulated market conditions, a DRL agent could, in theory, internalize a strategy that works in calm markets, volatile crashes, and everything in between. Achieving this in practice will require careful design of randomization schemes—possibly hybridized with other generalization techniques (ensembles, meta-learning, etc.)—to ensure realism and effectiveness. The current literature provides encouraging hints but also makes it clear that more research is needed. Key open questions include: How can we generate realistic yet diverse market scenarios for training? What aspects of market behavior should be randomized versus kept historical? How should improvements in generalization be measured quantitatively? Addressing these questions will be crucial for bringing the benefits of DR to financial DRL. The work reviewed above provides a strong foundation and motivation to pursue these issues, suggesting that a well-crafted DR approach could substantially enhance the robustness and adaptability of DRL-based portfolio managers.

3. Scope of the Study & Reach

This section defines the scope of the study conducted on the use of DR in DRL agents for financial portfolio management and its reach. It presents the specific experimental objectives, the working hypotheses, and the limitations of the experimental setup that contextualize the obtained results.

3.1. Experimental Objectives

This work investigates whether DR can help DRL agents generalize beyond historical financial data. A key challenge in financial DRL is overfitting to past market conditions, as agents trained on static histories often struggle with regime shifts (Charpentier et al., 2021; Cobbe et al., 2019).

We aim to quantitatively assess whether a DRL agent trained with DR yields higher Sharpe ratios (a standard measure of risk-adjusted returns) than an agent trained on fixed historical data. The objectives that we will follow are:

a) To determine whether DR leads to higher Sharpe ratios (Sharpe, 1966) compared to historical-only training.

b) To evaluate the consistency of DR improvements by examining mean, upper-quartile, and maximum performances.

c) To statistically validate whether observed differences are significant using hypothesis testing or if these are due to chance.

3.2. Hypotheses

We define three hypotheses designed to capture different dimensions of improvement, thereby allowing for a nuanced evaluation of whether DR provides statistically significant advantages over conventional historical training, and if so, to what degree:

- ***H1*** $\Rightarrow$ ***Best-case improvement:*** The best Sharpe ratio from the 25 DR agent runs is greater than that of the historically trained agent:

$$\begin{aligned} H_0 &: \ \max[\text{Sharpe}_{\text{DR}}] \leq \mu[\text{Sharpe}_{\text{hist}}] \\ H_1 &: \ \max[\text{Sharpe}_{\text{DR}}] > \mu[\text{Sharpe}_{\text{hist}}] \end{aligned} \tag{1}$$

- ***H2*** $\Rightarrow$ ***Consistency (Q3):*** The 75th percentile (Q3) of the DR agent runs is greater than that of the historically trained agent:

$$\begin{aligned} H_0 &: \ Q3[\text{Sharpe}_{\text{DR}}] \leq \mu[\text{Sharpe}_{\text{hist}}] \\ H_1 &: \ Q3[\text{Sharpe}_{\text{DR}}] > \mu[\text{Sharpe}_{\text{hist}}] \end{aligned} \tag{2}$$

- ***H3*** $\Rightarrow$ ***Mean superiority:*** The mean Sharpe ratio of the DR agent runs is greater than the mean of the historical runs:

$$\begin{aligned} H_0 &: \ \mu[\text{Sharpe}_{\text{DR}}] \leq \mu[\text{Sharpe}_{\text{hist}}] \\ H_1 &: \ \mu[\text{Sharpe}_{\text{DR}}] > \mu[\text{Sharpe}_{\text{hist}}] \end{aligned} \tag{3}$$

3.3. Assumptions of the Experimental Setup

This section outlines the assumptions underlying the experimental setup. These assumptions are not expected to significantly affect the replicability of results when modified.

a) **Data assumptions**: All training and test data are derived from the same underlying financial dataset, which may limit generalizability to other assets or time periods (Wang et al., 2021).

b) **Algorithmic assumptions**: Even though different algorithms could be applied, the core findings are not expected to change significantly.

c) **Model assumptions**: Even though different models could be applied, the core findings are not expected to change significantly.

3.4. Limitations of the Experimental Setup

This section outlines the limitations of the experimental setup. These factors may affect the interpretability and generalizability of the findings; and should be carefully considered when evaluating the robustness, replicability, and applicability of the results.

a) **Computational constraints**: Only 25 runs per condition were executed due to limited computational resources. Larger sample sizes would improve the power of the statistical tests.

b) **Simplified DR**: The DR procedure uses a limited set of features (e.g., price dynamics, volatility, etc.) and may not capture the full complexity of financial markets (Benhamou et al., 2021).

c) **Evaluation metric**: Only Sharpe ratio is considered. Other metrics such as maximum drawdown or Sortino ratio could complement the analysis (H. Park et al., 2020).

d) **Model tuning**: The agent architecture and hyperparameters were not fully optimized. More sophisticated tuning (e.g., Bayesian optimization) could yield better results (Garrido-Merchán, 2025).

Despite these limitations, the experimental setup offers valuable preliminary evidence on the potential of DR to enhance the generalization capabilities of DRL agents in financial portfolio management, thereby contributing to more robust and effective investment strategies.

4. Theoretical Framework & Methodology

In this section, we provide the theoretical foundations and methodological tools underpinning our study. The aim is to contextualize the strategies explored in the following chapters within established financial theory and to explain the analytical techniques used for model development and evaluation.

4.1. Technical Analysis & Portfolio Management

Financial trading encompasses two major problem domains: the identification of profitable trading signals and the optimal allocation of assets in a portfolio. The former is the focus of technical analysis, while the latter falls under portfolio management. In this subsection, we outline key principles of each domain, highlighting their theoretical underpinnings and the challenges they present in the context of developing quantitative trading strategies.

4.1.1. *Technical Analysis*

Technical analysis refers to forecasting future price movements based on patterns and trends in historical market data. It relies on the premise that past trading information (price and volume) contains signals about future dynamics, in contrast to fundamental analysis which examines economic and financial indicators of an asset. Technical analysis has been part of trading practice for decades (Lo et al., 2000), and it remains widely used among market practitioners. For instance, Menkhoff (2010) reports that 87% of surveyed fund managers incorporate technical analysis into their decision-making process, particularly for short-term investment horizons. The appeal of technical analysis lies in its attempt to identify recurrent market patterns (such as trends or momentum) that can inform buy or sell decisions.

At the core of technical analysis is the time series of five key data points recorded for each period: the open, high, low, close, and volume (OHLCV). For each time interval t, the *open* price O_t is the first traded price, and the *close* price C_t is the last traded price in that period. The *high* H_t and *low* L_t denote the extreme upper and lower prices achieved during the interval. The *volume* V_t represents the total number of units traded in period t, capturing the level of market activity and liquidity. Together, these OHLCV elements provide a comprehensive snapshot of market behavior for each period, serving as the raw input for both visual charting and quantitative technical indicators.

Figure 1 illustrates the relationship between a time-series line chart *(left)* and its corresponding OHLC bar representation *(right)*. On the *left*, we observe a stylized price trajectory highlighting the Open, High, Low, and Close points across a hypothetical time window. On the *right*, price level points across a hypothetical time window are encoded in a compact OHLC bar format. These are commonly used in technical analysis to visualize price movement per period.

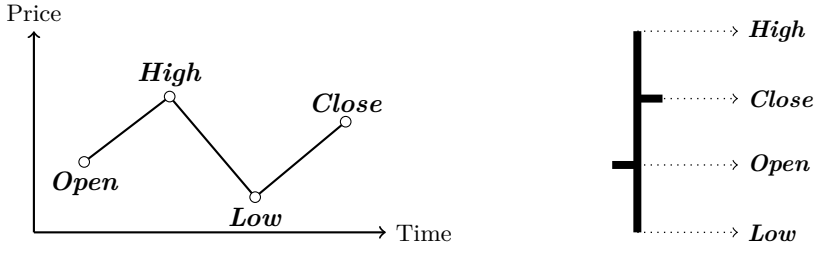

Figure 1: Line chart *(left)* and its corresponding OHLC bar representation *(right)*.

From OHLC prices, analysts frequently derive synthetic variables such as the Range, which captures intraperiod price dispersion by subtracting the *Low* to the $High$:

$$\text{Range}_t = H_t - L_t \tag{4}$$

Another widely used metric is the Typical Price (TP for short), which sums up *High*, *Low*, and *Close*; and then divides that number by 3:

$$\text{TP}_t = \frac{H_t + L_t + C_t}{3} \tag{5}$$

In other cases, a weighted version using $Volume$, such as the Volume Weighted Average Price (VWAP for short), where P_i and V_i represent transaction $Price$ and $Volume$, respectively, across N_t trades in period t:

$$\text{VWAP}_t = \frac{\sum_{i=1}^{N_t} P_i V_i}{\sum_{i=1}^{N_t} V_i} \tag{6}$$

The Volume Weighted Average Price metric is widely used as a benchmark for trade execution and intraday price analysis (Berkowitz et al., 1988).

In summary, OHLCV data forms the foundational input for nearly all of technical analysis (Murphy, 1999). It supports both visual inspection, through the identification of chart patterns such as candlestick formations, support and resistance levels, and trend lines, as well as the systematic computation of a wide range of technical indicators. These include momentum indicators (e.g., RSI, MACD), trend-following tools (e.g., moving averages, ADX), and volatility measures (e.g., Bollinger Bands, ATR). By capturing essential market microstructure—price extremes, directional movement, and trading volume—OHLCV data enables analysts and algorithmic systems alike to extract insights about market sentiment, trend strength, and potential reversals.

In practice, traders employ a variety of technical indicators and chart patterns to generate trading signals. These include trend-following indicators (e.g., moving averages), oscillators (e.g., the Relative Strength Index), and specific price patterns (e.g., head-and-shoulders formations). For example, a Simple Moving Average (SMA for short) of price P over N days is defined as follows:

$$\text{SMA}_N(t) = \frac{1}{N} \sum_{i=0}^{N-1} P_{t-i} \tag{7}$$

A widely used trading strategy involves comparing a short-term and a long-term SMA to detect shifts in market momentum. Specifically, a "golden cross" occurs when the short-term SMA (e.g., 50 days) crosses above the long-term SMA (e.g., 200 days), indicating a transition from a bearish (i.e. the prospect of prices to fall) to a bullish (i.e. the prospect of prices to rise) trend. This crossover is interpreted as a strong signal of upward momentum, suggesting that recent price gains may continue, and is therefore commonly used by traders as a cue to initiate long positions. Conversely, a "death cross"—when the short-term SMA falls below the long-term SMA—is typically viewed as a bearish signal, prompting caution or the consideration of short positions.

The effectiveness of technical analysis has been a subject of considerable debate in finance. The weak-form Efficient Market Hypothesis (EMH for short) asserts that asset prices already reflect all information contained in past prices, implying that no trading rule based solely on historical data can consistently yield excess profits (Fama, 1970). Nevertheless, numerous studies have found evidence of predictability in asset returns that technical strategies can exploit. Brock et al. (1992), in an influential study, showed that simple technical rules (such as moving-average and trading-range break-out strategies) applied to U.S. stock market data produced returns significantly better than chance. More recent work by Lo et al. (2000)

applied rigorous statistical pattern-recognition algorithms to identify classic chart formations (e.g., head-and-shoulders or double bottoms) and found that several of these patterns had modest but statistically significant predictive power. Additionally, C.-H. Park & Irwin (2007) provide a comprehensive review of 95 empirical studies of technical trading strategies, of which roughly 56% report positive net profitability after accounting for transaction costs and other factors. Such findings suggest that markets are not perfectly efficient in the weak form, and that carefully designed technical trading rules can sometimes achieve above-average risk-adjusted returns.

However, technical analysis faces the following important challenges: patterns that worked in the past may weaken or disappear as market regimes change or as more traders recognize and trade on them, thus diminishing their profitability (C.-H. Park & Irwin, 2007); and there is also a risk of overfitting-discovering spurious patterns in historical data that do not generalize to future periods (Bailey et al., 2014). Consequently, successful technical trading requires continual adaptation and robust validation to ensure that identified signals have genuine predictive value and are not merely artifacts of noise.

In summary, technical analysis provides a framework for decision-making based on market price behavior. It frames trading as a sequential decision problem: at each time step, the trader interprets the latest price information (through the lens of technical indicators) and must decide whether to buy, sell, or hold. The inherently dynamic and data-driven nature of this process makes it amenable to quantitative modeling approaches.

4.1.2. *Portfolio Management*

Portfolio management is concerned with the optimal allocation of wealth across multiple assets to balance return and risk. Unlike technical analysis, which often focuses on timing trades for a single asset, portfolio management considers a broader investment context: "how to construct and adjust a collection of assets (i.e. a portfolio) to meet an investor's objectives".

The cornerstone of modern portfolio management is Markowitz's Modern Portfolio Theory (MPT for short). In his seminal paper, Markowitz (1952) formalized the mean-variance optimization framework, which quantifies the trade-off between expected return and risk (which is measured as the variance of portfolio returns). Investors aim to maximize returns for a given level of risk or, conversely, to minimize risk for a target level of return. This can be expressed as the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbf{w}^\top \Sigma \mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{1} = 1 \\ & \mathbf{w}^\top \boldsymbol{\mu} = R_T \end{aligned} \tag{8}$$

where $w = (w_1, \ldots, w_n)^\top$ is the vector of portfolio weights for n assets, Σ is the $n \times n$ covariance matrix of asset returns, μ is the vector of expected returns, and R_T is a target portfolio return. The first constraint ensures the weights sum to one (full investment of capital), and the second constraint fixes the portfolio's expected return to R_T. Solving this optimisation for different values of R_T yields a set of optimal portfolios that delineate the efficient frontier. The efficient frontier represents the set of portfolios offering the highest expected return for each level of risk; portfolios lying below this frontier are suboptimal because they achieve lower returns for the same risk.

Building on MPT, the Capital Asset Pricing Model (CAPM for short) introduced by Sharpe (1964) further characterizes the relationship between risk and expected return in equilibrium. CAPM posits that if investors optimize portfolios according to mean-variance principles, the market portfolio (representing

the aggregate of all assets) will be on the efficient frontier, and asset prices will adjust such that only systematic risk influences expected returns. In the CAPM formulation, the expected return of asset i is linear in its beta (β_i), which measures the asset's co-movement with the market:

$$E[R_i] = R_f + \beta_i\big(E[R_m] - R_f\big) \tag{9}$$

with $\beta_i = \dfrac{\mathrm{Cov}(R_i,\, R_m)}{\mathrm{Var}(R_m)}$, where R_f is the risk-free interest rate, $E[R_m]$ is the expected return of the market portfolio, and $E[R_m] - R_f$ is the market's expected excess return (or risk premium). According to the CAPM, an asset with $\beta_i = 1$ should have an expected return equal to the market average, while an asset with $\beta_i > 1$ should earn above-average returns at the cost of a higher systematic risk, and an asset with $\beta_i < 1$ should earn lower returns but with a lower systematic risk. Idiosyncratic risk (i.e. risk unique to a single asset) is not rewarded because it can be eliminated through diversification. Although empirical tests have exposed limitations of the CAPM and led to the development of multi-factor extensions (e.g., Fama & French, 1993), it remains a fundamental framework for understanding risk-return trade-offs.

Another important aspect of portfolio management is performance evaluation and risk control. A widely used performance metric is the Sharpe ratio (Sharpe, 1966), which measures risk-adjusted return. The Sharpe ratio S for a portfolio p is defined as follows:

$$S = \frac{E[R_p - R_f]}{\sigma_p} \tag{10}$$

where R_p is the portfolio return, R_f is the risk-free rate, $E[R_p - R_f]$ is the expected excess return, and σ_p is the standard deviation of the portfolio's return. A higher Sharpe ratio indicates a more attractive portfolio on a risk-adjusted basis, as it delivers greater excess return per unit of volatility. Portfolio managers often seek to maximize the Sharpe ratio when selecting investments or adjusting portfolio weights, subject to constraints such as liquidity and regulatory requirements.

In practice, portfolio management is a dynamic process rather than a one-time calculation. Market conditions and asset characteristics (expected returns, volatilities, correlations, etc.) change over time, so a portfolio that is optimal today may not remain optimal in the future. Investors typically rebalance their portfolios periodically or in response to significant market moves, engaging in sequential decision-making under uncertainty. The portfolio manager must continuously decide how to adjust holdings to respond to new information, manage risk exposures, and exploit emerging opportunities. Thus, effective portfolio management demands not only an optimal initial allocation but also a robust strategy for ongoing adjustments as conditions evolve.

4.1.3. *Conclusion*

Overall, technical analysis and portfolio management together encompass the key decision-making challenges in quantitative finance. Both involve sequential choices under uncertainty—whether it is timing individual trades or continuously reallocating assets—that require balancing potential returns against risks. These characteristics motivate the need for sophisticated, data-driven methods capable of learning and adaptation, which we explore in the subsequent part of this methodology.

4.2. Methodology

In this section we outline the methodology employed in this work, which is divided in: *a)* the introduction of the general framework of DRL, which serves as the foundation for our trading agent; *b)* the use of the DDPG algorithm, a popular actor-critic method well-suited for continuous action spaces; and *c)* the DR strategy used to enhance the agent's generalization by exposing it to diverse training environments.

4.2.1. *Deep Reinforcement Learning*

RL provides a framework for sequential decision-making in which an agent learns to maximize cumulative rewards by interacting with an environment (Sutton & Barto, 2018). Formally, the problem is modeled as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s' \mid s, a)$ defines the transition probabilities, $R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is a discount factor. At each time step t, the agent observes a state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$ according to its (possibly stochastic) policy $\pi(a_t \mid s_t)$, and receives a reward $r_t = R(s_t, a_t)$. The system then transitions to a new state s_{t+1} with probability $P(s_{t+1} \mid s_t, a_t)$. The goal of RL is to learn an optimal policy π^* that maximizes the expected cumulative discounted reward from each initial state. This expected return can be written as:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, s_0 = s \right] \tag{11}$$

where the expectation is taken over trajectories generated by following policy π from initial state s (Sutton & Barto, 2018). An optimal policy π^* maximizes $J(\pi)$ for all reachable states. In practice, solving for π^* exactly is intractable for large or continuous state spaces, so agents resort to iterative learning algorithms that improve the policy based on observed experience.

DRL refers to the class of RL methods that leverage deep neural networks as function approximators for the policy $\pi_\theta(a|s)$ or value functions (e.g. the state-value $V(s)$ or action-value $Q(s, a)$), allowing the agent to handle high-dimensional state inputs and complex non-linear decision boundaries (Lillicrap et al., 2016; Mnih et al., 2015). By using neural networks, DRL algorithms can generalize across states and learn directly from raw inputs such as price series or technical indicators, which is crucial in domains like finance where the state space (historical market data, technical features, etc.) can be very large.

Figure 2, illustrates the DRL process in which the agent continually interacts with the market environment in a closed loop and learns employing Deep Neural Networks (DNNs):

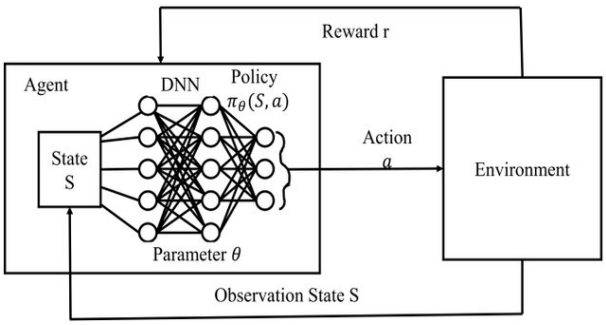

Figure 2: DRL architecture (Kabir et al., 2023).

As shown in Figure 2, in each decision interval, the agent observes the current market state (e.g.,

recent price changes, indicators, portfolio holdings), executes a portfolio action (reallocating weights among assets), and then receives a reward signal reflecting the quality of that action (for instance, the resulting portfolio return or a risk-adjusted profit). Through repeated interactions, the agent adjusts its policy to favor actions that yield higher long-term returns.

In the portfolio management setting, a typical state s_t may consist of features such as recent asset prices, returns, technical indicators, and possibly the agent's current portfolio allocation (Jiang et al., 2017; Ye et al., 2020). An action a_t is a vector of portfolio weight adjustments or asset trades, essentially specifying how to redistribute capital among the assets at time t. The reward r_t can be defined in various ways, but a common choice is the change in portfolio value (e.g., log return of the portfolio for that interval) minus any transaction costs (Yang et al., 2020; Z. Zhang et al., 2020). The agent's goal is to learn a trading strategy (policy) that maximizes the expected cumulative reward, which in this context translates to maximizing investment returns while implicitly handling risk through the reward design (e.g., using risk-adjusted returns or penalties for large drawdowns).

According to Sutton & Barto (2018), DRL algorithms can be broadly categorized into the following three categories: *a)* value-based methods, *b)* policy-based methods, and *c)* actor-critic methods:

a) **Value-based approaches**, such as *Deep Q-Networks* by Mnih et al. (2015), learn an approximate Q-function and derive a policy by selecting actions that maximize $Q(s, a)$. However, pure value-based methods struggle with continuous action spaces because finding the action that maximizes $Q(s, a)$ requires optimization in continuous domains.

b) **Policy-based methods**, such as *REINFORCE* by Williams (1992), directly optimize the policy by gradient ascent on $J(\pi)$, and can naturally handle continuous actions by parameterizing the policy $\pi_\theta(a|s)$ (e.g., as a Gaussian). They can suffer from high variance in gradient estimates and typically require careful tuning.

c) **Actor-critic methods** combine both approaches by maintaining an explicit policy (the actor) and a value function (the critic) to critique the actor's decisions, thereby reducing variance while providing guidance to the policy updates. The actor-critic framework is well-suited for continuous control problems and has shown strong performance in many domains (Lillicrap et al., 2016; Schulman et al., 2015).

In this work, we adopt an actor-critic DRL algorithm for portfolio management, which we detail next. Specifically, we use the DDPG algorithm as our learning backbone, due to its ability to handle continuous action spaces and its sample-efficient, off-policy learning characteristics.

4.2.2. *Deep Deterministic Policy Gradient*

DDPG is an off-policy, model-free, actor-critic DRL algorithm that was introduced by Lillicrap et al. (2016) to enable effective learning in continuous action spaces. It can be viewed as an extension of Deep Q-learning by Mnih et al. (2015) combined with the Deterministic Policy Gradient (DPG) framework provided by Silver et al. (2014).

In DDPG, the actor is a neural network $\mu(s|\theta^\mu)$ that deterministically maps states to a specific action (hence "deterministic" policy), and the critic is a neural network $Q(s, a|\theta^Q)$ that approximates the action-value function (i.e., the expected return from state s after taking action a and thereafter following the current policy). The critic provides feedback to improve the actor.

As shown in Figure 3, both networks interact in an iterative training cycle: the *actor* proposes actions and the *critic* evaluates them, while both are updated based on experience data:

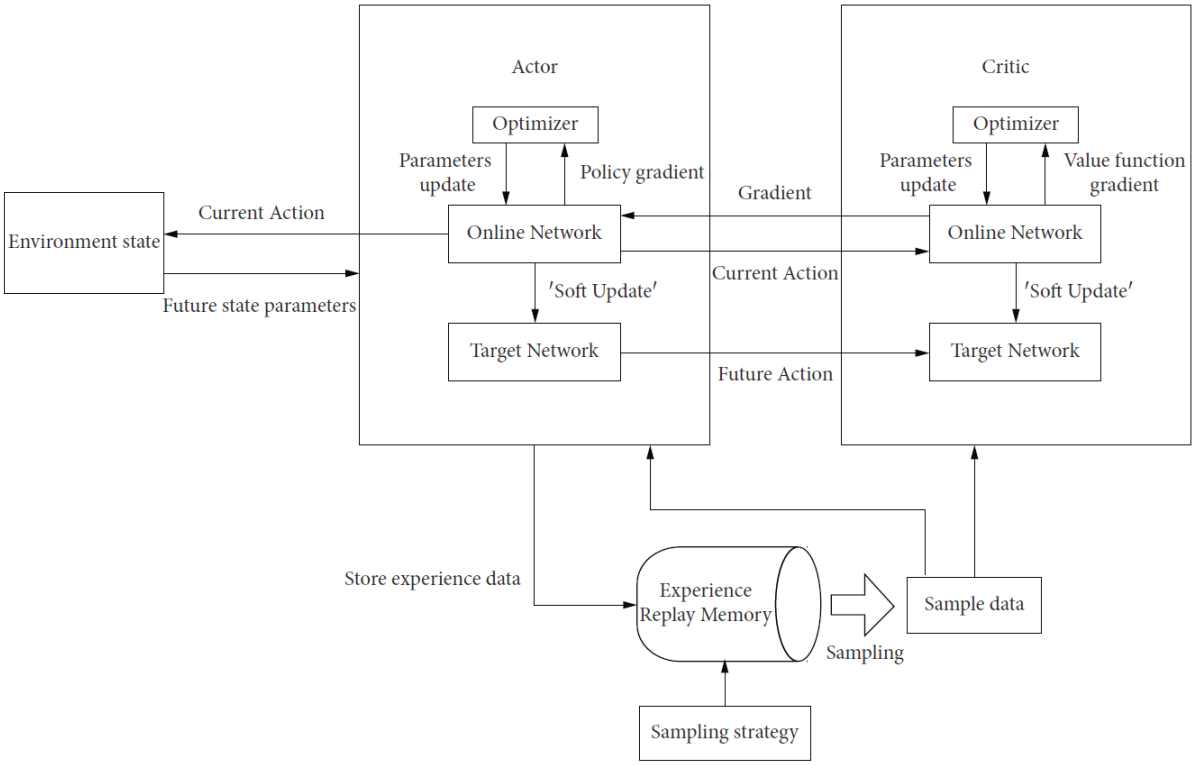

Figure 3: DDPG architecture (H. Zhang et al., 2022).

The training procedure of DDPG can be articulated through the following five systematic sequence of steps that capture the essence of its actor-critic learning framework:

1. Initialize the primary networks $Q(s, a|\theta^Q)$ and $\mu(s|\theta^\mu)$ with random weights, and create target networks Q' and μ' with identical initial weights $(\theta^{Q'} \leftarrow \theta^Q)$ and $(\theta^{\mu'} \leftarrow \theta^\mu)$.
2. Initialize an experience replay buffer $\mathcal{D}$ to store transitions (s_t, a_t, r_t, s_{t+1}).
3. For each training time step t:
   - Observe the current state s_t.
   - Select the action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$, where $\mathcal{N}_t$ is exploration noise.
   - Execute action a_t in the environment, receive reward r_t and next state s_{t+1}.
   - Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$.
4. When enough experience is collected:
   - Sample a mini-batch $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$ from $\mathcal{D}$.
   - Compute target values:
   $$y_i = r_i + \gamma\, Q'\big(s'_i,\, \mu'(s'_i|\theta^{\mu'})\big) \tag{12}$$
   - Update the critic by minimizing:
   $$L(\theta^Q) \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big(y_i - Q(s_i, a_i \mid \theta^Q)\Big) \tag{13}$$
   - Update the actor using:
   $$\nabla_{\theta^\mu} J \;\approx\; \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q(s_i, a \mid \theta^Q)\big|_{a=\mu(s_i)} \cdot \nabla_{\theta^\mu}\, \mu(s_i|\theta^\mu) \tag{14}$$

5. Update target networks via Polyak averaging:

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{aligned} \tag{15}$$

In the domain of financial portfolio management, the DDPG algorithm presents a number of distinct advantages over traditional RL methods:

a) Being an off-policy algorithm, it can leverage past experiences effectively and is not constrained to follow its current policy when gathering data, which is useful when using historical market data that can be re-sampled or replayed for training.

b) The continuous action output $\mu(s_t)$ naturally fits the portfolio allocation problem, as it can represent a continuous vector of asset weights.

c) Constraints such as budget limits or no-short-selling can be incorporated by choosing appropriate action parameterizations.

d) The critic in turn learns to evaluate the quality of a given allocation by estimating the expected future returns (or other long-term performance measures) from that state-action pair.

e) Due to its model-free nature, it does not require an explicit model of market dynamics, it learns directly from observations of price movements and rewards, thus making it flexible.

However, it is important to recognize that in stationary or narrowly defined training environments, agents based on the DDPG algorithm are susceptible to overfitting. This overfitting can significantly hinder the agent's ability to generalize to unseen market conditions. To address this limitation, our work incorporates DR into the training process.

4.2.3. *Domain Randomization*

DR is a training technique whereby one deliberately introduces variability or noise into the simulation environment's parameters in order to expose the learning agent to a wide range of scenarios (Sadeghi & Levine, 2017; Tobin et al., 2017). The key idea is that by experiencing many diverse environments in simulation, the agent learns a policy that is not tailored to any single environment configuration, and thus the policy is more likely to generalize robustly to new, unseen conditions. In the context of DRL, this approach addresses the well-known generalization problem: agents trained on a fixed environment often overfit to its particular dynamics and can falter when conditions change even slightly (Cobbe et al., 2019). By contrast, an agent trained with DR is constantly forced to adapt to varied conditions, preventing it from relying on brittle heuristics that only work in one specific scenario.

In practice, DR involves defining a set of environment parameters to randomize, along with plausible ranges or distributions for these parameters, and then sampling a new configuration from these distributions for each training episode (or periodically during training). This concept was first popularized in robotic simulation-to-real transfer. For example, Tobin et al. (2017) randomized visual attributes (textures, lighting, colors, object positions, etc.) in a simulator so that a vision-based robotic policy could better handle the gap between simulation and reality. Similarly, Sadeghi & Levine (2017) showed that randomizing camera angles and scene appearance enabled a drone's navigation policy to transfer to the real world without any real-world training. Beyond vision, other works randomize physical dynamics: Peng et al. (2018) and Pinto et al. (2017) varied physics parameters like masses, friction coefficients, and forces during training, which yielded agents that are more robust to modeling errors and changing real-world conditions. These studies demonstrated that DR can effectively act as a form of implicit data

augmentation for RL, injecting a form of structured noise into the training process that encourages the emergence of more general strategies.

Applying DR to financial portfolio management entails perturbing the simulated market environment in structured ways. In a financial simulator, we can identify key market parameters and mechanisms that, if varied, lead to different market behaviors. For instance, one can randomize:

a) The asset price dynamics, by sampling model parameters for returns (e.g., randomly adjusting drift and volatility in a stochastic price model, or shuffling/corrupting historical price sequences with noise).

b) The volatility structure, by introducing volatility regimes or shocks at random intervals (simulating calm markets vs. turbulent periods).

c) The correlation structure between assets, by randomly altering cross-asset correlations or covariances in the return generation process.

d) The transaction cost and market microstructure properties, by varying the bid-ask spread, slippage, or delay in trade execution in the simulator.

Each training episode can sample a different combination of these factors. For example, one episode might simulate a bullish market with low volatility and low transaction costs, while the next episode simulates a sideways market with occasional price jumps and higher trading frictions. By cycling through many randomized market scenarios during training the DRL agent is prevented from over-specializing to any specific market condition.

In our implementation, we integrate DR into the training loop of the DDPG agent. At the beginning of each training episode, a new set of market parameters is drawn. Concretely, we define a parameter space Θ (volatility scale, reward distribution skew, cost multiplier, etc.), and for each episode we sample $\theta \sim p(\Theta)$ from predefined distributions. The environment is then configured according to θ for the duration of that episode. The DRL agent (DDPG) trains on this episode, then in the next episode a new θ' is sampled, and so on. Over many episodes, the agent sees a wide variety of market behaviors. This procedure can be seen as training on a distribution of environments $E(\theta)$ rather than a single environment E_0. Crucially, the randomization is structured and not purely arbitrary: the ranges of parameters $p(\Theta)$ are chosen to reflect plausible real-world conditions, so that every randomized scenario remains realistic (albeit possibly extreme). This ensures that the agent's training experiences cover not only typical market variations but also rare or extreme events, which is important for financial risk management. Intuitively, DR serves a similar role as data augmentation in supervised learning; by widening the training distribution, we aim to make the learned policy invariant to irrelevant variations and robust to perturbations. By training with DR, the agent effectively "stress-tests" its strategy under many different scenarios during learning, which should improve its resilience. The expectation is that when deployed in the real market (which can be seen as yet another drawn scenario), the agent will handle it better because it has seen analogues of many different market conditions during training.

This approach directly tackles the generalization challenge identified in the literature: real financial markets are non-stationary and can undergo regime shifts that a static-trained agent would not anticipate (Charpentier et al., 2021; Cobbe et al., 2019). Indeed, previous research has noted that DRL agents trained only on historical market data often lack the ability to adapt to new market regimes (Wang et al., 2021). DR provides a systematic way to inject anticipated uncertainty into the training phase. Recent studies in quantitative finance are beginning to explore this idea. For example, Spooner & Savani (2020) propose training trading agents on multiple synthetic market scenarios to improve robustness, and Benhamou et al. (2021) suggest that training under varied market conditions can yield strategies that

maintain performance in volatile periods. However, the use of DR in finance is still nascent, and our work contributes to this emerging area by providing a concrete implementation and empirical evaluation of DR for financial portfolio management.

In summary, DR acts as a form of regularization for DRL agents, promoting the learning of strategies that generalize across environments. In the following sections, we will evaluate how incorporating DR influences the performance and generalization of our DDPG-based trading agent, comparing DR training against conventional training on fixed historical data.

5. Experimental Design & Results

In this section, we detail the experimental design implemented to rigorously evaluate the proposed methodology. This framework ensures reproducibility and enables a comprehensive assessment of model performance. The subsequent results provide quantitative and qualitative insights into the effectiveness of our approach.

5.1. Experimental Design

The experimental design focuses on optimizing DRL agents for stock trading using the FinRL framework. Historical market data for the 30 Dow Jones stocks is collected and preprocessed to serve as the environment for training. The experiment leverages Optuna to systematically tune model hyperparameters, with each trial maximizing a trade-based metric (the ratio of average winning to losing trade value). Agents are trained and evaluated across multiple trials, ensuring a robust and reproducible evaluation of hyperparameter impact on trading results, a t-test is then applied to compare the results of the agents that use DR and those who don't.

5.1.1. *Data Collection & Preparation*

For this study, historical financial data was collected for the 30 constituent stocks of the Dow Jones Industrial Average, ensuring a diverse and representative sample of the U.S. equity market. Data acquisition was performed using the Yahoo Finance API, facilitated by the FinRL framework's YahooDownloader utility, which allows for the efficient retrieval of daily price data, including OHLCV information.

5.1.2. *Experimental Setup*

The experimental environment simulates a stock trading scenario using historical market data from the Dow Jones Industrial Average. We implement the environment where an agent interacts with a dynamic portfolio allocation task across 30 assets. To evaluate the impact of DR, we define the following two training conditions:

a) **Baseline** (i.e. without DR): The agent is trained solely on fixed historical data without any modifications or stochastic perturbations.

b) **DR**: The agent is trained on a randomized environment in which key components are perturbed across episodes. Specifically, we apply randomization injecting Gaussian noise to simulate uncertainty in asset prices.

Both baseline and DR agents are trained using the same architecture and hyperparameter search space. The only difference between them lies in the nature of the environment they are exposed to. This design enables us to isolate the effect of DR on the agent's performance and generalization ability. All other factors like data sampling, model architecture, and training budget are held constant.

5.1.3. *Training Phase*

During the training phase, each agent is initialized and trained independently within the multi-asset trading environment, employing the DDPG algorithm. The agent interacted with the simulated environment by observing state representations derived from the engineered features and making sequential portfolio

allocation decisions. Throughout training, the agent's objective was to maximize a predefined reward function based on the Sharpe ratio (Sharpe, 1966).

5.1.4. *Evaluation Metrics*

To ensure statistical validity and robustness of the results, the experimental design incorporates 25 independent observations for each experimental setting. For each observation, 10 DRL agents are trained and evaluated, with the resulting Sharpe ratios (Sharpe, 1966) aggregated to obtain three summary statistics: the maximum Sharpe ratio, the third quartile, and the mean Sharpe ratio. As explained in the section corresponding to the scope and reach of the study, the analysis involves a total of three comparative assessments to evaluate the effectiveness of DR:

a) The **maximum (*H1*)** Sharpe ratio obtained across the 10 agents for each observation with DR is compared to the mean Sharpe ratio from the corresponding non-DR baseline.

b) The **third quartile (*H2*)** of the Sharpe ratios under DR is compared to the mean of the Sharpe ratios without DR.

c) The **average (*H3*)** Sharpe ratio with DR is evaluated against the baseline Sharpe ratio.

These comparisons collectively provide insight into the benefits of DR not only in terms of peak agent performance but also in improving the distribution and average quality of trading outcomes, thus illustrating the different levels of achievement that these can reach.

5.1.5. *Hypothesis Testing*

To ensure statistical validity, we perform hypothesis testing to evaluate the effect of DR. All experiments use the Sharpe ratio (Sharpe, 1966) as the performance metric, evaluated on a fixed test set. A total of 25 independent training runs are conducted for each condition (with DR and without DR). A two-sample t-test is used to compare the mean Sharpe ratios. We employ the one-sided (right-tailed) t-test formula:

$$t = \frac{\mu_x - \mu_0}{\frac{s}{\sqrt{n}}} \tag{16}$$

Welch's correction is applied in cases where the assumption of equal variances is violated (Welch, 1947); similarly, the non-parametric Mann-Whitney U test is implemented when the assumption of statistical normality is not met (Mann & Whitney, 1947).

5.1.6. *Confidence Intervals*

The tests are one-sided (i.e. right-tailed) and conducted at a rigorous 1% significance level ($\alpha = 0.01$), which is justified given the nature of the experiment and the increased likelihood of Type I errors due to multiple comparisons and stochastic variability (Kim, 2015).

5.2. Results

In this section, we present the outcomes of the experimental procedure described above. The results highlight the comparative performance of agents trained with and without DR, evaluated through financial metrics and statistical significance tests. We calculate the Baseline Agent Sharpe Ratio, then we calculate the recorded Sharpe Ratios of the 10 agents for each of the 25 observations, and afterwards we perform

a one-sided t-test. This analysis serves to quantify the impact of DR on agent robustness and trading effectiveness.

5.2.1. *Baseline Sharpe Ratio (without DR)*

The Baseline Sharpe Ratio recorded for the DRL agent without injecting any noise perturbation, stochasticity or DR. We will use this value as a reference point to assess the potential improvement introduced by incorporating DR during the training phase.

	Baseline Agent
Sharpe Ratio	1.601392

Table 1: Baseline Sharpe Ratio (without DR)

5.2.2. *Sharpe Ratios employing Domain Randomization*

The Sharpe ratios recorded for the 10 agents in their 25 observations are summarized in the following table according to the metrics previously discussed:

	H1	***H2***	***H3***
Experiment nº	$\max[\text{Sharpe}_{\text{DR}}]$	$Q3[\text{Sharpe}_{\text{DR}}]$	$\mu[\text{Sharpe}_{\text{DR}}]$
1	1.863253	1.732702	1.606896
2	2.006231	1.855320	1.765119
3	1.872960	1.788143	1.659716
4	1.832878	1.746652	1.626307
5	1.970346	1.849455	1.757641
6	1.913919	1.803041	1.617699
7	2.034160	1.827360	1.630844
8	1.911241	1.748060	1.689416
9	1.742590	1.617830	1.548108
10	1.726062	1.671880	1.553235
11	2.019963	1.767428	1.723631
12	1.768805	1.732181	1.703302
13	1.900987	1.730282	1.694839
14	1.953399	1.759086	1.709777
15	1.953691	1.759004	1.697709
16	1.953568	1.728486	1.653343
17	1.873078	1.720458	1.625542
18	1.970622	1.813374	1.756471
19	1.949268	1.773935	1.710465
20	1.806630	1.710974	1.601419
21	1.749910	1.704215	1.613061
22	1.888595	1.729161	1.659456
23	2.049885	1.781178	1.718880
24	1.848652	1.770212	1.656863
25	1.882910	1.701051	1.635116

Table 2: Metrics from Sharpe Ratios across 25 DR Agent Experiments

5.2.3. *One-sided t-test Results*

In order to determine the effectiveness of DR techniques we perform a t-test comparing the Baseline Sharpe ratio from table 1 to the different ratios shown in table 2.

	H1	***H2***	***H3***
	$\max[\text{Sharpe}_{\text{DR}}]$	$Q3[\text{Sharpe}_{\text{DR}}]$	$\mu[\text{Sharpe}_{\text{DR}}]$
t-statistic	-15.95338723	-14.04511879	-5.24692208
p-value	8.07×10^{-19}	1.27×10^{-17}	3.46×10^{-6}

Table 3: One-sided t-test results comparing DR agents to the historical (no DR) agent

5.2.4. *Discussion*

While the t-test results indicate that DR has a strong effect on agent performance, these findings should be interpreted with caution. DR techniques are designed to mitigate overfitting by introducing controlled stochasticity during training, which in turn encourages agents to learn more generalizable policies. This mechanism can plausibly account for the statistical significance observed in the t-test results.

Nonetheless, a conservative interpretation would posit that the absence of DR increases the likelihood of overfitting to specific patterns in the training data, whereas its implementation promotes robustness and better generalization. Under this view, the observed improvements in Sharpe ratios may stem more from a reduction in overfitting than from a net gain in predictive power or trading intelligence.

In light of these considerations, the results suggest that DR contributes to producing more reliable and repeatable outcomes across various performance metrics. However, to validate these findings in real-world financial settings, additional robustness checks are warranted.

5.2.5. *Reproducibility*

To ensure reproducibility, all code and documentation used in this thesis are publicly available in the associated GitHub repository: `https://github.com/GaussLighter/DR4DRL`. The repository contains the full Jupyter Notebook used for training and evaluation (`DR4DRL.ipynb`) as well as this manuscript. The implementation relies on widely used open-source libraries such as `FinRL`, `Optuna`, and `PyTorch`, and all dependencies are listed in a `requirements.txt` file. While the dataset used for training is not included due to size constraints, all data can be automatically retrieved from Yahoo Finance through the `FinRL` framework's built-in downloader. This setup enables third parties to replicate the experiments and verify the results under the same conditions, reinforcing the transparency and robustness of the study.

6. Conclusions & Future Work

In this section, we synthesize the key findings derived from our experiments and analyses, highlighting the extent to which our proposed methodology has met the objectives set forth at the beginning of this study. We also identify the limitations encountered throughout the research process and suggest potential avenues for future work that could enhance, generalize, or further validate the approach. These directions aim to bridge the current gaps, refine the proposed framework, and encourage continued exploration within the domain of financial portfolio management.

6.1. Conclusions

This research demonstrates that applying DR during the training of DRL agents yields statistically and practically significant improvements in performance and generalization. Through a robust experimental design, we tested three hypotheses regarding the improvement in Sharpe ratios due to DR. All three hypotheses were supported by the data:

a) The best-case Sharpe ratio improved substantially under DR.

b) The third quartile ($Q3$) of DR-trained agents surpassed the baseline.

c) The average Sharpe ratio across DR agents was higher than the baseline.

Statistical testing confirmed these differences to be significant ($p < 0.01$ in all cases), suggesting that DR contributes not just to peak outcomes, but also to consistency and stability across training runs.

These results imply that DR serves as an effective regularizer, reducing overfitting and fostering strategies that are robust to changes in market regimes. Agents trained on a variety of randomized environments generalized better to out-of-sample data, maintaining higher Sharpe ratios in unseen conditions. Overall, DR appears to be a powerful mechanism for improving the reliability and robustness of DRL-based trading agents in financial applications.

6.2. Future Work

Despite the demonstrated benefits of DR in mitigating overfitting, further reducing overfitting remains a promising avenue for future research. Financial markets are highly non-stationary and noisy, so even a domain-randomized agent may inadvertently learn spurious patterns that do not generalize to new conditions (Sutton & Barto, 2018). Strengthening the agent's ability to generalize beyond its training experience is therefore paramount. In this regard, several extensions and improvements can be considered to enhance the robustness and adaptability of DRL agents for portfolio management.

One promising direction is to deploy an ensemble of DR agents to mitigate variance and capture a wider array of market behaviors. Instead of relying on a single agent, multiple parallelized agents would be trained, each on different randomized market scenarios or with different random seeds, yielding a diverse set of trading policies. By combining their decisions (for example, by averaging action preferences or voting), the ensemble can smooth out idiosyncratic behaviors of any individual agent. This approach takes inspiration from ensemble methods in machine learning, which often achieve lower generalization error by averaging out individual models' biases and noise. In the context of DRL, an ensemble can reduce the variance of returns and provide more stable performance across market regimes. Prior work in RL suggests that such diversity-driven ensembles not only improve robustness but also can implicitly estimate uncertainty, leading to more cautious and reliable decision-making (Osband et al., 2016). In

essence, each agent in the ensemble serves as an "expert" on certain market patterns, and their aggregate output would potentially yield a more consistent and robust trading strategy than any single policy alone.

However, introducing an ensemble raises the question of how to decide which agent's decisions to trust at any given time. Rather than a naive averaging of actions, a more sophisticated mechanism could dynamically assign decision authority to the agent (or subset of agents) most competent in the current market context. Designing such a gating or selection mechanism is an open research challenge, but several criteria and heuristics can be explored as the basis for decision authority:

a) **Recent Performance**: One intuitive criterion is to monitor each agent's recent performance and risk-adjusted returns. For example, the agent with the highest rolling Sharpe ratio over a recent window could be given greater weight or even full control of the portfolio decisions until another agent demonstrably outperforms it. This approach treats the ensemble as a competition, where the best-performing policy in the current market regime takes the lead. It ensures that, at any moment, the strategy with the most favorable balance of return and volatility (as evidenced by recent Sharpe ratio) drives the decision-making. Such a scheme would need safeguards (e.g., minimum lookback period or performance difference thresholds) to avoid excessive switching due to short-term noise, but it could effectively adapt the policy to regime changes by favoring whichever agent is currently excelling.

b) **Environment Similarity**: Another promising metric for agent selection is the similarity between the current market environment and the training domain of each agent. If each agent in the ensemble is subtly specialized (whether intentionally or as a byproduct of training on different random market parameters), we can compute features of the ongoing market (trendiness, volatility, correlation structure, etc.) and compare them to the environments where each agent performed best. For instance, one agent might be particularly adept in high-volatility markets (as it was trained on or evolved to handle large price swings), while another excels in calm, mean-reverting conditions. By defining a distance or similarity measure in this feature space of market conditions, the system can activate the agent whose "expertise" best matches the present regime. This approach is akin to a contextual or regime-based gating: the ensemble controller essentially asks, "Which agent was built for a market like this?" and biases the decision toward that agent. Implementing this might involve clustering historical scenarios, training a meta-classifier to recognize market regimes, or using measures like volatility indices, trend strength indicators, or reward patterns to identify the closest match between current and past conditions. Over time, this could lead to a form of specialist ensemble where each agent handles the scenarios it knows best, improving overall performance across diverse conditions.

c) **Bayesian Confidence Estimates**: A more principled approach could leverage uncertainty estimates from each agent to guide the ensemble's choice. If each agent can quantify its confidence in the current state or action (for example, via the variance of its value function, an approximation of a Bayesian posterior, or the disagreement among an agent's internal models), the ensemble controller could assign higher weight to the agent that is most confident about the correct action. Conversely, if an agent's predictions carry high uncertainty in a particular situation, it might defer to others that are more sure-footed. This idea resonates with Bayesian RL principles and ensemble methods that treat the spread in predictions as a measure of epistemic uncertainty (Osband et al., 2016). In practice, one could implement this by having each agent output not only an action recommendation but also an uncertainty score (or confidence level) for that recommendation. The agent with the lowest estimated uncertainty (or highest confidence) for the current decision would then be trusted more heavily. This could be realized through techniques like bootstrapped Q-networks or Monte

Carlo dropout (if using policy networks), where the variance across predictions serves as the confidence metric. By harnessing these uncertainty signals, the ensemble as a whole becomes risk-aware, potentially avoiding overconfident bets in unfamiliar states and leaning on the most knowledgeable agent for the task at hand.

It is worth noting that these criteria for agent selection are not mutually exclusive; they could be combined or used in different layers. For example, a hierarchical approach might first filter agents by environment similarity (choosing a subset most suited to the detected regime) and then select or weight among those based on recent performance and confidence levels. Designing an optimal gating mechanism could be formulated as another learning problem in itself. One could train a meta-controller (a high-level policy) that observes state features or summary statistics (including each agent's confidence and recent returns) and outputs a weighting over the ensemble's agents. Such a meta-controller could be trained via reinforcement learning or bandit algorithms, with the reward being the portfolio performance, thereby learning to blend the experts in a way that maximizes long-term returns. This approach is conceptually related to the mixture-of-experts frameworks in machine learning, where a gating network learns to route each input to the most appropriate expert. In the context of our problem, the "input" is the current market state (and possibly the agents' internal signals), and the "experts" are the individual domain-randomized agents.

By pursuing an ensemble-of-agents strategy with an intelligent selection mechanism, future work can further guard against overfitting and improve adaptability. The ensemble's diversity provides a hedge against the risk that any single policy was over-specialized or lucky in backtesting, thereby addressing one of the core concerns in algorithmic trading research—that a model might simply be exploiting historical noise (Sutton & Barto, 2018). If one agent overfits to a particular artifact of the training data, its poor confidence or subpar recent Sharpe ratio in a new regime would cause the ensemble to shift focus to other agents, thus reducing the chance of catastrophic failures. This dynamic approach would make the trading system more robust to regime shifts or unmodeled market phenomena, as it can quickly pivot to a different decision-maker more suited to the new conditions.

Finally, beyond the ensemble techniques, continued evaluation and validation of these methods in varied settings is crucial. Future research should test DR ensembles on different asset classes and longer time horizons, and possibly in live or paper-trading environments with greater granularity, to ensure that the gains in generalization observed in our study translate to real-world reliability. It would also be valuable to incorporate transaction costs, market impact, and other practical constraints into the training and evaluation; reducing overfitting is not only about achieving high Sharpe ratios on paper, but also about maintaining performance once all frictions are accounted for. As we extend these approaches, careful risk management and statistical validation (e.g., out-of-sample tests, rolling window analysis, and significance checks as used in this thesis) will remain essential to confirm that improvements are genuine and not the result of new forms of overfitting.

In summary, enriching our DRL framework with ensemble methods and intelligent agent selection mechanisms offers a promising path forward to build trading agents that are more robust, generalizable, and resilient in the face of the ever-changing financial markets. By tackling overfitting head-on and giving the agent toolbox the ability to adapt or diversify its decisions, we move closer to safe and effective deployment of DRL in real-world portfolio management.

Declarations

Funding: This research received no external funding.

Conflicts of Interest: The authors declares no conflict of interest.

Ethical Approval: Not applicable.

Data Availability: The datasets used in this study are publicly available through the FinRL library and associated GitHub repositories.

Author Contributions: The entirety of this thesis was conceived, implemented, and written by the authors as part of the Master's program in Business Analytics at Comillas Pontifical University.

References

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., & Zhang, L. (2019). Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*. https://doi.org/10.48550/arXiv.1910.07113

Anschel, O., Baram, N., & Shimkin, N. (2017). Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Bailey, D. H., Borwein, J. M., Lopez de Prado, M., & Zhu, Q. J. (2014). Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. *Notices of the AMS*, 61(5), 458–471.

Benhamou, E., Saltiel, D., Ohana, J.-J., & Atif, J. (2021). Detecting and adapting to crisis patterns with context-based deep reinforcement learning. *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, 10050–10057.

Berkowitz, S. A., Logue, D. E., & Noser, J., Eugene A. (1988). The total cost of transactions on the nyse. *The Journal of Finance*, 43(1), 97–112. https://doi.org/10.1111/j.1540-6261.1988.tb02591.x

Brock, W., Lakonishok, J., & LeBaron, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *The Journal of Finance*, 47(5), 1731–1764. https://doi.org/10.1111/j.1540-6261.1992.tb04681.x

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

Cobbe, K., Klimov, O., Hesse, C., Kim, T., & Schulman, J. (2019). Quantifying generalization in reinforcement learning. *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 1282–1289.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25(2), 383–417.

Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1), 3–56. https://doi.org/10.1016/0304-405X(93)90023-5

Farebrother, J., Hughes, M. C., & Doshi-Velez, F. (2018). Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Garrido-Merchán, E. C. (2025). Information-theoretic bayesian optimization: Survey and tutorial [arXiv preprint arXiv:2502.06789]. https://arxiv.org/abs/2502.06789

Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*.

Kabir, H., Tham, M.-L., & Chang, Y. C. (2023). Twin delayed ddpg based dynamic power allocation for mobility in IoRT. *Journal of Communications Software and Systems*, 19(1), 19–29. https://doi.org/10.24138/jcomss-2022-0141

Kim, J. H. (2015). *How to choose the level of significance: A pedagogical note* [MPRA Paper No. 66373]. https://mpra.ub.uni-muenchen.de/66373/1/MPRA_paper_66373.pdf

Laskin, M., Srinivas, A., & Abbeel, P. (2020). Curl: Contrastive unsupervised representations for reinforcement learning. *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Lee, K., Levine, S., Abbeel, P., & Laskin, M. (2021). Sunrise: A simple unified framework for ensemble reinforcement learning. *International Conference on Machine Learning (ICML)*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lo, A. W., Mamaysky, H., & Wang, J. (2000). Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The Journal of Finance*, 55(4), 1705–1765. https://doi.org/10.1111/0022-1082.00265

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1), 50–60.

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

Menkhoff, L. (2010). The use of technical analysis by fund managers: International evidence. *Journal of Banking & Finance*, 34(11), 2573–2586. https://doi.org/10.1016/j.jbankfin.2010.04.014

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533. https://doi.org/10.1038/nature14236

Murphy, J. J. (1999). *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. New York Institute of Finance.

Osband, I., Blundell, C., Pritzel, A., & Roy, B. V. (2016). Deep exploration via bootstrapped dqn. *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*.

Park, C.-H., & Irwin, S. H. (2007). What do we know about the profitability of technical analysis? *Journal of Economic Surveys*, 21(4), 786–826. https://doi.org/10.1111/j.1467-6419.2007.00519.x

Park, H., Sim, M., & Choi, D. (2020). An intelligent financial portfolio trading strategy using deep q-learning. *Expert Systems with Applications*, 158, 113573.

Peng, X. B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. *IEEE International Conference on Robotics and Automation (ICRA)*. https://doi.org/10.1109/ICRA.2018.8460528

Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust adversarial reinforcement learning. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2817–2826.

Raileanu, R., Maksymets, O., Pan, X., Schulman, J., Kostrikov, I., & Abbeel, P. (2021). Automatic data augmentation for generalization in reinforcement learning. *NeurIPS*.

Rakelly, K., Zhou, A., Quillen, D., Finn, C., & Levine, S. (2019). Efficient off-policy meta-reinforcement learning via probabilistic context variables. *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Sadeghi, F., & Levine, S. (2017). CAD^{2}RL: Real single-image flight without a single real image. *Proceedings of Robotics: Science and Systems (RSS)*.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015). Trust region policy optimization. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 37, 1889–1897. http://proceedings.mlr.press/v37/schulman15.html

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3), 425–442. https://doi.org/10.2307/2977928

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of Business*, 39(1), 119–138.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 387–395.

Spooner, T., & Savani, R. (2020). Robust market making via adversarial reinforcement learning. *arXiv preprint arXiv:2003.01820*.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 23–30.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T.-J., & Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 969–977.

Wang, R., Wei, H., An, B., Feng, Z., & Yao, J. (2021). Commission fee is not enough: A hierarchical reinforced framework for portfolio management. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1), 626–633. https://doi.org/10.1609/aaai.v35i1.16142

Welch, B. L. (1947). The generalization of "student's" problem when several different population variances are involved. *Biometrika*, 34(1-2), 28–35.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4), 229–256. https://doi.org/10.1007/BF00992696

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

Ye, Y., Pei, H., Wang, B., Chen, P.-Y., Zhu, Y., & Li, X. (2020). Reinforcement-learning based portfolio management with augmented asset movement prediction states. *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 34(1), 1112–1119.

Yu, P., Lee, J. J., Kulyatin, I., Shi, Z., & Dasgupta, S. (2019). Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740*.

Zhang, H., Xu, J., Zhang, J., & Liu, Q. (2022). Network architecture for optimizing deep deterministic policy gradient algorithms. *Computational Intelligence and Neuroscience*, 2022, 1–10. https://doi.org/10.1155/2022/1117781

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