UNIVERSITY OF TURKU
Turku School of Economics

AI-Driven Portfolio Management: A Comparative Research of Deep Reinforcement Learning Techniques Against The 1/N Portfolio Strategy

Master's thesis
in Accounting and Finance

Author:
Joni Aarnio

Supervisor:
Prof. Luis Alvarez Esteban

18/09/2025
Turku

Master's thesis

Subject: Accounting and Finance
Author: Joni Aarnio
Title: AI-Driven Portfolio Management: A Comparative Research of Deep Reinforcement Learning Techniques against the 1/N Portfolio Strategy
Supervisor: Luis Alvarez Esteban
Number of pages: 70
Date: 18/09/2025

Recent advances in deep reinforcement learning (DRL) for portfolio management offers promising methods, yet their real-world edge over simple heuristic allocations remains unclear. This thesis evaluates whether state-of-the-art DRL agents can outperform the naive but hard-to-beat 1/N strategy. Three algorithms: Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C) and Deep Deterministic Policy Gradient (DDPG) are trained to allocate capital across ten highly liquid U.S. large-capitalisation equities drawn from diverse sectors. Daily total-return data from January 2010 to December 2024 are split chronologically: 2010-2019 forms the learning window, while 2020-2024 provides an untouched out-of-sample testing period, capturing the COVID-19 shock and subsequent regime shifts.

The study contributes a rigorously controlled, multi-algorithm comparison that integrates realistic costs and robust statistics. The environment frames portfolio management as a sequential Markov decision process. Each state aggregates recent price dynamics, technical indicators, rolling fundamentals and macro variables where actions are continuous weight vectors constrained to full investment. A risk-adjusted reward embeds a 10 bp transaction-cost penalty to discourage excessive turnover. Hyper-parameters are tuned via grid search, and model robustness is checked across multiple random seeds.

Out-of-sample results reveal that none of the DRL agents delivers a statistically significant improvement over equal weighting. The 1/N benchmark achieves a compound annual growth rate of 20.9 % and the highest annualised Sharpe ratio (1.075), marginally ahead of DDPG (0.916), A2C (0.840) and PPO (0.805). A Ledoit-Wolf circular block bootstrap with 1 000 replications finds p-values between 0.46 and 0.51 for Sharpe-ratio differentials, confirming that observed gaps are indistinguishable from noise at conventional significance levels. Overall, the evidence indicates that algorithmic ingenuity alone does not guarantee superior risk-adjusted returns in liquid equity markets.

AI disclaimer: AI-based tools, particularly ChatGPT and Grammarly AI, were used during the research for language editing, project coding, and LaTeX formatting.

Key words: Reinforcement learning, Stock markets, Portfolio management

Gradututkielma

Oppiaine: Laskentatoimi ja rahoitus
Tekijä: Joni Aarnio
Otsikko: Tekoälyvetoinen salkunhoito: Vertaileva tutkimus syvävahvistusoppimisen menetelmistä suhteessa 1/N-salkkustrategiaan
Ohjaaja: Luis Alvarez Esteban
Sivumäärä: 70
Date: 18/09/2025

Viimeaikaiset edistysaskeleet syvävahvistusoppimisen saralla (DRL) salkunhoidossa tarjoaa lupaavia menetelmiä, mutta niiden todellinen etu verrattuna yksinkertaisiin heuristisiin allokointisääntöihin on yhä epäselvä. Tämä pro gradu -tutkielma selvittää, kykenevätkö huipputason DRL-agentit päihittämään naivin mutta vaikeasti voitettavaksi tunnetun 1/N-strategian. Kolme algoritmia: Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C) ja Deep Deterministic Policy Gradient (DDPG) koulutetaan kohdentamaan pääomaa kymmeneen hyvin likvidiin yhdysvaltalaiseen suuryhtiöosakkeeseen useilta toimialoilta. Päivittäinen kokonaistuottodata ajalta tammikuu 2010 – joulukuu 2024 jaetaan kronologisesti: vuodet 2010–2019 muodostavat oppimisjakson, kun taas 2020–2024 toimii koskemattomana ulkoisen testauksen ajanjaksona, kattaen muun muassa COVID-19-shokin ja sitä seuranneet rakennemuutokset.

Tutkimus tarjoaa tiukasti kontrolloidun, useita algoritmeja vertailevan asetelman, joka yhdistää realistiset kustannukset ja vankan tilastollisen analyysin. Salkunhoito mallinnetaan peräkkäisenä Markovin päätösprosessina, jossa tilavektori koostaa viimeaikaiset hintaliikkeet, tekniset indikaattorit, rullaavat fundamentit ja makromuuttujat ja jossa toiminnot ovat jatkuvia painovektoreita, joiden on täytettävä täysinvestoinnin ehto. Riskikorjattu palkkio sisältää 10 korkopisteen transaktiokustannuspenaltin liiallisen vaihtuvuuden hillitsemiseksi. Hyperparametrit viritetään ruutuhakumenetelmällä, ja mallien kestävyys testataan useiden satunnaissiementen avulla.

Ulkoisen testiaineiston tulokset osoittavat, ettei mikään DRL-agenteista saavuta tilastollisesti merkittävää parannusta tasapainottuvaan 1/N-strategiaan nähden. Vertailustrategia tuottaa 20,9 %:n yhdistetyn vuotuisen kasvuvauhdin ja korkeimman annualisoidun Sharpe-suhteen (1,075), niukasti DDPG:n (0,916), A2C:n (0,840) ja PPO:n (0,805) edellä. Ledoit–Wolfin syklinen lohkobootstrap (1 000 replikointia) antaa Sharpe-eroille p-arvot 0,46–0,51, mikä vahvistaa, että havaitut erot ovat perinteisin raja-arvoin erottamattomia satunnaisvaihtelusta. Tulokset viittaavat siihen, että jopa kehittyneet DRL-mallit jäävät likvideillä osakemarkkinoilla yksinkertaisen, kustannustehokkaan 1/N-strategian varjoon.

Tekoälyseloste: Tutkielman laatimisessa on hyödynnetty tekoälypohjaisia työkaluja, erityisesti ChatGPT:tä ja Grammarly AI:ta, kielenhuoltoon, projektikoodin tuottamiseen ja LaTeX-muotoiluun.

Avainsanat: Vahvistusoppiminen, Osakemarkkinat, Salkunhoito

CONTENTS

FIGURES

TABLES

1 INTRODUCTION

1.1 Background

Every rational investor wants to maximize the utility received from their investments. This stems to the central research topic in finance about the profitability of active portfolio management and whether it is possible to consistently outperform the market benchmarks. One of the cornerstones in finance theory is the Efficient Market Hypothesis (EMH) proposed by Fama (1970), which states that in efficient markets asset prices fully reflect all available information. This theory suggests that all new information is incorporated into stock prices instantly, making it difficult to systematically outperform benchmark indexes even trough advanced modelling techniques. This assumption challenges the ability of even the most advanced methods such as deep reinforcement learning (DRL), to outperform the market benchmark if the market is truly efficient.

Alternative theories have been proposed to challenge the idea of perfectly efficient markets and to provide additional understanding about the price information. Behavioral Finance argues that individual cognitive biases, such as overconfidence or loss aversion, play significant role in investment decisions and can systematically differ from rational expectations (Barberis and Thaler, 2003, 1063-1070). This notion is extended by heuristics that are fuelled by emotions, such as fear, that propagate these biases throughout financial markets (Shiller, 2017, 974). These perspectives underscore that market behavior are not only shaped by informational efficiency but also by investors psychological and behavioral factors, which can create exploitable, albeit transient, anomalies. The Adaptive Market Hypothesis (AMH), proposed by Lo (2004), takes into account principles from EMH but also from Behavioral Finance. AMH acknowledges that market efficiency is dynamic and instead of being static market efficiency becomes more efficient as investors adapt to changing environments. Therefore AMH framework suggests that markets are capable for temporary inefficiencies, as well as learning mechanisms occurring among market participants.

Modern artificial intelligence (AI) and machine learning (ML) technologies are providing new non-linear multiphase methods that can be used in portolio management to support investment decisions in complex market conditions. These methods now challenge traditional portfolio management models that are widely used in both academia and institutional portfolio management. The Capital Asset Pricing Model (CAPM), first introduced by Sharpe (1964), has generally been a foundational approach in the field. CAPM is built on the presumption that all equity risk premiums are originated from a single market risk factor, also known as systematic risk. The model suggests that asset returns are linearly

related to market movements, implying that equity prices are purely determined by this single market factor. Even though CAPM and similar models have laid the foundation for portfolio theory, they are often viewed problematic because they often make assumptions which oversimplify the financial markets that in reality hold complex dynamics.

Recent research by Gu et al. (2020) suggests that multi-factor models and non-linear forecasting methods can significantly enhance prediction accuracy, as they can capture a broader range of variables influencing market behaviour. This shift has provoked a new wave of research that focuses on leveraging advanced machine learning techniques, such as DRL, to identify non-linear patterns within complex financial data structures and to improve investment outcomes. An advanced branch of machine learning, DRL, combines reinforcement learning (RL) principles with deep learning architectures. This combination allows DRL agents to make autonomous, data driven decisions in uncertain and constantly shifting market conditions. DRL's adaptability is a particularly valuable attribute, as it can optimize portfolio allocations by processing and reacting to vast quantities of real-time data more effectively than static models. Recent study by Huang et al. (2022) has shown DRL's capacity to consistently optimize returns while maintaining lower transaction costs by employing short-selling and arbitrage mechanisms. These findings underscore the potential for DRL-based models to outperform traditional strategies, especially in highly volatile environments. Additionally, another key advantage of DRL based models is its potential for explainability, which is crucial in financial decision-making. While machine learning models are usually considered as hardly explainable black box types, techniques like integrated gradients, for example a study by Guan and Liu (2021), have been developed to elaborate which data features influence the agent's decision-making process. This transparency can help enhance investors trust in AI-driven investment strategies and make them more understandable.

Existing research of deep reinforcement learning in finance context often involve methodological shortcomings. Key essential shortcome is using single algorithms against weak baselines and lack of in-depth analysis of the results. In this thesis these gaps are addressed by providing added value on several fronts. First, a controlled framework for multiple key DRL algorithms is deployed and evaluated against a robust 1/N portfolio benchmark aligning this research with important strand of literature by DeMiguel et al. (2009) that calls for caution against complex models. More importantly, this study provides deeper analysis to uncover the sources of returns by linking empirical results to financial theory. For example, agent's success might stem from genuine market-timing ability challenging Efficient Market Hypothesis or simply by taking on greater systematic risk that could be explained by CAPM. Further, these findings are elevated with statistical testing to assess whether any outperformance is statistically significant. Finally, the durability of the models is verified through grid search and multiple seeds simulation

to account for hyperparameter variations and stochastic factors to offer more robust and repeatable findings.

In summary, DRL models represent a new promising orientation in portfolio management, not only because they allow better advanced adaptability but also for their capabilities to deal with complex and interconnected nature of modern financial market environments. By employing DRL, this study aims to find out whether AI-driven strategies can deliver superior risk-adjusted returns than traditional allocation methods. This research addresses a critical need in finance for sophisticated, responsive strategies that align with the unpredictable nature of contemporary markets and investor expectations.

1.2 Objectives and structure

The aim of this research is to examine the applicability and potential of deep reinforcement learning agents in portfolio management by evaluating risk-adjusted returns against traditional strategies, especially the 1/N allocation strategy. The 1/N allocation strategy, also known as naive diversification strategy, is a very common strategy among investors due to its ease of use and equal asset weights (DeMiguel et al., 2009, 1916-1917). For academic purposes naive allocation strategy provides a solid benchmark with no regard for market variation. Since financial markets environment is very complex and volatile, there is demand for more advanced and adaptable predictive models that can process real-time data to dynamically adjust portfolio allocations. DRL offers a platform to optimize long-term rewards through continuous learning, which may offer significant advantages over such traditional methods.

Despite the principles of EMH, evidence of market inefficiencies, such as behavioral biases and temporary mispricings can provide opportunities for DRL to potentially outperform traditional models. This leads to a null hypothesis in this study that DRL strategies will not systematically outperform a market benchmark, aligning with EMH's implication of market unpredictability. Hypothesis testing within a DRL context provides a critical evaluation for the EMH under practical conditions and highlighting any noticeable deviations that DRL may capitalize on.

To investigate this hypothesis, the study is examined by a main research question that is whether can deep reinforcement learning (DRL) achieve better portfolio performance than the naive 1/N allocation strategy in terms of risk-adjusted returns. To provide a clear answer, this question is addressed with two sub-questions. The first sub-question seeks to answer which DRL algorithms are most effective in optimizing portfolio allocation. Then the second sub-question is a direct comparison of using DRL-based portfolio management with the 1/N strategy, evaluating their performance based on risk-adjusted returns.

The rest of the thesis is organized as follows. Chapter 2 delves into the foundational finance theories behind portfolio management. Chapter 3 introduces the machine learning framework to provide solid understanding of research methods and to contextualize the potential and limitations of AI-driven strategies in theoretical context. Together these sections supply the conceptual formation for subsequent modelling and benchmarking. Additionally also wrapping earlier research results in to context. Chapter 4 discusses methodology used in this research detailing data pipeline, agent environment, state representation, model training, and statistical validation methods. Chapter 5 discusses and represents empirical results providing observations from the outcome. Conclusions are presented in Chapter 6, closing the research by summarising the contributions, acknowledging limitations, and suggesting avenues for further research. Additionally, appendixes for pseudo algorithms are provided in the end along with references. By progressing from theory to method to evidence, this structure ensures that each chapter addresses a distinct level of inquiry while laying the groundwork for the next, thereby providing a coherent narrative from initial motivation to actionable insights.

2 THEORETICAL FRAMEWORK

2.1 Stock market predictability

The Efficient Market Hypothesis (EMH) is introduced in this chapter as a foundational theory in finance. First proposed by Fama (1970), EMH states that financial markets are informationally efficient, meaning the prices of assets fully reflect all available information in markets. In an informationally efficient market, all new information such as earnings news or macroeconomic data reflects into stock prices so fast that no trading strategy can systematically earn excess risk-adjusted returns beyond what could be expected by chance. In other words, one cannot "beat the market" consistently except by luck or by accepting higher risk. Closely associated with EMH is so called Random Walk Theory. Malkiel (2003) argues that stock market prices are following so-called "random walk" which means that stock prices exhibit random and unpredictable patterns. Under EMH, price changes follow a near-random walk because any predictable patterns would be arbitraged away by informed traders (Malkiel, 2003, 59-72). Jones and Netter (2008) add that because new information is randomly favourable or unfavourable towards expectations, all changes in stock prices in an efficient market should be random and result as random walk in stock prices.

This information efficiency is divided into three stages by Fama (1970): weak form efficiency, semi-strong efficiency, and strong efficiency. The weak form efficiency is suggesting that current prices are reflecting all past trading information, implying that technical analysis can not provide an edge against the markets. Semi-strong efficiency suggests that all publicly available information is already reflected in the prices. This includes any public information about the company beyond historical price data, such as earnings reports, market news, and analyst reports. This form suggests that fundamental based analysis can not gain consistent advantage over the markets. Strong-form efficiency, suggests that even the insider information is priced in the assets, meaning that no investor could be able to achieve excess returns even through information based strategies.

For a predictive model to be able to function in the context of stock markets, there needs to be an assumption made about market efficiency. If markets represent full efficiency, these predictive models would not be beneficial in forecasting asset performance. Additionally, according to Random Walk Theory price changes are independent and identically distributed and also hold no memory meaning that historical prices give no indication of future (Fama, 1965, 34). This leads to a point where any agent that tries to find patterns in historical price data would theoretically find none that persist, as markets have no memory beyond random noise. Therefore, full market efficiency would mean that any

changes in asset prices would be caused by new information, which cannot be predicted by financial models. Additionally, if the EMH holds, at least in its weak or semi-strong forms, it casts doubt on the efficacy of any complex trading strategy.

The EMH provides a skeptical baseline: any observed outperformance by a DRL strategy might indicate a challenge to market efficiency, or it might result from luck, selection bias, or use of information not fully appreciated by the market. It is worth noting that in practice, markets are not perfectly efficient. Numerous anomalies and behavioral factors can create opportunities to find patterns from data that provide excessive returns. However, the persistence of anomalies remains a subject of ongoing debate, as recent empirical studies produce mixed evidence, suggesting that market efficiency can evolve over time in response to changing market conditions and the widespread dissemination of anomaly-related research (Schwert, 2003, 968).

Within the EMH framework, a DRL policy has zero expected risk-adjusted excess returns relative to an appropriate asset-pricing model, net of transaction and implementation costs. Beating a cap weighted benchmark can arise from risk exposures, rebalancing, or sampling variation and does not by itself violate efficiency. This reflects the joint-hypothesis problem which means that tests of efficiency are inseparable from the asset-pricing model used to measure excess returns.

Behavioral finance presents an alternative perspective to EMH, suggesting that markets are not always efficient due to psychological biases and cognitive limitations that affect investor behaviour. Barberis and Thaler (2003) provide a comprehensive overview of these biases, highlighting how overreactions, underreactions, and crowd behavior can lead to temporary market inefficiencies. This framework suggests that DRL models could, in theory, exploit these inefficiencies to achieve excess returns, especially in situations where investor behavior diverges from rational expectations.

In practice, DRL algorithms could potentially be tailored to detect and capitalize on behavioral biases, particularly in volatile markets. By doing so, DRL may challenge the EMH perspective, suggesting that, markets may have exploitable inefficiencies that advanced ML algorithms can adapt to.

The Adaptive Market Hypothesis (AMH), proposed by Lo (2004), integrates EMH with insights from behavioural finance, suggesting that market efficiency is not static but evolves based on environmental conditions and the experiences of market participants. In AMH, markets fluctuate between periods of efficiency and inefficiency as investors adapt to new information and competitive pressures.

In earlier days Black (1986) argued that prices are typically "efficient within a factor of two" because the noise supplied by uninformed traders makes markets liquid yet leaves persistent mis-pricings. This fluid definition of efficiency anticipated the Adaptive Market Hypothesis by recognising that rational and irrational forces coexist and vary over time. Two centuries of data analysed by Bouchaud et al. (2017) confirm Black's intuition: markets exhibit medium-term trending that eventually mean-reverts on multi-year horizons, a pattern the authors attribute to an adaptive tug-of-war between trend-following "chartists" and valuation-driven "fundamentalists." Together, these studies portray markets as noisy-efficient: liquid enough to provide abundant data, but systematically distorted enough to offer transient opportunities. Additionally, a recent multi-scale research by Safari and Schmidhuber (2025) indicates that efficiency is horizon dependent: prices usually mean-revert at very short (under 15 minutes) and very long (over 2 year) horizons, whilst indicating trend persistence from 30 minutes up to two years. This was measured by trend following premium, captured by linear trend coefficient β, that has declined steadily since the early 1990s. This indicates that competitive capital progressively arbitrages away patterns that were previously profitable.

This evolution allows for occasional inefficiencies that advanced models, like DRL, may exploit. Empirical evidence by Safari and Schmidhuber (2025), shows that the linear coefficient β flips sign outside the 30 minute to 2 year window, whilst the cubic term c alternates accordingly. This is a pattern that they interpret as evidence of how markets hover near a self organised critical point that peridodically shifts between efficient and inefficient regimes. Such cyclical behavior provides concrete empirical support for Lo's view and implies that adaptive algorithms can bring value by continually inferring where the market sits on the efficiency spectrum. However, in practice this means that trends in financial markets tend to reverse before coming statistically significant, which implies that when a trend is obvious on a price chart, it is typically already over which highlights the short time period learning cycles for trading models. For DRL, AMH provides a theoretical basis for developing adaptive strategies that respond dynamically to market shifts. This aligns well with DRL's strength in learning from and adapting to new patterns, suggesting that DRL could be particularly effective in markets that exhibit cyclical inefficiencies.

2.2 Portfolio theory

Modern Portfolio Theory (MPT), introduced by Markowitz (1952), provides the foundational quantitative framework for portfolio selection. MPT formalizes the concept of diversification: by holding a portfolio of assets, an investor can reduce unsystematic risk, also known as asset-specific risk, and achieve a better trade-off between risk and return. The key insight is that the risk of a portfolio, measured as variance or standard deviation

of returns, is not simply the weighted sum of individual volatilities, but also depends on the correlations between asset returns. Markowitz showed that for a given level of expected return, an investor should choose the portfolio with minimum variance. Conversely, for a given variance level, investor should choose the portfolio with maximum expected return. The set of such optimal portfolios is known as the efficient frontier.

In the classical Markowitz mean-variance framework the decision variable is the weight vector $w = (w_1, w_2, \ldots, w_N)$, whose elements specify the fraction of total wealth invested in each of the N available assets. Assuming the portfolio is fully invested with no leverage, the weights satisfy the single budget constraint $\sum_{i=1}^{N} w_i = 1$.

This can be framed as a quadratic optimization:

$$\begin{cases} \min\limits_{w} & w^T \Sigma\, w \\ \text{s.t.} & w^T \mu = \bar{R}, \\ & \sum\limits_{i} w_i = 1, \end{cases}$$

where μ is the vector of expected returns for each asset, Σ is the covariance matrix of asset returns, and $\bar{R}$ is a target expected return. An equivalent single-objective formulation uses a Lagrange multiplier $\lambda > 0$ to trade off expected return and risk:

$$\max_{w} \quad w^T \mu \; - \; \frac{\lambda}{2}\, w^T \Sigma\, w,$$

where λ controls the trade-off between return and risk. This framework yields a specific optimal portfolio for given inputs μ and Σ.

A DRL-based approach does not explicitly have to compute μ or Σ, but if the policy is conditioned to maximize the risk-adjusted return, then it should implicitly try to time and allocate assets to maximize returns and manage risk. In effect, the DRL agent could learn a dynamic portfolio strategy that might achieve a better risk-return trade-off than any static weights. One may view a well-trained DRL agent as trying to approximate the moving target of the optimal portfolio under changing market conditions. Traditional mean-variance optimisation can of course also be made dynamic by re-estimating μ and Σ on a rolling window and rebalancing periodically, yet each update still solves a static quadratic programme based only on first- and second-moment estimates from that window. By contrast, a DRL policy can react at every time step, exploit higher-order or path-dependent features, and adapt nonlinearly to structural breaks. This way any added value is therefore most likely to emerge in nonstationary, regime-shifting markets where window-based Markowitz reoptimizations may lag.

The 1/N strategy is a naive application of diversification where it ignores μ and Σ altogether, simply allocating equally. Surprisingly, DeMiguel et al. (2009) noted that 1/N often lies not too far from the efficient frontier in practice because estimation errors in μ and Σ can make optimized portfolios perform worse out-of-sample. For example, if expected returns are overestimated for some stocks, a mean-variance optimizer will overweight them, often leading to poor realized performance. The 1/N strategy avoids estimation altogether, thereby avoiding overfitting to historical data. DeMiguel et al. (2009) showed that only with an extremely long estimation window, with thousands of months of data for a moderate asset universe, can optimized portfolios statistically outperform 1/N portfolio.

The Capital Asset Pricing Model (CAPM), developed by Sharpe (1964) and others, builds on Modern Portfolio Theory (MPT) to provide an equilibrium model of asset prices. CAPM asserts that in equilibrium, investors will hold some combination of the risk-free asset and the market portfolio (the portfolio of all risky assets weighted by market value). CAPM's central equation is:

$$E[R_i] = R_f + \beta_i(E[R_m] - R_f),$$

where $E[R_i]$ is the expected return of asset i, R_f is the risk-free rate, $E[R_m]$ is the expected return of the market portfolio, and

$$\beta_i = \frac{\mathrm{Cov}(R_i, R_m)}{\mathrm{Var}(R_m)}.$$

Asset i's beta, β_i, is a measure of its systematic risk relative to the market.

This equation implies that an asset's excess return $E[R_i] - R_f$ is proportional to its beta, meaning that asset-specific risk is not rewarded because it can be diversified away. The market portfolio itself has $\beta = 1$ and provides the highest Sharpe ratio achievable by any combination of risky assets in the CAPM world. If CAPM holds, any portfolio's performance should be evaluated in terms of alpha, which means excess return beyond what its beta would predict. A consistently positive alpha would indicate skill or exploitation of market inefficiency.

For example, if a DRL strategy has a certain beta exposure to the market, part of its returns might just reflect being leveraged to the market or certain risk factors. We might find that a DRL agent is implicitly tilting the portfolio towards higher-beta stocks or certain sectors to chase returns, something that 1/N does not do, since 1/N is neutral in not favoring any stock beyond equal weighting. It will be examine whether DRL strategies yield excess returns unexplained by market risk (alpha).

CAPM is a single-factor model (the market). More modern approaches like the Fama-French multi-factor models incorporate more factors such as size, value, and momentum. If the DRL agent exploits something like momentum, one could argue it is harvesting a known factor rather than truly "inventing" a new strategy. In academic terms, beating the 1/N and even beating the market might simply mean the agent loaded on known factors without taking into account the potential confounders that might have influenced the outcome. For example, it might learn to overweight high-momentum stocks, effectively implementing a momentum strategy. This is not necessarily trivial, it would show the agent learned a sensible strategy but it would not violate market efficiency if those factors are known risk premia. Long-run mutual-fund evidence likewise shows that most apparent alpha disappears once returns are benchmarked against multi-factor models, implying that any “outperformance” is explained almost entirely by factor exposures and expenses, not genuine skill (Carhart, 1997, 57-58). On the other hand, if the agent’s performance cannot be explained by exposures to common factors or higher beta, it could indicate finding some niche inefficiency or superior timing ability.

In summary, classical finance provides normative models of how portfolios should be managed under certain assumptions (rational investors, efficient markets, multivariate normal returns, etc.). These models yield elegant solutions but often fall short in practice due to model mis-specification, estimation error, or inability to adapt to real-world complexities like transaction costs, changing distributions, and investor behavior biases. Data-driven machine learning approaches, including DRL, take a different path: rather than assuming a model for returns, they learn directly from data. This can potentially capture phenomena that static models miss. For example, a DRL agent could learn a dynamic allocation strategy that increases equity exposure in rising markets and shifts to defensive assets during downturns, a form of market timing that static MPT do not allow since those assume a fixed allocation to each asset or risk factor.

However, ML approaches are not guaranteed to find a truly optimal strategy. They require large amounts of data, and they risk overfitting to historical patterns that may not repeat. Moreover, they typically lack the clear theoretical guarantees of classical methods. For example, one might achieve a good Sharpe ratio in backtesting with a complex network, but it’s harder to prove why it should continue. This is where combining insights is valuable: for instance, incorporating ideas from finance (like risk aversion or transaction cost penalties) into the DRL reward function can guide the learning to more sensible solutions. Indeed, recent studies have tried to merge domain knowledge with DRL, such as using Modern Portfolio Theory within the reward design (Zhang et al., 2019) or momentum strategies (Wang et al., 2019) as a guide.

2.3 Previous studies

One of the early demonstrations of deep reinforcement learning in portfolio management was by Jiang et al. (2017), who proposed a framework called Ensemble of Identical Independent Evaluators (EIIE) for cryptocurrency trading. They employed an ensemble of deep neural networks as policy networks to allocate a portfolio over multiple cryptocurrencies. Their results indicated that RL-based portfolios outperformed several traditional strategies in backtests, even when accounting for substantial transaction costs. This was a striking result given the high volatility and noise in crypto markets, suggesting that the DRL agent could extract useful trading signals. Around the same time, another early practical implementation emerged by Almahdi and Yang (2017) employed a recurrent reinforcement learning agent to trade an equity portfolio with the Calmar ratio as the objective. They reported that on a risk-adjusted basis this approach outperformed mean-variance optimization. These early results helped catalyze an active research area at the intersection of finance and machine learning.

Following, Liu et al. (2018) extended the idea by applying DDPG to portfolio optimization. Their study showed remarkable performance improvement by using a continuous action RL method (DDPG) compared to prior policy gradient methods. This implied that allowing the agent to finely adjust portfolio weights (rather than picking discrete actions) and to learn from off-policy data can yield better trading performance. Their work provided evidence that DRL can handle the multi-asset allocation problem effectively and motivated the inclusion of DDPG in this comparative study.

Another notable line of research involves incorporating domain knowledge into DRL. For example, Wang et al. (2019) introduced AlphaStock, which combined a momentum-based strategy with deep RL. The RL agent in AlphaStock had two components: one focusing on buying recent winners and another on selling losers, echoing the “buy winners, sell losers (BWSL)" momentum strategy. By optimizing for the Sharpe ratio and using an attention-based neural network, AlphaStock’s agent was not a pure black-box but it was guided to exploit a known anomaly (momentum) in an intelligent way. The authors also performed a sensitivity analysis on stock features, finding that the learned strategy favored stocks with low volatility and high long-term growth, aligning with intuitive investment principles (avoid extremely volatile stocks, prefer fundamentally strong ones). This kind of result is encouraging, as it shows RL can rediscover sensible patterns and also highlights the importance of interpretability (they could identify what features were important to the agent’s decisions).

Ye et al. (2020) took a different approach by augmenting the state space of the RL agent with predictions of asset movements. They used an LSTM-based predictive model

to forecast short-term returns for each asset and fed these predictions as part of the RL agent's state (along with other market features). Their RL algorithm (based on a policy gradient method) could then use this enriched state information to allocate the portfolio. Essentially, this merges supervised learning (predictive modeling) with reinforcement learning (portfolio decision-making). The results in their research showed improved performance, suggesting that hybrid models that use external signals or models can enhance a pure RL approach.

Yang et al. (2020), proposed an ensemble strategy that dynamically switches between PPO, A2C, and DDPG based on market conditions. This is very relevant to this work because they literally considered the same algorithms this study is researching. In their approach, they trained separate agents with PPO, A2C, DDPG and then developed a meta-agent that looks at market volatility/regime indicators to decide which agent's actions to follow at a given time. They reported that this ensemble preserved robustness across different market scenarios. For instance, in stable market periods a value-based or deterministic strategy might do well, whereas in highly volatile times a more exploratory strategy might cope better. Their approach successfully underscores that no single algorithm may be universally best but each has strengths under certain conditions. Their ensemble achieved better overall performance than any single algorithm by essentially performing an algorithm selection based on regimes. This provides an interesting perspective: rather than seeking one champion algorithm, combining them could yield more consistent results.

Beyond equity portfolios, similar DRL methods have been applied to other financial problems. For example studies like Nevmyvaka et al. (2006) and Beysolow II (2019) applied RL for market making and execution, where RL algorithms (including DDPG variants) were used to optimize order placement in limit order books. For asset allocation with different assets, some studies looked at portfolios including bonds, commodities, etc. For example, a method called DeepPocket by Soleymani and Paquet (2021), represented the portfolio as a graph of assets to capture relationships, and used a graph convolutional network with RL to manage a multi-asset portfolio. Such approaches show the flexibility of deep RL in handling complex relationships.

Beyond standard single-agent RL, researchers have started to explore multi-agent and meta-learning approaches. Lee et al. (2020) introduced a multi-agent reinforcement learning system called MAPS (Multi-Agent Portfolio Management System), in which multiple agents each learn distinct portfolio strategies and collectively form an ensemble portfolio. The agents are trained with a diversity seeking objective so that each specializes in different market conditions, thereby achieving a more robust overall performance through diversification. Meanwhile, meta-reinforcement learning (meta-RL) is being examined as

a way to handle non-stationarity. The concept is to train an agent that can quickly adapt to new market regimes by learning how to learn. For instance, recent work by Tian et al. (2024) applies model agnostic meta-learning (MAML) to trading, pairing a PPO meta-learner with a fast adaptive learner, and shows improved performance in fast-changing markets. Though still nascent, meta-RL could allow an agent to generalize knowledge from one market or period (say, a bull market) to another (a sudden crash) with minimal additional training which is a valuable trait given the covariate shifts in finance.

Survey by Bai et al. (2024) reviewed RL in finance, stating that while many papers claim positive results, the field faces issues like lack of standardized benchmarks, difficulty in reproducing results, and insufficient testing of robustness. That said, there has been positive moves, for example Liang et al. (2018) published a GitHub repository for their work, and others like Liu et al. (2018) also open sourced their trading agent code. Even with code, hyperparameter tuning is still a weakness. RL algorithms have many settings (learning rate, exploration noise, etc.) and finance provides no clear solution to tune against except final profit, which is a high variance metric. Liang et al. (2018) specifically experimented with different learning rates and network structures and found that some algorithms (like DDPG) were very sensitive to tuning, sometimes getting stuck in local optimum. This suggests RL needs carefully calibrated hyperparameters to perform well on financial problems, which is a drawback compared to more straightforward methods. Table 1 below contains a summary of recent studies in the field.

Study (Year)	**RL Algorithm(s)**	**Market / Context**
Moody et al. (1998)	PG (RRL)	Single asset
Almahdi and Yang (2017)	PG (RRL)	Multi-asset equities
Jiang et al. (2017)	AC (DDPG)	Cryptocurrency
Jiang and Liang (2017)	PG (direct)	Cryptocurrency
Pendharkar and Cusatis (2018)	VB (Q-learning)	Equity indices
Liang et al. (2018)	AC (DDPG, PPO)	China A-shares
Xiong et al. (2018)	AC (DDPG var.)	U.S. equities
Buehler et al. (2019)	AC (Deep Hedging)	Options hedging
Jeong and Kim (2019)	VB (DQN, transfer)	U.S. single stocks, daily
Ye et al. (2020)	AC (PPO, DDPG/PG)	"HighTech" portfolio
Lee et al. (2020)	Multi-agent (PPO/agent)	U.S. equities
Huang et al. (2020)	AC (DDPG/PPO)	China A-shares
Park et al. (2020)	VB (DQN)	Korean equities
Wang et al. (2021)	PG+PPO (two-unit)	Global stock indices
Wu et al. (2021)	PG (CNN/RNN policy)	U.S. stocks & sector ETFs
Betancourt and Chen (2021)	AC (PPO, RNN policy)	Cryptocurrency
Wu et al. (2021)	AC (CNN/RNN policies)	Taiwan 50 and S&P 500

Table 1: Summary of representative studies on reinforcement learning in portfolio management. Abbrev: PG = Policy Gradient, AC = Actor–Critic, VB = Value-based.

3 MACHINE LEARNING FRAMEWORK

3.1 Neural Networks and the Backpropagation Algorithm

Deep learning is a subfield of machine learning that uses artificial neural networks with multiple processing layers to learn data representations at increasing levels of abstraction. In a deep neural network, each layer transforms its input (the outputs of the previous layer) into a more abstract representation, enabling the model to capture complex patterns in data. The concept of training such networks dates back several decades, but it was the reintroduction of effective backpropagation combined with large datasets and computing power that led to recent breakthroughs in speech recognition, computer vision, natural language processing, and many other domains. The process described in this section forms the foundation of modern deep learning methods and closely follows the work of Lecun et al. (2015).

The backpropagation algorithm is used to efficiently train these deep models by computing how the network's parameters (weights and biases) should be adjusted to minimize errors. Formally, backpropagation applies the chain rule of calculus to propagate the gradient of a loss function (which measures prediction error) from the output layer back through the hidden layers, accumulating partial derivatives. This allows gradient-based optimizers such as gradient descent to incrementally update the network's weights and improve performance. In practice, variants like stochastic gradient descent (SGD) are used, where network parameters θ are updated iteratively:

$$\theta \leftarrow \theta - \eta \nabla_\theta E(\theta),$$

where η is the learning rate and E the loss function. Through iterative adjustments, the network learns to approximate the complex function that maps inputs to outputs, often achieving high accuracy on training data. Importantly, deep neural networks (those with many hidden layers) can model extremely nonlinear relationships, given sufficient data and computational resources.

Figure 1 illustrates a fully connected feed-forward network with two hidden layers H1 and H2 and one output layer. Let the input be $x = (x_1, \ldots, x_n)$. For any unit u, denoted by z_u its pre-activation (the affine input before the nonlinearity) and by y_u its activation (output). Using indexing as w_{ab} is the weight from source unit a to destination unit b (e.g. w_{ij} connects input i to hidden unit j). Set $y_i = x_i$ for input units $i \in$ Input.

The forward pass is then

$$z_j = \sum_{i \in \text{Input}} w_{ij}\, y_i + b_j, \qquad y_j = f(z_j) \quad (j \in H1),$$

$$z_k = \sum_{j \in H1} w_{jk}\, y_j + b_k, \qquad y_k = f(z_k) \quad (k \in H2),$$

$$z_l = \sum_{k \in H2} w_{kl}\, y_k + b_l, \qquad y_l = f(z_l) \quad (l \in \text{Output}).$$

Here $f(\cdot)$ is the activation function. Inputs propagate as activations y to the next layer, each unit forms a pre-activation z via a weighted sum plus bias, and the nonlinearity produces the next activations.

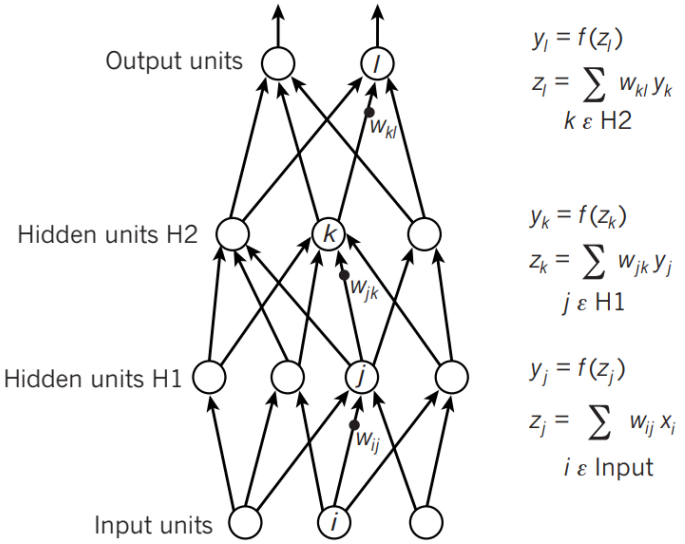

Figure 1: A multilayer neural network with two hidden layers and one output layer. (Lecun et al., 2015)

The activation function introduces non-linearity, which is crucial. Without it, a stack of linear neurons would collapse into an equivalent single linear model. In modern deep networks, the most commonly used activation functions are the logistic sigmoid, the hyperbolic tangent (tanh), the Rectified Linear Unit (ReLU), and (for multi-class outputs) the softmax.

In binary classification, the logistic sigmoid is defined by

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

It maps real inputs to the open interval $(0, 1)$, enabling a probabilistic interpretation.

A downside is saturation near 0 and 1, where the derivative becomes small and learning can suffer from vanishing gradients.

The hyperbolic tangent,

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

takes values in $(-1, 1)$ and is zero-centered, which typically yields more balanced updates around the origin and in practice often stronger gradients than the sigmoid.

The Rectified Linear Unit (ReLU) is defined piecewise as

$$\mathrm{ReLU}(x) = \begin{cases} 0, & x < 0, \\ x, & x \geq 0, \end{cases}$$

where negative inputs are set to zero while positive inputs pass through linearly. ReLU is computationally efficient and helps mitigate vanishing gradients (Glorot et al., 2011, p. 318).

For multi-class outputs, the softmax function converts a vector $x \in R^K$ into a categorical distribution:

$$\mathrm{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, \qquad i = 1, \ldots, K.$$

The resulting components are non-negative and sum to one, providing a probabilistic interpretation over K classes. Figure 2 visualizes these activation functions.

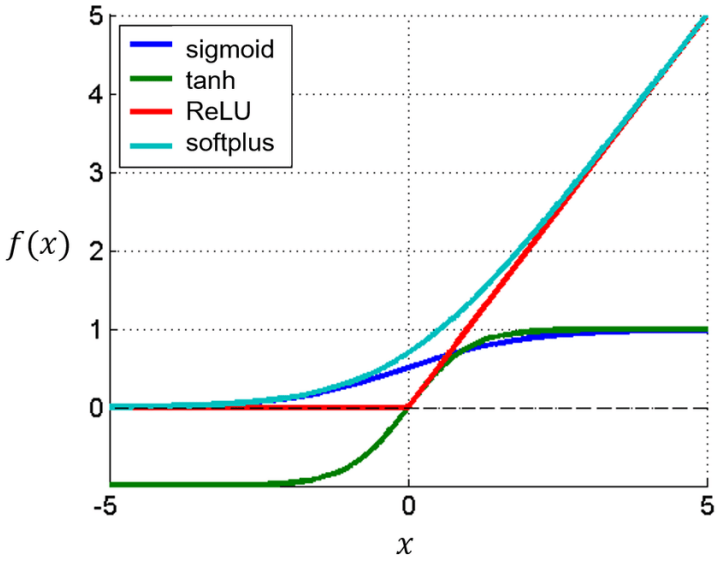

Figure 2: Different activation functions. (Musiol, 2016).

Once the hidden layer activations are computed, they are propagated forward to the next layer. The same process of weighted sums and nonlinear activation continues until the final layer (the output layer), which provides the network's prediction. In classification tasks, for instance, the output layer often uses a softmax or sigmoid function, while in regression tasks, the output can be linear or another suitable function.

Training the network consists of finding the set of weights $\{w_{ij}, w_{jk}, \ldots\}$ that minimizes a cost function E, which measures the discrepancy between the network's prediction and the target. The key to performing this minimization is the backpropagation algorithm, which relies on the chain rule of derivatives (Figure 3).

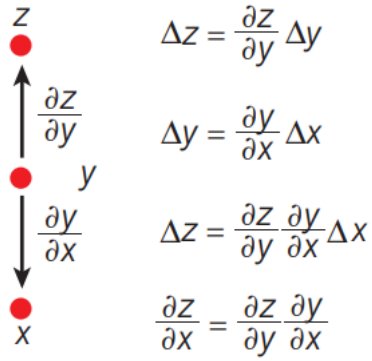

Figure 3: An illustrative example of the chain rule. (Lecun et al., 2015).

If x affects y according to $\frac{\partial y}{\partial x}$, and y affects z according to $\frac{\partial z}{\partial y}$, then a small change Δx in x induces a change $\Delta z = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta x$. This principle naturally extends to multiple variables and is crucial in deriving the backpropagation equations.

After the forward pass, the error at the output layer is computed by comparing the network's output y_l with the target t_l. A common choice is the mean squared error $E = \frac{1}{2}\sum_l (y_l - t_l)^2$, whose derivative with respect to y_l is simply $\frac{\partial E}{\partial y_l} = y_l - t_l$. Using the chain rule again, we obtain the error derivative with respect to the net input z_l: $\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l} \cdot \frac{\partial y_l}{\partial z_l}$.

For a hidden unit k with pre-activation $z_k = \sum_j w_{jk} y_j + b_k$ and activation $y_k = f(z_k)$, the error with respect to its output is obtained by summing the downstream error signals weighted by the outgoing weights to all units l in the next layer:

$$\frac{\partial E}{\partial y_k} = \sum_l w_{kl} \frac{\partial E}{\partial z_l},$$

since $z_l = \sum w_{ul} y_u + b_l$ implies $\partial z_l / \partial y_k = w_{kl}$. This is then converted to the pre-activation derivative via the activation slope,

$$\frac{\partial E}{\partial z_k} = f'(z_k) \frac{\partial E}{\partial y_k}.$$

Finally, the gradient of a weight from neuron j (previous layer) to neuron k (current layer) is

$$\frac{\partial E}{\partial w_{jk}} = y_j \frac{\partial E}{\partial z_k}.$$

Figure 4 shows a diagram of how these derivatives move backward in the network.

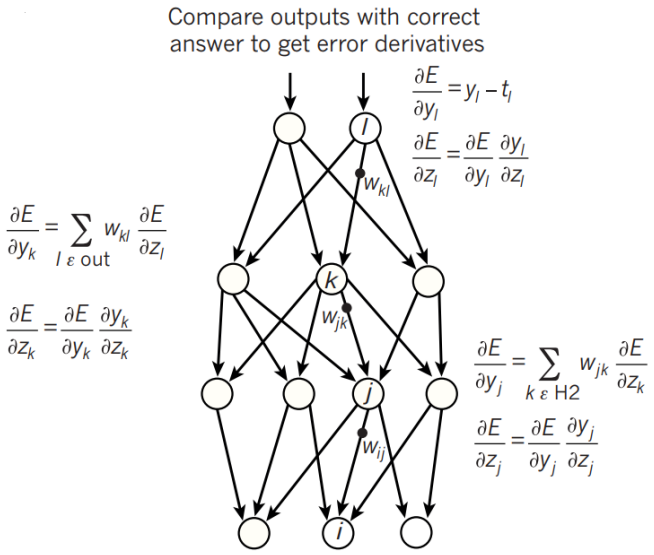

Figure 4: Schematic diagram showing the backward pass. (Lecun et al., 2015).

3.2 Reinforcement Learning

Sutton et al. (1998) introduced reinforcement learning (RL) as a paradigm of machine learning in which an agent learns to make sequential decisions by interacting with an environment, with the goal of maximizing cumulative reward. In contrast to supervised learning, where the correct actions are provided for each example, an RL agent must discover good strategies through trial and error. Sutton et al. (1998) famously defined reinforcement learning as "learning what to do - how to map situations to actions - so as to maximize a numerical reward signal." In this framework, the agent is not explicitly told which action to take at any state but instead it receives feedback in the form of rewards and must learn from experience. Over time, through repeated interaction, the agent improves its policy (decision making strategy) to achieve higher accumulated rewards.

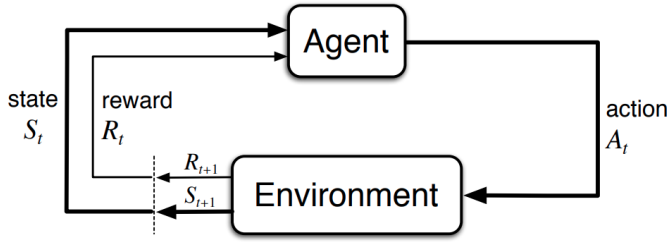

Figure 5: The agent-environment interaction in reinforcement learning. (Sutton et al., 2018, 48).

According to Sutton et al. (1998), the standard formalization can be presented as a Markov Decision Process (MDP). This can be defined by a 4-tuple (S, A, P, R), where S is the set of states, A the set of actions, $P(s' \mid s, a)$ the transition kernel, and $R(s, a)$ the immediate reward. Additionally, to ensure that the total reward sum converges, a discount factor $0 \leq \gamma \leq 1$ that weights future rewards, is used here. At each time step t, the agent observes state $s_t \in S$, chooses an action $a_t \in A$ according to some policy $\pi(a_t \mid s_t)$, and the environment transitions to a new state s_{t+1} and provides a reward according to function $R(s_t, a_t)$.

The value of a state under policy π is

$$V_\pi(s) = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t \, R(s_t, a_t) \,\middle|\, S_0 = s\right],$$

and the state-action value is

$$Q_\pi(s, a) = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t \, R(s_t, a_t) \,\middle|\, S_0 = s, \; A_0 = a\right].$$

These satisfy the Bellman expectation equations:

$$V_\pi(s) = \sum_a \pi(a \mid s)\left[R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \, V_\pi(s')\right],$$

$$Q_\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \sum_{a'} \pi(a' \mid s') \, Q_\pi(s', a').$$

The optimality equations are

$$V^*(s) = \max_a \left[R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \, V^*(s')\right],$$

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

and the optimal policy chooses

$$\pi^*(s) = \arg\max_a Q^*(s, a).$$

In most real-world problems, including portfolio management, the state and action spaces are large (possibly continuous), and the environment dynamics are unknown. The Bellman equations cannot be solved exactly, so instead approximate values or policies are used and iteratively improved which is the essence of reinforcement learning. In the portfolio application, the control problem is cast in the MDP framework. Let $G_{t+1} = 1 + R_{t+1}$ denote next period gross and the portfolio weights at time t to be denoted by $w_t = (w_{t,1}, \ldots, w_{t,n})$, $w_{t,i} \geq 0$ and $\sum_{i=1}^n w_{t,i} = 1$. Wealth evolves according to

$$W_{t+1} = W_t\,(w_t^\top G_{t+1}), \qquad W_0 > 0.$$

Then the portfolio-specific state to be $s_t = (X_t, w_{t-1})$, where X_t collects predictive features/market signals, and the action to be $a_t = w_t$. A natural per-period reward for logarithmic growth with trading frictions $C(w_t, w_{t-1}) \geq 0$ is

$$r_t \;=\; \log(w_t^\top G_{t+1}) \;-\; C(w_t, w_{t-1}).$$

Over a finite horizon T (or in episodic tasks with $\gamma = 1$),

$$\log W_T = \log W_0 + \sum_{t=0}^{T-1} \log(w_t^\top G_{t+1}) \;\;\Rightarrow\;\; E\left[\sum_{t=0}^{T-1} r_t\right] = E[\log W_T] \;-\; E\left[\sum_{t=0}^{T-1} C(w_t, w_{t-1})\right],$$

so maximizing the expected cumulative reward coincides with maximizing expected terminal log-wealth net of costs, aligning the RL objective with the classical log optimal criterion.

When trading costs are set to zero ($C = 0$), there is no market impact (actions do not affect future return distributions or features), and $X_{t+1} \sim P(\cdot \mid X_t)$ is action-independent, Bellman's recursion decomposes and the optimal decision becomes myopic:

$$w_t^\star \;\in\; \arg\max_{w \in \Delta^n} \; E\big[\log(w^\top G_{t+1}) \,|\, X_t\big]\,.$$

In other words, the long-horizon "coupling" between actions is absent by construction because current actions do not influence future states or costs and the problem reduces to a contextual bandit. A special case is i.i.d. returns with no informative features, which yields a time-invariant Kelly portfolio

$$w^\star \;\in\; \arg\max_{w \in \Delta^n} \; E\big[\log(w^\top G)\big]\,.$$

Intertemporal coupling reappears once one introduces trading costs $C(w_t, w_{t-1}) > 0$, action-dependent dynamics/market impact, or path-dependent constraints (e.g., leverage, turnover, drawdown, taxes). In those cases the problem is genuinely dynamic and long-horizon credit assignment becomes essential.

The objective is to find a policy that maximizes expected discounted return,

$$J(\pi) \;=\; E_\pi\left[\sum_{t=0}^{T-1} \gamma^t \, r_t\right],$$

with T finite or infinite. The policy-gradient theorem provides a direct optimization route with:

$$\nabla_\theta J(\pi_\theta) \;=\; E\Big[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\; A_{\pi_\theta}(s_t, a_t)\Big],$$

where $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$ is the advantage. A common estimator is Generalized Advantage Estimation (GAE) (Schulman et al., 2015, 5), which trades bias and variance via $\lambda \in [0,1]$:

$$\delta_t \;=\; r_t + \gamma\, V_\phi(s_{t+1}) - V_\phi(s_t), \qquad \widehat{A}_t^{(\lambda)} \;=\; \sum_{l=0}^{\infty} (\gamma\lambda)^l \, \delta_{t+l}.$$

In Deep RL, neural networks are used as function approximators for the policy and/or value functions. The neural network parameters serve as θ (for policy) and ϕ (for the value function), which are optimized via stochastic gradient descent on appropriate loss functions. For instance, a policy network might output a probability distribution over actions or parameters of a distribution (e.g. mean and variance for continuous actions), and then adjust its weights to increase the probability of actions that lead to higher returns. The policy of a Markov Decision Process can be optimised with a wide range of reinforcement learning algorithms. Numerous methods have been proposed, each excelling under different trade-offs in data quality, sample efficiency, and computational cost. At the highest level under deep reinforcement learning these algorithms are usually divided into model-based and model-free approaches

Model-based RL learns (or is given) an explicit model of the environment's transition dynamics and reward function. The policy is then improved by planning, which refers to a computational process that searches an optimal path from the state space (Sutton et al., 2018, 160-163). In the model-free setting an agent foregoes any explicit model of the environment's dynamics and instead shapes its behaviour solely through sampled transitions $(S_t, A_t, R_{t+1}, S_{t+1})$. The most widely used learning signal is Temporal-Difference (TD) error, where value estimates are forwarded toward a bootstrap target that already contains the current prediction, thus blending the low variance of one-step look-ahead

with the long-range credit assignment of simulation returns.

Where TD methods treat the policy as an implicit consequence of a value function, policy-gradient theory poses control directly as stochastic optimisation in parameter space. These policy-based methods directly learn the policy $\pi(a \mid s; \theta)$ parameterized by θ (the weights of a neural network). These methods adjust θ to maximize $J(\pi)$ using gradient ascent. A famous example is the REINFORCE algorithm by Williams (1992). Although unbiased, this estimator suffers from high variance. Subtracting a learned baseline greatly reduces that variance without altering the expectation, motivating the actor-critic architecture in which a critic trained by TD stabilises the policy updates of an actor (Sutton et al., 2018, pp. 325–332). These actor-critic methods combine both, an actor (policy) that decides actions and a critic (value function) that critiques them. Deep reinforcement learning became widely practical when neural networks were paired with policy-gradient algorithms that stabilise assignments through the use of an auxiliary critic. The critic helps reduce variance in policy gradient updates by providing an estimate of how good an action was compared to an average baseline. Another key approach is value-based methods that learn an estimate of $Q^*(s, a)$ (or $V^*(s)$) and derive the policy from it. Classic examples: Q-learning (Watkins and Dayan, 1992) and its deep neural network variant Deep Q-Network (DQN) by (Mnih et al., 2016).

PPO, DDPG, and A2C were chosen as they are among the most prominent DRL methods, each representing a different approach to policy learning. Each algorithm has been used in prior finance research. For instance, A2C/A3C has been applied in trading scenarios for its simplicity and parallel training capabilities, DDPG has been employed to optimize trading strategies with continuous position sizes, and PPO has gained popularity in many domains due to its reliability, including ensemble approaches for stock trading.

3.3 DRL Algorithms

3.3.1 Advantage Actor-Critic

Advantage Actor–Critic (A2C) is a policy-gradient method that trains two networks together and combines their results to learn effectively. These networks include an actor that tries to estimate what should be done next, and a critic that evaluates states as how good is the current situation. A2C may be viewed as the synchronous counterpart of the earlier asynchronous A3C algorithm by Mnih et al. (2016), operating within a single process rather than relying on parallel workers. This synchronization simplifies implementation and often makes training more reproducible without changing the underlying learning signals.

Formally, the actor defines a stochastic policy $\pi_\theta(a \mid s)$ that maps a state s to a distribution over actions, whereas the critic approximates the state-value function $V_\phi(s)$, which is the expected discounted return available from s when following the current policy. Using the critic's value estimate as a baseline, A2C centers the policy-gradient update on the advantage, which measures how much better or worse a chosen action performed compared with the policy's average expectation in that state. The advantage at time t is

$$\hat{A}_t = \hat{R}_t - V_\phi(s_t),$$

where $\hat{R}_t$ in A2C is an n-step bootstrap return,

$$\hat{R}_t = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V_\phi(s_{t+n}),$$

with discount factor $\gamma \in (0, 1]$. Intuitively, $\hat{A}_t > 0$ signals that the taken action was better than expected, and $\hat{A}_t < 0$ that it was worse. Using advantages reduces the variance of gradient estimates while keeping them unbiased, which typically yields faster and more stable learning than a plain policy gradient.

The actor is updated to increase the likelihood of advantageous actions and decrease the likelihood of disadvantageous ones. The canonical policy-gradient direction is

$$\nabla_\theta J(\theta) = E_t\Big[\, \hat{A}_t \, \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Big] .$$

In parallel, the critic is trained as a regressor to fit the bootstrap returns, by minimizing a squared-error loss,

$$L_{\text{value}}(\phi) = \Big(\hat{R}_t - V_\phi(s_t)\Big)^2 .$$

To avoid premature collapse of the policy's exploration, A2C commonly adds an entropy bonus that rewards higher-entropy action distributions. This is implemented as a regularizer in the loss:

$$L_{\text{entropy}} = -\beta \, E_t[H(\pi_\theta(\cdot \mid s_t))] ,$$

where H is the Shannon entropy and $\beta > 0$ controls the strength of exploration.

These components are combined into a single objective optimized by stochastic gradient methods. Writing the training criterion as a loss to be minimized,

$$L(\theta, \phi) = -\, E_t\Big[\hat{A}_t \log \pi_\theta(a_t \mid s_t)\Big] + c_1 \, E_t\Big[\Big(\hat{R}_t - V_\phi(s_t)\Big)^2\Big] - c_2 \, E_t[H(\pi_\theta(\cdot \mid s_t))] ,$$

with weights $c_1, c_2 \geq 0$ tuning the relative importance of value-function accuracy and exploration. A typical training iteration proceeds by rolling out the current policy to

collect short trajectories (s_t, a_t, r_t, s_{t+1}), computing n-step returns and advantages, and then applying gradient steps that simultaneously push the actor toward actions with positive advantages and refine the critic to make value estimates better predictors of future returns. This synchronized loop is repeated until the policy stabilizes. Pseudocode provided in Appendix 1.

A2C benefits lie in its balance of bias and variance. The critic's baseline reduces the variance of policy-gradient estimates without introducing bias, the multi-step bootstrap returns trade off short-horizon noise and long-horizon bias, and the entropy bonus sustains exploration early in training. In practice, these ingredients yield a simple, efficient algorithm that often learns faster and more stably than vanilla policy gradients while avoiding the engineering overhead of asynchronous methods.

3.3.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO), proposed by Schulman et al. (2017), is an on-policy policy gradient method that achieves stable and reliable training by not allowing large updates to the policy at once. Like Advantage Actor–Critic (A2C), it trains an actor to select actions and a critic to estimate values, but its hallmark is how it constrains each policy update so the new policy does not drift too far from the old one in a single step. This addresses a well-known failure mode of vanilla policy gradients, where overly large gradient steps can push the policy into regions with poor performance and high variance. PPO can be seen as a simplified version of the earlier Trust Region Policy Optimization (TRPO) method, which enforced a hard constraint on the change in policy per update. PPO instead uses a soft constraint by clipping the policy change.

PPO maximizes expected return but augments it with a safety mechanism that prevents overly large policy changes. This is done by comparing the new policy π_θ to the old policy $\pi_{\theta_{\text{old}}}$ via the probability ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}.$$

If $r_t(\theta) > 1$, the new policy assigns higher probability to the sampled action than before and if $r_t(\theta) < 1$, it assigns less. As in A2C, updates are guided by an advantage estimate $\hat{A}_t$, which scores how much better (positive) or worse (negative) the sampled action was relative to the state's baseline value.

The PPO surrogate objective maximized in practice is the clipped objective

$$L_{\text{CLIP}}(\theta) = E_t\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta),\, 1-\epsilon,\, 1+\epsilon)\ \hat{A}_t\Big)\Big].$$

The clip$(\cdot)$ operator cuts the ratio to the interval $[1-\epsilon,\, 1+\epsilon]$ for a small ϵ (e.g., 0.1–0.2).

Intuitively, when $\hat{A}_t > 0$ the objective refuses to reward increases in r_t beyond $1 + \epsilon$ and when $\hat{A}_t < 0$ it refuses to reward decreases beyond $1 - \epsilon$. In both cases the "min" selects the pessimistic (clipped) improvement if the un-clipped term would push the policy too far. This acts as a soft trust region: it allows helpful, local updates while discouraging destructive jumps.

As with actor-critic methods more broadly, PPO augments the policy objective with a value-function loss and an entropy bonus. Writing the overall training target as a maximization problem,

$$L_{\text{PPO}}(\theta, \phi) = E_t\Big[L_{\text{CLIP}}(\theta) - c_1 \left(\hat{R}_t - V_\phi(s_t)\right)^2 + c_2 \, H(\pi_\theta(\cdot \mid s_t))\Big],$$

where V_ϕ is the critic, $\hat{R}_t$ is a bootstrap estimate of return, H is the Shannon entropy of the policy, and $c_1, c_2 \geq 0$ weight value accuracy and exploration. In code this is implemented as a loss to minimize by negating the objectives first and last terms and leaving the squared-error term positive.

Training proceeds in short on-policy batches. The algorithm freezes θ_{old}, rolls out the current policy for several episodes (an episode is a complete run in the environment from an initial state to a terminal state) and computes advantages frequently using Generalized Advantage Estimation (GAE). With the batch fixed, PPO then performs multiple epochs of stochastic gradient ascent on L_{PPO} using minibatches, standardizing advantage to stabilize the scale of updates. Pseudocode provided in Appendix 3.

Compared with A2C, PPO differs in two practical ways. First, it replaces the plain advantage-weighted policy gradient with the clipped surrogate, which explicitly prevents large per-sample likelihood changes. Second, it reuses each on-policy batch for several optimization epochs, substantially improving sample efficiency relative to a single update per batch. These two design choices translate to greater stability and competitive performance while keeping the implementation as simple as a standard first-order optimizer.

PPO is known for its stability and ease of use. It generally requires fewer parameter tweaks than DDPG and avoids the complexity of a replay buffer. Being on-policy, it does require more environment interactions to learn effectively, but with modern compute and using parallel simulation, this is manageable. In the finance context, on-policy means that it is always using the latest policy to generate new trajectories. One potential issue is that financial time series are not easily resettable to random states (like game states), instead it is often trained on one long chronological sequence. To apply PPO, one should split the historical data into multiple segments (or use multiple parallel markets or time periods) to simulate multiple episodes. Alternatively, treat each day as a continuing

episode with for example random start years. PPO has been successfully used in various financial studies (sometimes combined with other models). For example, Yang et al. (2020) include PPO in an ensemble strategy for stock trading and found it helpful for adapting to different market conditions. PPO's clipping mechanism also makes it easier to incorporate additional objectives (like risk penalties) without destabilizing the training too much. In this work I will implement a version of PPO with a multi-output neural network (one output for the action distribution over assets, another for the value estimate) often called an actor-critic PPO.

3.3.3 DDPG

Deep Deterministic Policy Gradient (DDPG) by Lillicrap et al. (2015) is an off-policy actor-critic algorithm tailored to continuous action spaces where it learns a Q-function by using the Bellman equation and a policy by using the Q-function. It expands the idea of deterministic policy gradient (DPG) framework from (Silver et al., 2014) by adding value-based stabilization techniques from deep Q-learning (Mnih et al., 2015). Core idea of DDPG is an actor network $\mu_\theta(s)$ that deterministically maps state to a specific action. In continuous action spaces, this is more efficient than learning a probability distribution over actions. A critic network $Q_\phi(s, a)$ estimates the value of state-action pairs.

DDPG also keeps target networks $\mu_{\theta'}$ and $Q_{\phi'}$, which are copies of the actor and critic that lag behind, updated slowly via:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta', \quad \phi' \leftarrow \tau\phi + (1-\tau)\phi', \quad \text{with } \tau \ll 1$$

These targets are used for computing stable target Q-values. This lagged tracking prevents destructive feedback loops in which rapidly changing targets destabilize the critic, which in turn misguides the actor.

Because DDPG is off-policy, it decouples data collection from learning. Transitions (s_t, a_t, r_t, s_{t+1}) are stored in an experience replay buffer, from which the algorithm draws random minibatches for updates. Replay reduces the temporal correlations present in sequential data and allows each sample to be reused across many gradient steps, improving sample efficiency and smoothing the learning signal.

The critic is trained by minimizing a temporal-difference (TD) regression loss toward a bootstrap target computed with the target networks:

$$y_t = r_t + \gamma\, Q_{\phi'}(s_{t+1},\, \mu_{\theta'}(s_{t+1}))\,,$$

$$L(\phi) = \frac{1}{N} \sum_t \left(Q_\phi(s_t, a_t) - y_t \right)^2 .$$

Here $\gamma \in (0, 1)$ is the discount factor. The target y_t uses a slowly moving critic and actor, which makes the supervised signal for the online critic less volatile and reduces the risk of divergence.

The actor is optimized to choose actions that the current critic deems valuable. Using the deterministic policy-gradient theorem, the update direction follows the critic's action-gradient:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_t \left[\nabla_a Q_\phi(s_t, a)|_{a=\mu_\theta(s_t)} \nabla_\theta \mu_\theta(s_t) \right] .$$

Intuitively, the critic supplies a local slope in action space to nudge the action to improve long-run return while the actor backpropagates this signal through its own parameters to make such actions more likely in similar states.

Because the policy is deterministic, exploration must be injected explicitly at action time. DDPG disturbs the actor's output with noise:

$$a_t = \mu_\theta(s_t) + N_t.$$

(Lillicrap et al., 2015) chose Ornstein-Uhlenbeck process noise by Uhlenbeck and Ornstein (1930) for temporally correlated exploration in physical control tasks but uncorrelated Gaussian noise is a common alternative.

A typical training iteration proceeds as follows. Agent interacts with the environment using the noisy policy, appending each transition to the replay buffer. Periodically, it samples a minibatch from the buffer, updates the critic by minimizing the TD loss toward the target networks' bootstrap, updates the actor using the critic's action-gradients, and then softly updates the targets toward the new online parameters. This loop improves the quality of value estimates and the policy they guide. Pseudocode in Appendix 2.

DDPG's off-policy nature allows efficient reuse of past experience, especially beneficial in finance where historical data is limited. It handles continuous actions such as portfolio weights. However, off-policy methods like DDPG can be fragile regarding exploration or function approximator instability. It might get stuck in suboptimal policies due to insufficient exploration or overestimate Q-values. Improvements like TD3 by Fujimoto et al. (2018) have addressed some issues via twin critics. DDPG's continuous action formulation is particularly attractive in finance, for instance Liu et al. (2018) successfully applied DDPG to stock trading, showing improved performance over simpler policy gradient methods. Portfolio weightsare constrained with normalization to ensure valid portfolio configurations (sum to 1, non-negative).

4 METHODOLOGY

4.1 Data description

A compact yet realistic selection of U.S. large-cap equities was selected so that the RL agents (and the 1/N benchmark) can allocate across liquid, sector-diverse assets. Chosen stocks have high liquidity to ensure that trading signals from the DRL agent are viable in practice meaning that there are no issues filling orders and also to minimize microstructure noise in price data. Liquidity is typically measured by metrics such as average daily trading volume or market capitalization. In this research the focus is on large-cap stocks which trade millions of shares per day. Concretely, the construction starts from the S&P 500 contenders and then the chosen stocks are picked from the most traded among them that fulfill other criteria. High liquidity also means fewer missing trading days and more reliable pricing data.

To ensure a diversified portfolio, stocks are included from a variety of sectors, for example technology, finance, healthcare, consumer, and industrials. This prevents the portfolio from being too concentrated in a single industry, which could bias results and also increases collinearity of returns. By having a mix of sectors, the DRL agent will have to learn to allocate among different industries, which often respond differently to market conditions, for instance tech and utilities. This tests the agent's ability to rotate between sectors if advantageous (a form of dynamic allocation that can add value if it predicts sectoral trends). Stocks included with a range of volatility profiles, some more stable with lower volatility e.g. utilities or large consumer staples, and some more volatile e.g. tech or smaller caps. This ensures the agent deals with different risk profiles. High volatility stocks present more opportunity and risk for the agent to exploit. Low volatility stocks provide stability and are often favored in risk-adjusted strategies. By mixing them, the agent might learn to overweight volatile stocks when confident and shift to stable ones in uncertain times, similar to a market volatility timing strategy. However, to keep things reasonable, extremely volatile or distressed stocks are avoided, because they could skew results or present irregular data patterns due to corporate actions.

Applying these criteria a portfolio of 10 is constructed. Ten is a common number in academic studies to keep state-space manageable for RL, for example Jiang et al. (2017) used 11 including cash. A cash asset is sometimes included (or money market fund proxy) as one of the "assets" in the portfolio. This would allow the agent to deallocate from equities into cash if it predicts a broad downturn. The 1/N strategy including cash would mean 1/(N+1) in each stock and cash, however, often 1/N refers to fully invested in equities. For simplicity and alignment with many studies, I proceed with fully invested

portfolios with no cash. The agent must always distribute 100 percent among the stocks. This also simplifies the action space constraint. The chosen stocks are listed in Table 2.

Table 2: Final stock universe by sector and style

Company	Sector	Formal Style
Apple (AAPL)	Information Technology	Sensitive
Nvidia (NVDA)	Semiconductors / IT	Sensitive
Google (GOOGL)	Communication Services	Sensitive
JPMorgan Chase (JPM)	Financials	Cyclical
Johnson & Johnson (JNJ)	Health Care	Defensive
Walmart (WMT)	Consumer Staples	Defensive
Coca-Cola (KO)	Consumer Staples (Beverages)	Defensive
McDonald's (MCD)	Consumer Discretionary	Cyclical
NextEra Energy (NEE)	Utilities	Defensive
Lockheed Martin (LMT)	Industrials / Defense	Sensitive

This set ensures each asset is well traded. Each stock in the list had no major data issues, like a merger causing a disappearance in time series. This of course leads to a look-ahead bias, but since both DRL and 1/N benchmark portfolios are built from the same identical survivorship universe, any residual look-ahead bias applies equally to each strategy and thus cancels out in their relative performance comparison. Yet it is worth noting that this may overstate absolute risk-adjusted returns and thus should therefore be cautious when making any generalisations about attainability of these results in real time. By diversifying sectors, some colinearity reduction is introduced: stock returns within a sector are usually more correlated than across sectors. So multi-sector means the correlation matrix of returns is less dominated by a single factor. However, since all are equities, there will still be a common market factor. That is fine and reflection of reality. Sector diversity and mixed volatility profiles encourage the RL agent to rotate between cyclical and defensive exposures rather than memorising a single dominant factor, thereby providing a more stringent test of dynamic portfolio skills.

Historical prices are downloaded from *Refinitiv Datastream*, where total return series incorporate splits and cash dividends. Additionally, the 3-month Treasury-bill yield, price to earning ratios, and 1-month bond volatility index are also downloaded from *Refinitiv Datastream*. The data coverage includes a daily panel spanning from January 2010 to December 2024, which is roughly 3 800 trading days that cover the post global financial crisis recovery, the COVID-19 turmoil, and a variety of subsequent market regimes. Observations from 2010–2019 (about 2 520 days) are reserved for model training, while the 2020–2024 window is held strictly out-of-sample for performance evaluation to maintain the temporal flow of information.

Total-return indices, which automatically adjust for corporate actions such as splits and dividends, are used to compute continuous logarithmic daily returns as

$$r_{t,i} \;=\; \ln\!\left(\frac{P_{t,i}}{P_{t-1,i}}\right) \;=\; \ln(P_{t,i}) \;-\; \ln(P_{t-1,i}),$$

eliminating the need for separate dividend handling. The risk-free benchmark is the three-month U.S. Treasury bill. Its yield is converted into a constant maturity with daily series and merged with the price panel, ensuring that excess returns, Sharpe ratios, and Jensen's alphas are grounded in an observable short-term rate. This forward-chaining design where training the agent on one market era and evaluating it on genuinely unseen conditions creates a realistic forward looking scenario. The agent is trained on one market era and judged on future, unseen conditions, providing a harsh generalisation check.

4.2 Defining the state space

State representation for the reinforcement-learning state must relay market information for each asset and the agent's current allocation. In addition to historical price data, the RL survey suggests including techinical indicators and market indicators can help (Bai et al., 2024, 5). In line with quantitative finance practices, the feature set is designed to be expressive yet compact.

The state vector is built around four stock specific signals. First, the daily log-returns observed over the most recent one to five trading days capture short-term momentum and potential mean-reversion effects. Second, a rolling 21-day standard deviation of returns serves as a volatility proxy, linking each asset's expected reward to its recent risk. Third, a small set of higher-level signals such as 14-day RSI is added to accelerate learning. Fourth, a rolling price-to-earnings (P/E) ratio presents a fundamental feature that may help the agent discriminate between over- and undervalued stocks. To fulfill the state vector it is augmented with two additional elements. The previous portfolio weights w_{t-1} are appended, letting the agent weigh the cost of re-balancing when transaction fees apply. Lastly two macro variables are added: the three-month Treasury bill yield and a bond-market volatility index add information about prevailing funding conditions and market stress.

With N stocks and k stock-level features, the state dimension equals $N \times k$. For $N = 10$ and $k = 5$ this is 50 elements. Adding the previous weights (+10) and two macro variables still keeps the vector well below a few hundred inputs which is manageable for a fully connected neural network. For larger universes ($N = 50$) the dimension rises to roughly 250–400, so redundant signals are intentionally avoided.

Principal component or factor representations could decorrelate highly similar stocks, but at the cost of interpretability. However, machine-learning-based methodologies including neural networks are highly immune to collinearity caused biases (Lindner et al., 2022, pp. 1311–1312). Since moderate collinearity is tolerable for modern networks, direct stock representation is retained in this work. The final state vector s_t for day t chaines four stock-specific features including recent returns, 21-day volatility, a 14-day RSI, and the rolling P/E together with the previous allocation w_{t-1} and two macro variables. Refreshed each trading day, this vector is fed to the policy network, which outputs the next allocation w_t.

Neural networks train more robustly when input features share a comparable numerical range (Sola and Sevilla, 1997, pp. 1467–1468). Daily returns, technical indicators and fundamentals are therefore transformed as follows. Returns instead of raw prices are calculated as log returns: $r_{t,i} = \ln(P_{t,i}/P_{t-1,i})$. This representations removes the scale effect of share price. For example, a \$20 and a \$300 stock contribute equally if they move by $\pm 5\%$. Feature standardisation is applied by setting for every feature x its training set mean μ and standard deviation σ, and the standard score $\tilde{x} = (x - \mu)/\sigma$ is used in all splits (train and test). The transformation yields approximate zero mean and unit variance, which accelerates optimisation. For Outlier control, extremely volatile variables are capped inside $[\,-3\sigma,\, +3\sigma\,]$. Bounded indicators such as RSI are rescaled to $[0, 1]$. Non-stationarity causion is handled. Because financial series are non-stationary, fixed μ and σ from the training window may become sub-optimal in later periods. Nevertheless, static scaling avoids data leakage and sufficed in preliminary experiments. This pipeline delivers inputs of similar scale to the neural networks while preserving the temporal integrity of the train–test split.

In equity markets stocks often move together, particularly during market wide events which can lead to highly correlated returns and multicollinear feature sets. Extreme collinearity may obscure the attribution of risk or return to individual assets, yet in a portfolio setting an RL policy can simply treat strongly correlated stocks as a single factor exposure by assigning them similar weights. The following measures are adopted to keep collinearity at a manageable level: Stock choices are drawn from multiple sectors, reducing the likelihood of high pairwise correlations. Redundant indicators are avoided. For example, only one short-term momentum signal is chosen. Features are computed on returns rather than raw prices, improving stationarity and relevance to re-balancing. The training set correlation matrix is inspected as: if any asset pair or indicator pair exhibits $\rho > 0.90$, one variable is dropped or substituted to maintain informational diversity. Dimensionality reduction techniques such as PCA can transform highly correlated returns into orthogonal factor scores. Some prior works like Shen et al. (2015) did use PCA on stock returns to define "arms" for a bandit algorithm, essentially picking uncorrelated

portfolios rather than stocks. However, doing so sacrifices some transparency. In practice, correlated price movements do not break the approach: the RL agent is free to allocate similar weights to assets that move alike. By combining sector diversification, a non-redundant feature set and ongoing correlation checks, severe multicollinearity is prevented from dominating the state space.

Following this pipeline guarantees that each state presented to the RL agent contains well-aligned technical features, macro signals and fundamentals that are clean, scaled and strictly historical.

4.3 Model Implementation

The experiments are set up to answer the research questions. This includes the configuration of portfolio management environment for the RL agents, the training procedure for each algorithm, the definition of reward and any constraints, and the evaluation protocol comparing against the 1/N strategy.

All methods are implemented in Python and rely exclusively on open-source software. The implementation of the reinforcement learning algorithms were utilized with Stable Baselines3 (SB3), which provides production-ready versions of PPO, A2C, and DDPG. SB3 is a set of reliable implementations of reinforcement learning algorithms in PyTorch, documented by Raffin et al. (2021). PyTorch is a machine learning library that offers a deep learning framework that focuses on usability and speed, documented by Paszke et al. (2019). OpenAIGym offers the environment interface under which StockTradingEnv is defined. StockTradingEnv is name of the custom environment used in this research. OpenAIGym is a toolkit for reinforcement learning research, documented by Brockman et al. (2016). The complete software stack therefore combines reliable RL implementations with a flexible simulation interface and a modern deep-learning framework.

The portfolio-allocation problem is modelled as a custom OpenAIGym environment, designated as StockTradingEnv. The environment packages the market data, state representation, admissible actions, and reward signal into a sequential decision process in which a single trading day forms one environment step. In StockTradingEnv the state comprises historical features and current weights, actions are allocation vectors, and rewards are daily log returns. Simplified StockTradingEnv is presented in Appendix 4. In the following, the state and action spaces are specified, the transition dynamics are described, and the reward function is constructed.

At each trading day t, the state vector s_t gathers all information the agent needs to choose a new portfolio allocation. The environment provides several key components

for this state vector. Firstly, it includes a windowed history of technical features for each stock. A rolling window of length $w = 5$ days is kept. For each of the N stocks, F technical features including daily returns, RSI and rolling volatility are collected, producing a tensor of shape (w, N, F), which is flattened into one vector:

$$\text{Features} = [x_{t-w+1,1}, \ldots, x_{t,1},\ x_{t-w+1,2}, \ldots, x_{t,2},\ \ldots,\ x_{t-w+1,N}, \ldots, x_{t,N}],$$

where $x_{t,i}$ denotes the feature vector for stock i on day t.

Secondly, the previous portfolio allocation $w_{t-1} \in R^N$ is appended so the agent knows its current positioning, which is critical when weight changes (and any transaction costs) are calculated. Finally, to enable the policy to condition on broader market signals, the state includes a macro vector $m_t \in R^M$ and per-stock fundamentals $f_{t,i}$ are linked, enabling the policy to condition on both technical and macro/fundamental signals. Putting these parts together,

$$\mathbf{s}_t = \Big[\underbrace{x_{t-w+1,1}, \ldots, x_{t,N}}_{\text{flattened stock features}},\ \underbrace{w_{t-1,1}, \ldots, w_{t-1,N}}_{\text{prev. weights}},\ \underbrace{m_t}_{\text{macro}},\ \underbrace{f_{t,1}, \ldots, f_{t,N}}_{\text{fundamentals}}\Big]^\top.$$

This consolidated state is passed to the policy network at every step.

The action a_t on day t is the new weight vector

$$w_t = (w_{t,1}, \ldots, w_{t,N})^\top,$$

where each component satisfies $w_{t,i} \geq 0$ and $\sum_{i=1}^{N} w_{t,i} = 1$, meaning the agent invests 100% of its capital across the N stocks. The mechanism for generating this action vector varies depending on the reinforcement learning algorithm.

For stochastic policy-based methods like Proximal Policy Optimization (PPO) or Advantage Actor-Critic (A2C) the action is first represented as a continuous vector in R^N. The policy network outputs a diagonal Gaussian whichs mean is the unconstrained vector $\tilde{u}$. A softmax is subsequently applied to map $\tilde{u}$ onto the simplex,

$$w_t = \text{softmax}(\tilde{u}),$$

ensuring non–negative weights that sum to unity. The resulting w_t is used directly for rebalancing.

For deterministic algorithms like the Deep Deterministic Policy Gradient (DDPG), the actor's final layer is followed by a softmax, so that the network itself emits a valid weight

vector:

$$w_t = \text{softmax}(\tilde{u} + noise).$$

During training, exploration noise is added to $\tilde{u}$ before the softmax. The environment therefore always receives an allocation that lies strictly on the simplex. In every scenario the allocation satisfies $w_t \in [0,1]^N$ and $\sum_i w_{t,i} = 1$, a requirement consistent with an equity-only portfolio that disallows short selling and idle cash.

After the environment has been supplied with the weight vector w_t, the portfolio is re-balanced to those weights as a state transition. During the following day asset prices evolve exogenously, and the portfolio value is updated as

$$V_{t+1} = V_t\Big(1 + \sum_{i=1}^{N} w_{t,i} R_{t,i}\Big),$$

where $R_{t,i}$ is the realised return of asset i from day t to $t+1$. The next state s_{t+1} is then assembled from the next window of features $\{x_{t+1-w+1,i}, \ldots, x_{t+1,i}\}$ for every stock, the newly chosen weights w_t that are now stored as the "previous-weights" block, and the updated macro and fundamental data for day $t+1$.

The base reward is the logarithmic portfolio return,

$$r_t = \ln\Big(\frac{V_{t+1}}{V_t}\Big) = \ln\Big(1 + \sum_{i=1}^{N} w_{t,i} R_{t,i}\Big).$$

Instead of maximizing the discounted sum, a better objective is to maximize expected log-return, because the logarithm's concavity makes the problem concave aligning better with long-run terminal wealth. When turnover costs are included, the reward is reduced by

$$\tilde{r}_t = \ln(1 + R_{p,t}) - \underbrace{c \sum_{i=1}^{N} |w_{t,i} - w_{t-1,i}|}_{\text{cost\_term}},$$

where $R_{p,t}$ is the gross portfolio return and $c > 0$ is the proportional fee rate. The penalty discourages frequent or large re-balancing moves and ensures that long-term policy is updated and the model doesn't stop early.

4.4 Training the agents

Three deep-reinforcement-learning agents PPO, DDPG and A2C are trained on the StockTradingEnv introduced in Section 4.3. All agents receive identical features, episode boundaries and evaluation protocols so that the comparison remains fair. Hyperparameters are

selected either from values commonly reported in the literature or from a modest grid search, described below.

Identical network sizes are used for all agents to ensure a consistent basis for comparison. The actor network and, when applicapble, the critic network adopt a uniform structure. This architecture consists of two hidden layers, each containing 64 units and utilizing the Rectified Linear Unit (ReLU) as the activation function. This same two-layer, 64-unit ReLU structure is employed for the critic networks in algorithms that require one. While the hidden layers are standardized, the output layers differ based on the algorithm's requirements. For the stochastic agents PPO and A2C, the network outputs the mean and standard deviation parameters that define a diagonal Gaussian distribution from which actions are sampled. In contrast, the DDPG agent's network emits a single deterministic action. Regardless of the agent type, a final softmax activation function is applied to the output logits. This critical step ensures that the resulting portfolio weights are properly normalized, summing to one and thus representing a valid, fully invested allocation.

Each episode spans roughly 250 trading days, suggesting that a pure return-sum objective with $\gamma = 1$ could be used. However, practical and theoretical considerations favour a discount factor that is slightly below unity. A range of $\gamma \in [0.95, 0.99]$ is adopted for all agents because it preserves the standard convergence guarantees of RL algorithms, it places a mild emphasis on near-term returns while still valuing long-term performance, and it aligns with defaults in widely used libraries such as Stable Baselines3.

This range $\gamma \in [0.95, 0.99]$ has frequently been used in prior financial RL work (eg., (Mohammed et al., 2021; Zou et al., 2024)) to maintain theoretical convergence properties and to ensure that returns far in the future do not dominate the current updates in an unbounded manner.

A moderate grid search explores learning rates, discount factors, step sizes and transaction-cost rates introduced in Table 3:

Table 3: Hyperparameter Grid Search Values

Hyperparameter	PPO	A2C	DDPG
Discount Factor (γ)	$\{0.95, 0.99\}$	$\{0.95, 0.99\}$	$\{0.95, 0.99\}$
Learning Rate	$\{1e{-}4, 3e{-}4\}$	$\{1e{-}4, 3e{-}4\}$	$\{1e{-}3, 5e{-}4\}$
Steps (n_{steps})	$\{256, 512\}$	$\{5, 8\}$	N/A
Value Function Coeff.	N/A	$\{0.25, 0.5\}$	N/A
Transaction Cost Rate	N/A	N/A	$\{0.0005, 0.001\}$

Each hyperparameter combination is trained for 20k–50k time-steps on the training set (multiple episodes) and evaluated on a separate validation year. The configuration that maximises a composite metric of net return, Sharpe ratio and turnover is retained.

All algorithms are implemented with the Stable Baselines3 library (PyTorch backend). The custom StockTradingEnv Gym environment integrates seamlessly with the library's policy, replay-buffer and optimisation modules while allowing task-specific customisations such as action projection.

On-policy agents (PPO, A2C) the training data is divided into multiple episodes of roughly 250 trading days. After each episode the portfolio value is reset to 1 and weights are initialised to equal allocations. Episodes are sampled chronologically. For off-policy agent (DDPG), year-long episodes are used to establish clear boundaries while the replay buffer is filled across episodes and sampled uniformly for updates.

During the grid search each agent is trained for 15k–50k time-steps, enough to observe performance plateaus and identify stable hyperparameter choices. The best configuration is then re-trained for 200k–500k time-steps. Periodic validation checks allow early stopping if performance ceases to improve, reducing the risk of over-fitting.

4.5 Performance evaluation

The deep-RL agents are evaluated against a transparent, equal-weight benchmark that invests the same fraction in each of the N assets. Daily rebalancing keeps the comparison on the same one-step-per-day schedule used by the learning agents. At the start of the test period the capital is split equally, $w_{0,i} = 1/N$ for $i = 1, \ldots, N$, and the portfolio value is normalised to 1. At every trading day t the weights are reset to $w_{t,i} = 1/N$ $\forall\, i$, fully offsetting any drift caused by price movements. In the base experiment the benchmark is assumed frictionless, mirroring the initial DRL runs. When transaction costs are introduced, the 1/N portfolio pays the same costs rate each day to restore the equal weights, preserving fairness against DRL policies that may rebalance less often. The daily portfolio return is

$$R_t^{1/N} = \frac{1}{N} \sum_{i=1}^{N} R_{t,i},$$

and the value update follows $V_{t+1}^{1/N} = V_t^{1/N}(1 + R_t^{1/N})$. Cumulative return, Sharpe ratio, drawdown and turnover are computed exactly as for the learning agents.

A strict equal-weight strategy provides a low-information, diversified baseline. Although equal-weight funds often rebalance monthly or quarterly to mitigate turnover, a daily schedule simplifies the comparison: both benchmark and DRL agents act once per trading day. If a DRL policy genuinely exploits predictive structure in the features, it is expected to surpass this 1/N benchmark on risk-adjusted metrics such as annualised Sharpe and maximum drawdown.

The final policies of PPO, DDPG and A2C are back-tested on a held-out test period and compared with the 1/N benchmark. No further learning occurs during testing. In back-testing protocol the portfolio value is set to $V_0 = 1$ and the agent's trained parameters are loaded. In the daily simulation the agent observes the state s_t, a weight vector w_t is produced, the environment re-balances to w_t, and next-day returns R_t update the value via

$$V_{t+1} = V_t\Big(1 + \sum_i w_{t,i} R_{t,i}\Big).$$

This procedure creates a time series r_t of daily log-returns for both the RL agent and the 1/N benchmark.

For each strategy the following quantities are reported: compound annual growth rate (CAGR), annualised volatility σ_{ann}, maximum drawdown (MDD), total transaction costs, and Sharpe ratio $= \dfrac{\bar{r} - r_f}{\sigma_{\text{ann}}}$, where $\bar{r}$ is the mean daily return and r_f the daily risk-free rate.

For statistical comparison a robust Sharpe difference testing by (Ledoit and Wolf, 2008) is adopted. The block bootstrap approach proposed by Ledoit and Wolf resamples blocks of daily returns to preserve autocorrelation structure. Under the null hypothesis of equal Sharpe ratios, estimation about the distribution of Sharpe differences is tested. Then a p-value is computed to see if the observed difference in Sharpe is statistically significant.

Because RL training is partly stochastic (e.g., random initialization, exploration noise), each DRL agent is trained multiple times (with different random seeds) to gauge variability in outcomes. If, for instance, PPO consistently outperforms 1/N across several runs, that provides stronger evidence of robust outperformance. If outcomes vary widely, the average and standard deviation of the performance metrics is reported.

Final evaluation steps summarized: a set of performance metrics for each agent and for 1/N, statistical tests (Ledoit and Wolf (2008) block bootstrap) to ascertain if differences in Sharpe are robustly significant, and lastly seed repetition to check the consistency of learned policies. These steps provide a rigorous foundation for concluding whether or not our DRL agents truly surpass the straightforward 1/N baseline.

5 RESULTS AND DISCUSSION

5.1 Data Characteristics and Feature Behavior

The investment universe includes ten large-capitalisation U.S. equities: Apple (AAPL), JPMorgan Chase (JPM), Johnson & Johnson (JNJ), Walmart (WMT), NextEra Energy (NEE), NVIDIA (NVDA), Alphabet (GOOGL), Lockheed Martin (LMT), Coca-Cola (KO), and McDonald's (MCD).[1] Daily prices span several years and are split into a training and a test sample. Descriptive statistics are presented in Table 4.

Table 4: Key descriptive statistics for the ten constituent equities. Annual figures are geometric unless noted.

Ticker	Ann. Return (%)	Ann. Vol. (%)	Beta (1/n)	Skew	Kurtosis	Avg. P/E
AAPL	30.66	27.27	1.24	-0.05	5.61	20.42
GOOGL	23.92	26.85	1.22	0.41	8.68	32.73
JNJ	10.09	16.50	0.65	-0.11	9.49	21.44
JPM	19.81	27.23	1.20	0.27	10.68	11.56
KO	10.51	16.72	0.71	-0.62	9.51	25.67
LMT	18.47	20.54	0.79	-0.29	13.59	16.53
MCD	14.71	18.34	0.76	0.40	32.39	23.35
NEE	17.75	21.59	0.82	-0.21	11.72	25.13
NVDA	65.22	44.67	1.96	0.67	8.66	49.58
WMT	15.80	18.96	0.65	0.12	16.68	24.01

NVDA combines the highest annual return with the greatest volatility and a beta almost twice that of the equal-weight benchmark. Low-beta defensives (JNJ, KO) display muted skewness and thinner tails, whereas MCD and LMT show pronounced kurtosis, signalling fat-tail risk.

Table 5: Descriptive statistics for the two macro-economic features.

Variable	Mean	Std.	Min	Max	Skew	Kurtosis
TBILL_3M	1.24	1.74	−0.05	5.36	1.37	0.46
BONDVOL_1M	78.62	25.30	36.60	198.70	0.83	0.18

The three-month T-bill rate is strongly right-skewed, capturing the abrupt hiking cycle after a prolonged near-zero period. Bond-market volatility is likewise right-skewed, with occasional spikes above 200 basis points.

Tech names (AAPL, NVDA, GOOGL) exhibit the tightest cluster, suggesting a sector

[1] Ticker conventions follow CRSP. Company names are given only for clarity; the portfolio is managed strictly at the ticker level.

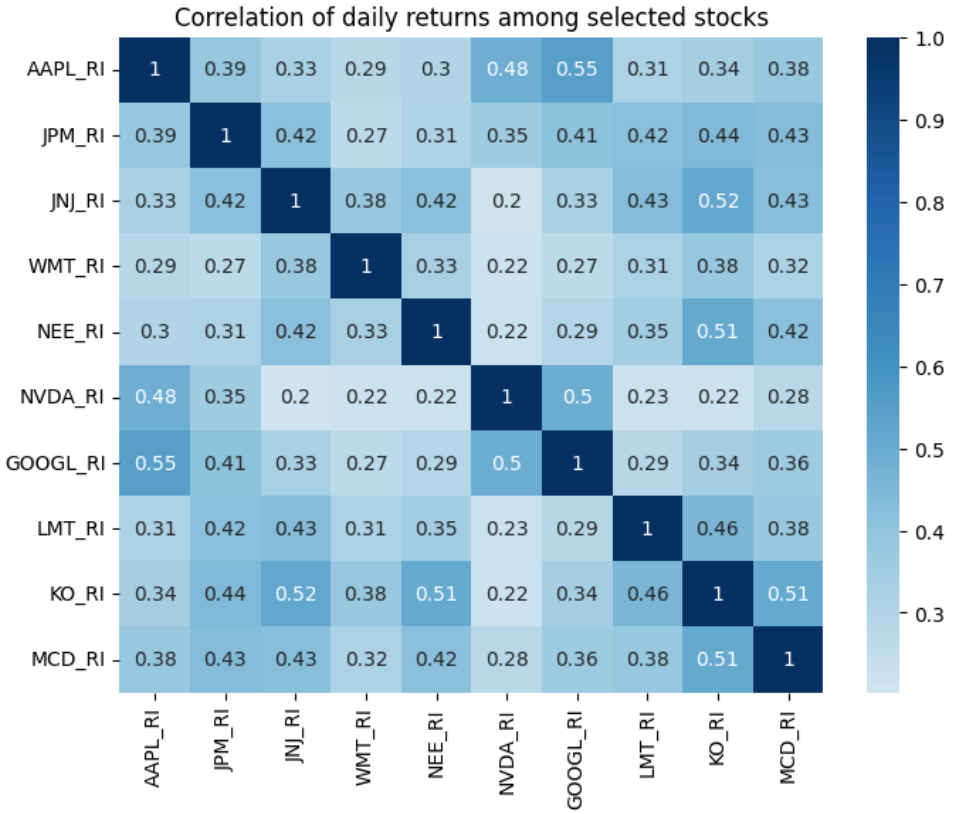

Figure 6: Correlation matrix of daily log-returns for the ten equities.
Darker shading denotes stronger positive correlation.

factor the DRL agents may hedge or exploit. Cross-sector correlations remain moderate, preserving diversification potential, in fact, no pairwise correlation exceeds roughly 0.60, so multicollinearity is limited and dimensionality reduction techniques such as PCA are not strictly necessary for the state space construction. Descriptive statistics for these variables appear in Figure 6.

It is worth noting differences between training and testing regimes. An essential observation is that the distribution of key features shifts between the two periods. During most of the training sample, short-term interest rates remained near the zero lower bound, whereas the test period contains episodes of rising rates and heightened volatility. Figure 7 illustrates the two way nature of TBILL_3M: the primary mode clusters around 0% while a secondary mode appears between 4 % and 5 %, signalling a regime change toward tighter monetary policy late in the sample.

Similarly, the distribution of BONDVOL_1M is markedly right-skewed (Figure 8). Although moderate volatility prevails for most observations, rare spikes to extreme levels occur. Some of these events lie entirely outside the range witnessed during training. Also, DRL agents must contend with out-of-sample conditions that partially violate the stationarity assumption, which is a known challenge in financial machine learning.

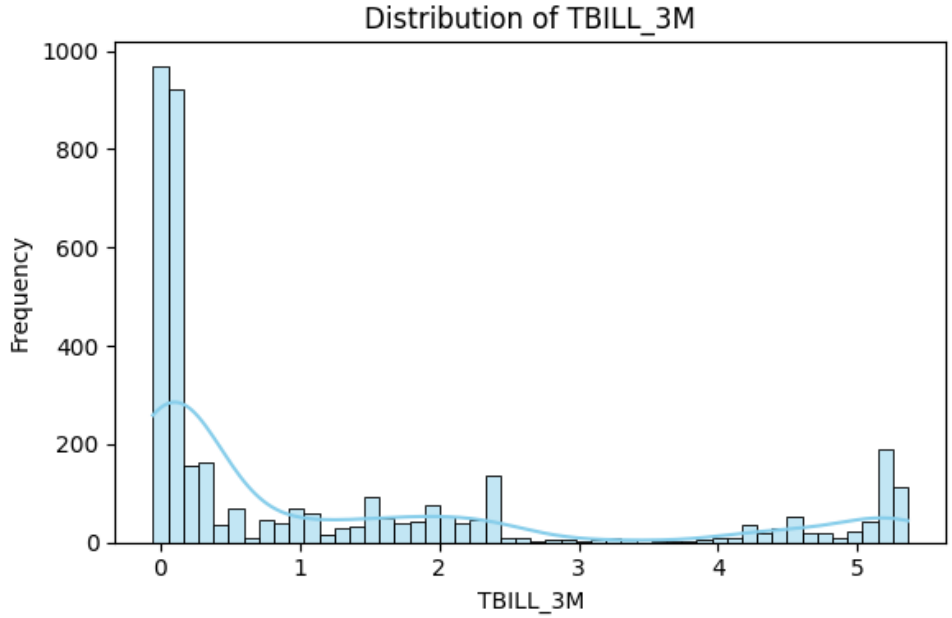

Figure 7: Empirical distribution of TBILL_3M across the full sample.
The primary mode clusters near 0 % (x in percentage points) while a secondary peak around 4 - 5 % marks the regime shift toward higher short-term rates.

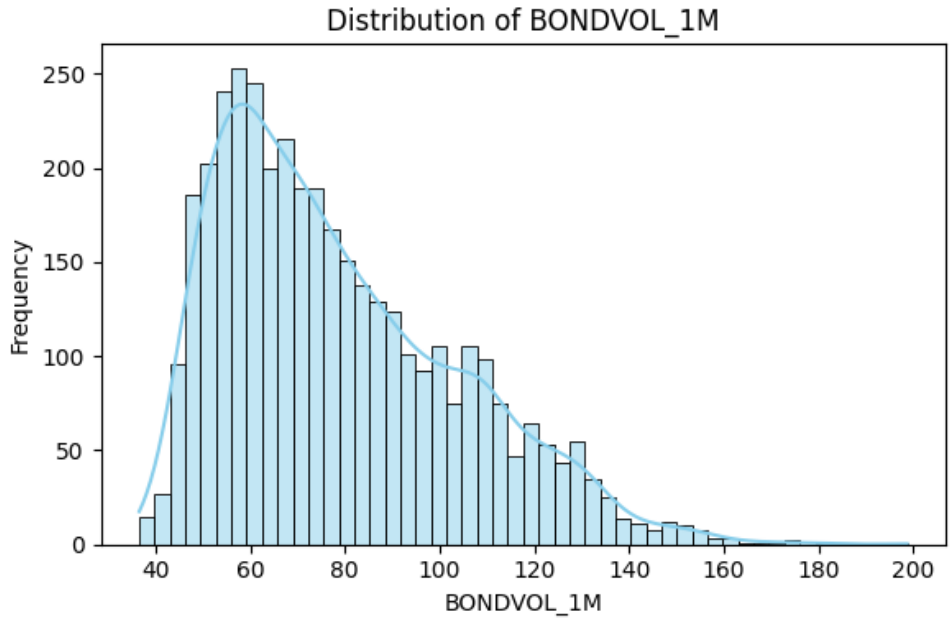

Figure 8: Distribution of the bond-market volatility BONDVOL_1M.
The pronounced right tail (x in bp) reveals rare but extreme volatility spikes that the DRL agents must handle out-of-sample.

Features such as interest-rate levels and volatility surges can exert first-order influence on optimal portfolio choice. A financially rational policy might reduce equity exposure when BONDVOL_1M exceeds a stress threshold, or rebalance toward defensive assets when TBILL_3M rises sharply, reflecting an increased risk-free alternative and a potential contraction in the equity risk premium.

If the DRL agents have genuinely internalised such relations, it is expected for their allocations to react coherently to these signals in the test sample. By contrast, overfitting to the low-volatility, low-rate training regime could manifest as poor generalisation once those assumptions break down. The performance analyses will revisit these hypotheses and examine whether the strategies adapt effectively or falter under the observed regime shifts.

5.2 DRL Model Configuration and Hyperparameters

Before presenting performance outcomes, here is a summarize of the final model configurations. Each DRL algorithm underwent a modest grid search on a validation set to balance reward maximisation with practical considerations such as risk and trading costs. All three agents share an identical neural network architecture for comparability: two hidden layers of 64 ReLU units to control for capacity differences. Table 6 lists the hyperparameters ultimately selected for PPO, A2C, and DDPG.

Table 6: Final hyperparameters chosen for each DRL algorithm.

Hyperparameter	PPO	A2C	DDPG
Discount factor γ	0.99	0.95	0.99
Learning rate	3×10^{-4}	1×10^{-4}	1×10^{-3}
Rollout length / n-steps	256	5	–
Value-loss coefficient	–	0.50	–
Transaction-cost rate (train)	0.001	0.001	0.001

Note: " - " indicates not applicable (e.g., DDPG is off-policy and does not use fixed-length rollouts in the same way as PPO/A2C share an implicit transaction cost penalty via environment).

The above values were arrived at by a modest grid search. For example, the discount factor was tested $\gamma \in \{0.95, 0.99\}$ for each algorithm, and ultimately $\gamma = 0.99$ was chosen for PPO and DDPG to emphasize long-term return (and because it improved validation performance), whereas A2C slightly preferred $\gamma = 0.95$, potentially because of stability concerns when bootstrapping with longer horizons. Learning rates were explored in the range 10^{-4} to 10^{-3}. PPO performed best with a learning rate of 3×10^{-4}, while A2C, being somewhat less sample-efficient, was kept at a conservative 1×10^{-4} to avoid divergence. DDPG's actor-critic optimizer benefited from a relatively larger step size (10^{-3}) for faster convergence, given its off-policy training can utilize more data. The n-steps (the

number of steps per update) for the on-policy methods were also tuned: PPO uses 256-step rollouts per update (roughly covering a bit more than one trading year per episode in the setup), balancing bias and variance in advantage estimation. A2C uses 5-step returns (which is standard for A2C and was found to work well in validation). Also the coefficient was adjusted on A2C's value-function loss (0.5 was found slightly better than 0.25), which helps A2C learn state values without overweighting this term. Lastly, an explicit transaction-cost rate of 0.1% (0.001 in fractional terms) was built into the environment for all strategies during training to penalize excessive turnover. It was confirmed that including this modest cost in training encourages the agents to moderate their trading frequency. Notably, in the grid search for DDPG I tried a slightly lower cost rate (0.05%), but the higher cost rate of 0.1% yielded a better return–turnover trade-off and was chosen as the final setting. Overall, these hyperparameters are consistent with values commonly used in the DRL literature for trading tasks and were selected to give each algorithm the best chance to learn a robust policy.

5.3 Out-of-Sample Performance Comparison

First examined is the cumulative portfolio value trajectories in the test period for each strategy. Figure 9 plots the evolution of portfolio wealth (normalised to an initial value of 1.0) over the entire test horizon for the equal-weight (1/N) strategy and the DRL-driven portfolios (PPO, A2C, DDPG).

Several observations can be made from this figure. All strategies achieved substantial growth over the test period, indicating positive returns in a broad bull-market trend. However, the 1/N portfolio ultimately ends with the highest terminal value (approximately 2.7 times the starting value), closely followed by PPO and A2C. The DDPG strategy's final wealth is a bit lower and has the highest volatility. In terms of sheer final return, none of the DRL trategies convincingly exceeds the naive 1/N yet indeed, the equal-weight strategy slightly outperformed all three DRL agents.

It is insightful to consider the journey of these returns, not just the endpoint. The plot reveals periods where DRL strategies overtook the benchmark and vice versa. For instance, in the middle of the test period (around days 500 - 600), the A2C portfolio surged above the 1/N, peaking around a value of 2.00 when 1/N was at 1.7. However, A2C then suffered a sharper drawdown around days 600 - 700 and 800 - 1000, erasing its lead. As contrast, PPO's trajectory is relatively more stable: it tracks the 1/N portfolio closely, with only minor deviations. The DDPG path is the most volatile of the three DRL agents, dipping more aggressively during certain downturns yet rallying strongly toward the end to close the gap with PPO.

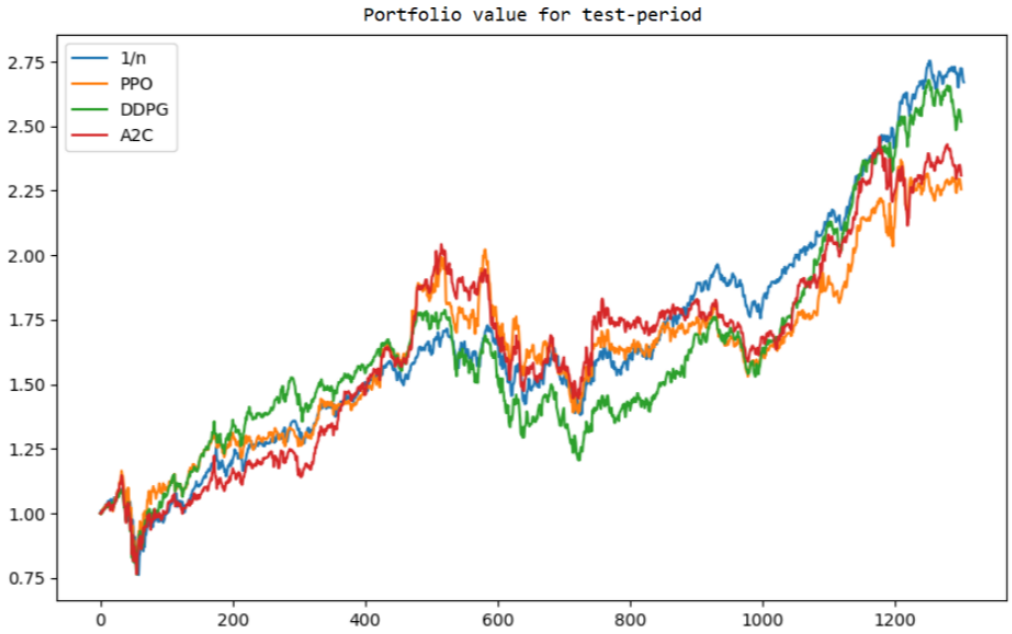

Figure 9: Cumulative portfolio value in the test period
Portfolio value normalised to 1.0 at inception. The equal-weight benchmark (blue) finishes marginally ahead of the DRL strategies: PPO (orange) and A2C (red) tracks closely, while DDPG (green) exhibits the greatest volatility.

These patterns indicate that the DRL strategies dynamically adjusted their exposures over time, occasionally capturing gains or avoiding losses better than 1/N, but at other times making missteps the static benchmark avoided. This hints a key theme: variability versus consistency as the equal-weight portfolio, being a passive buy-and-hold with periodic rebalancing, shows a steadier upward trend and smaller oscillations, aside from market-wide moves.

The return performance can be quantified more formally via the compound annual growth rate (CAGR). Over the test horizon of T years (roughly $T \approx 5$ years given ~ 1250 trading days), the 1/N portfolio achieved a CAGR of about $\sim 21\%$, whereas PPO's CAGR was approximately 17%, DDPG's about 20%, and A2C's about 18%. These CAGRs (which will be detailed in Table 7) confirm that the equal-weight strategy provided the highest long-run growth. Nonetheless, the differences are not enormous as DDPG in particular delivered almost the same growth rate, falling short by only a percentage point. In absolute return terms, therefore, the advantage of the benchmark is modest and is later tested by running models with multiple seeds. Finally, should be noted that all portfolios benefitted from a generally rising market. This means even the "worst" strategy (PPO) more than doubled the initial capital in the given period, which is a strong absolute result. However, for evaluating skill or added value, risk-adjusted returns and consistency should be more informative.

Table 7: Out-of-sample performance metrics.

Metric	**PPO**	**A2C**	**DDPG**	**1/n**
Final portfolio value rounded	2.20	2.25	2.55	2.70
CAGR (%)	17.1	17.6	19.6	20.9
Sharpe ratio	0.805	0.840	0.916	1.075
Max drawdown (%)	−31.9	−33.4	−32.5	−30.7
Trades (count)	421	153	1 242	1 258
Total cost	0.70519	0.31583	0.54653	0.02070

The Sharpe ratio provides a summary of risk-adjusted return (assuming reasonably stable distributions). Over the entire test period, the annualized Sharpe ratios of the strategies were as follows: the 1/N portfolio attained the highest Sharpe around 1.1, DDPG was slightly lower around 0.9, A2C further behind with 0.84, and PPO roughly in 0.8. These values are reported in Table 7. In effect, 1/N had the best trade-off between mean return and volatility. This might be surprising at first glance since one might expect a sophisticated DRL agent to deliver superior Sharpe if it can reduce risk during bad times. However, the results show that the DRL strategies did not substantially outperform the naive diversification on a Sharpe basis. One contributing factor is volatility of returns: the DRL strategies introduced additional variability by shifting portfolios. While they sometimes reduced exposure successfully before a downturn, at other times they shifted into the wrong assets and increased volatility. The equal-weight portfolio, by contrast, always holds a broad mix, which smooths out idiosyncratic fluctuations and yields relatively stable returns (aside from systemic market moves).

To illustrate the time-varying nature of risk-adjusted performance, Figure 10 plots the rolling 60-trading-day Sharpe ratio (annualized) for each strategy throughout the test period. The 1/N portfolio (red line in this figure) maintains a consistently positive Sharpe in most periods and reaches very high Sharpe values (above 5 or even 8 on occasion) during strong market rallies with low volatility. The DRL strategies show more volatility in their short-term Sharpe. For example, in early parts of the test, the A2C strategy (orange line) spiked to a rolling Sharpe of 5, outperforming 1/N at that moment, but shortly thereafter its Sharpe plunged toward 0 or negative when some trades turned bad. Similarly, DDPG (green line) had a period around day 600 where its 60-day Sharpe fell below 2, whereas 1/N at the same time was around 0. Later in the test (day 1050-1150), all strategies saw Sharpe ratios jump as the market rallied. Interestingly, 1/N's Sharpe (red) soared the highest (peaking above 8) because it was fully exposed to the rising market, whereas PPO and A2C (blue and orange) remained more muted around 5-6. By the very end of the test, the rolling Sharpes of all strategies converge downward as a minor drawdown occurs, with some even turning slightly negative. These fluctuations underscore that

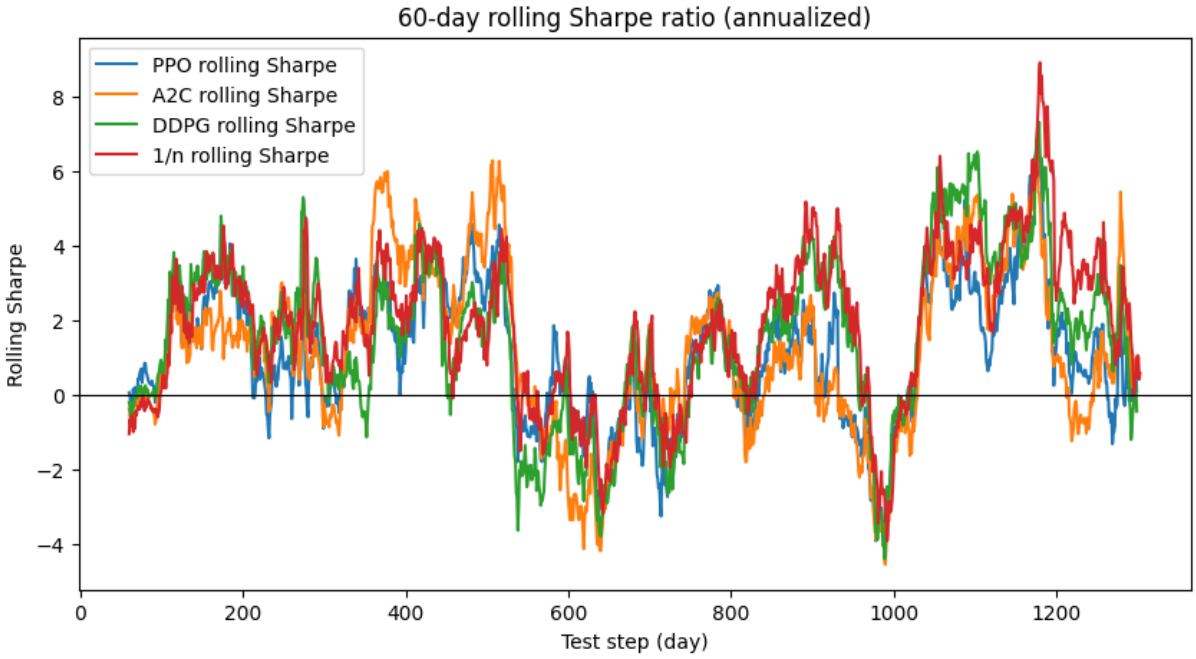

Figure 10: Rolling 60-day annualised Sharpe ratio for each strategy.

the DRL strategies' performance was more erratic in the short run. They achieved high risk-adjusted returns in certain windows but could not maintain that consistently.

Another critical risk metric is the maximum drawdown (the largest peak-to-valley percentage loss during the period). In results, the max drawdown for the 1/N portfolio was about -30.7%. The DRL strategies fared slightly worse overall: PPO suffered a drawdown of roughly -31.9%, A2C the largest at around -33.4%, and DDPG about -32.5%. These values in Table 7 show that none of the DRL agents improved on the benchmark's downside protection. A2C increased downside risk the most, while PPO and DDPG were only marginally deeper than the benchmark.

The portfolio weight evolution plot (Figure 11) vividly illustrates the differences in strategy behaviour. The 1/N benchmark (bottom-right panel) is trivial: ten static bands of equal width ($\approx 10\%$) apart from minor drift before rebalancing. By contrast, the DRL panels are highly dynamic: PPO (top-left) appears to cycle between two dominant states. In one state it splits capital almost evenly between two favoured stocks whilst in the other it reverts to an approximate equal-weight allocation across the remaining eight. Short excursions away from these states also occur. This "bimodal" pattern is plausibly an echo of the training phase, where portfolio weights were initialised at 1/N. Once PPO finds a near-balanced configuration, it tends to sit there to save transaction costs. A2C (bottom-left) shows a related but less symmetric behaviour: it also favours a handful of quasi-balanced mixes, yet the weights inside each mix are uneven, suggesting the agent locks onto a temporary "optimal" basket for a few days before abruptly rotating else-

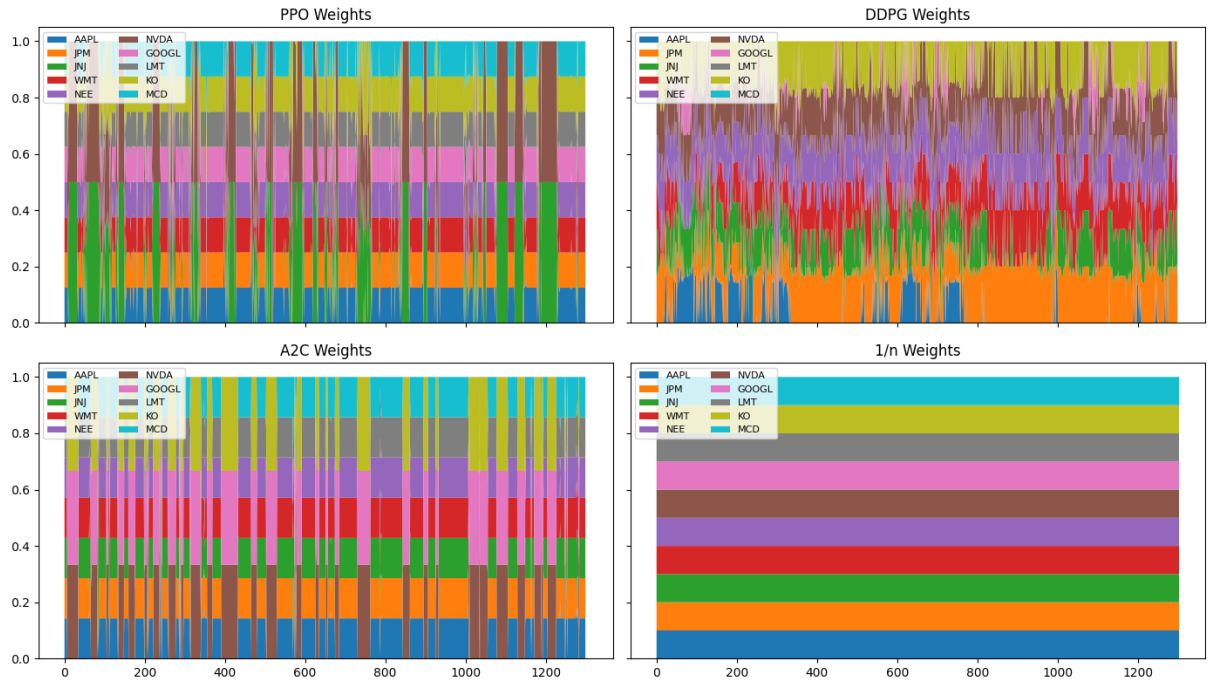

Figure 11: Evolution of portfolio weights
Each coloured band represents the fraction of capital invested in one of the ten stocks.

where. DDPG (top-right) is the most chaotic. Its continuous-action updates lead to almost frame-by-frame adjustments with no stable pattern, consistent with a policy that chases short-term signals rather than settling into discrete regimes.

Thus the DRL agents continually reshuffle their portfolios whereas the 1/N strategy simply holds everything. Active rotation is not inherently problematic, but given the 0.1% transaction cost penalty it erodes much of the DRL excess return, helping to explain why the simple equal-weight portfolio ultimately outperforms on a net basis.

Figures 12 and 13 repeat the experiment with markedly shorter training horizons: roughly 30 000 steps for A2C, 50 000 for PPO and 10 000 for DDPG, and then reinitialises each algorithm with distinct random seeds. For every seed the test period wealth path and Sharpe ratio is recorded, then plot the cross-seed median. The median return curves confirm the earlier pattern, not achieving sustainable overperformance but rather differing performances due to stochastic path. Equally important, the median Sharpe ratios cluster around the market portfolio's level, indicating that even when training time is curtailed and weight initialisation varies widely, no agent achieves a risk-adjusted edge. In other words, the conclusions drawn from the full-length runs are robust to randomness in both learning time and seed choice.

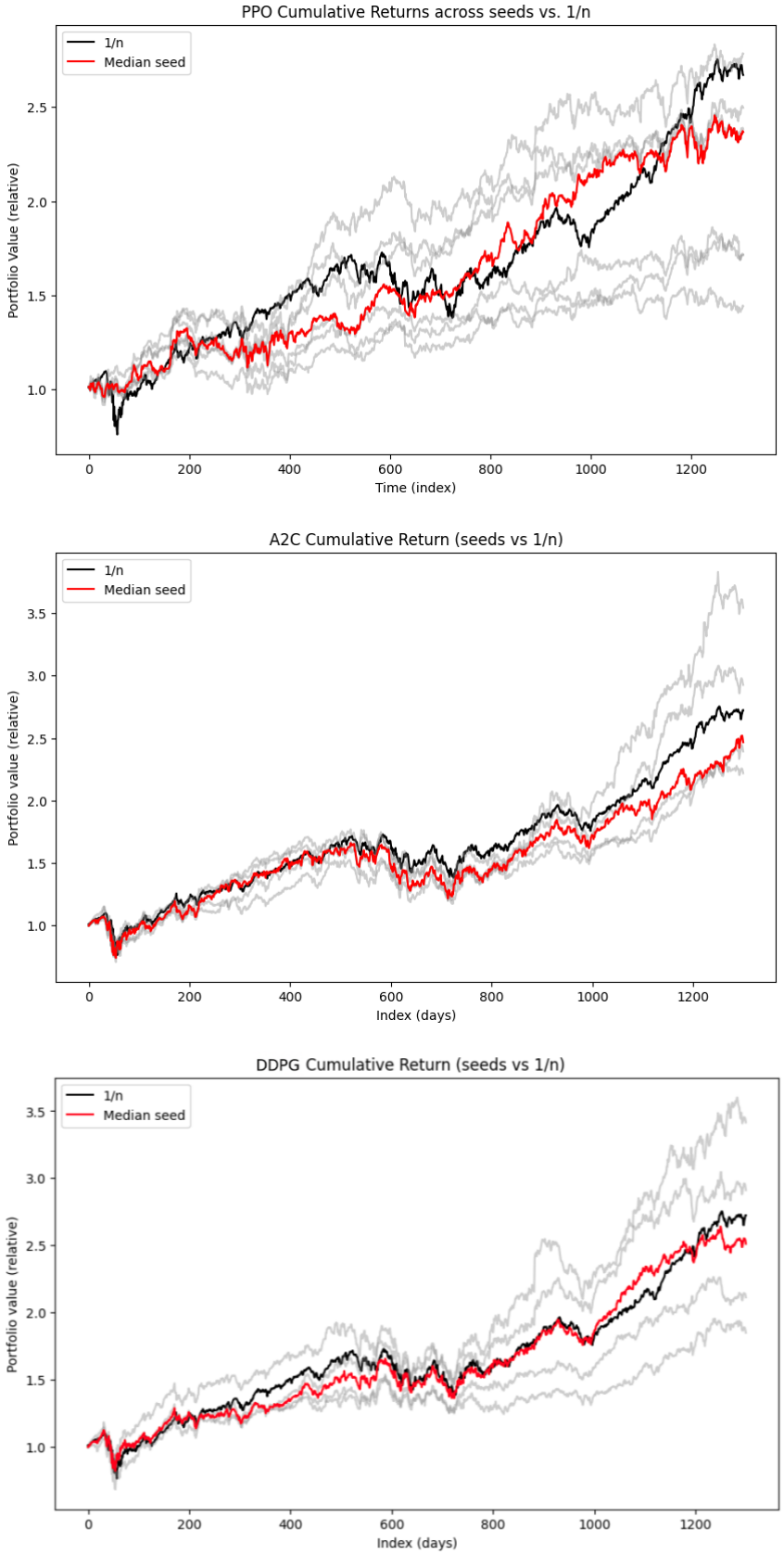

Figure 12: Median portfolio trajectories from multiple random seeds.

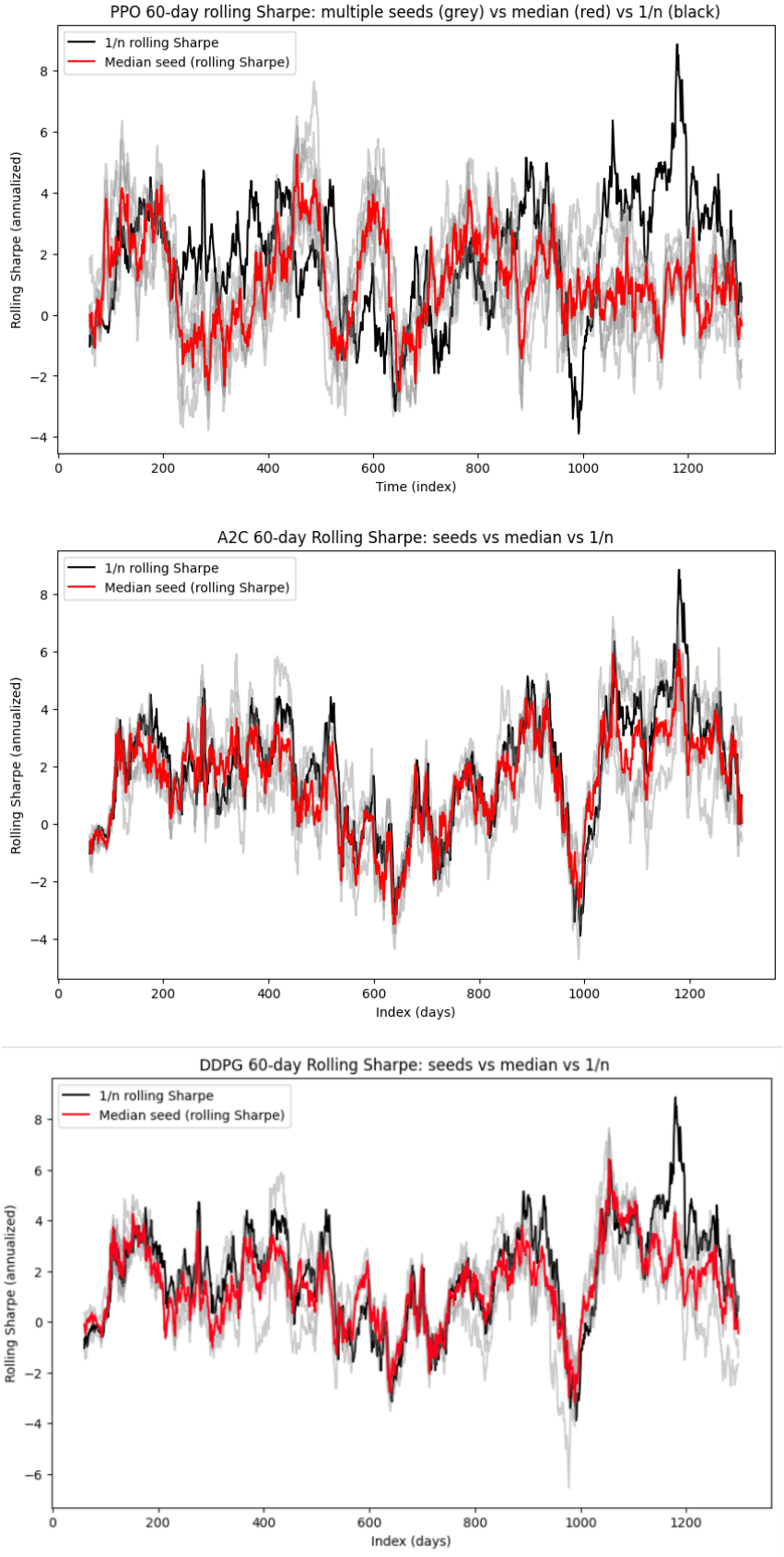

Figure 13: Median annualised Sharpe ratios from multiple seeds.

To summarise the performance metrics discussed so far, Table 7 provides a concise comparison. The benchmark 1/N strategy remains superior on every headline figure: it posts the highest final value, CAGR, and Sharpe ratio, and also the shallowest maximum drawdown. Among the DRL agents, DDPG comes closest to the benchmark in both return and Sharpe (2.55 final value, 19.6 % CAGR, 0.916 Sharpe), albeit with a deeper drawdown (-32.5 %) and the second-highest transaction-cost drag. A2C shows a slightly higher CAGR and Sharpe than PPO but suffers the worst drawdown of the group (-33.4 %), indicating an unstable risk profile despite its relatively low trading cost. PPO delivers the lowest CAGR (17.1 %) and Sharpe (0.805) of the three DRL strategies and a drawdown of –31.9 %, so its supposed stability advantage does not translate into better risk-adjusted returns. Although the raw trade counts in the table put the benchmark at 1 258 executed orders (monthly rebalances of small size) versus, for example 421 for PPO, the cost column reveals the practical picture: the DRL agents incur an order of magnitude higher cumulative transaction costs because their trades tend to be much larger reallocations. These costs help explain why none of the DRL policies convert their tactical activity into a Sharpe ratio that beats the simple equal-weight portfolio.

Given the relatively small performance differentials observed (and the inherent noise in financial returns), it is essential to test whether any apparent advantage of one strategy over another is statistically significant or could simply be due to random chance. This is tested by employing a block–bootstrap hypothesis test by Ledoit and Wolf (2008).

The null hypothesis that each DRL strategy has the same Sharpe ratio as the (1/N) benchmark is tested against the alternative that the Sharpe ratios differ. Because returns are time-dependent and non-normal, a naive analytic test is inappropriate. Instead, the robust procedure of Ledoit and Wolf (2008) is used, which constructs a studentised, time-series bootstrap confidence interval for the difference in Sharpe ratios. If zero is outside this interval (equivalently, if the bootstrap p-value ≤ 0.05), the null hypothesis of equal Sharpe ratios is rejected.

A circular block bootstrap with a block length equal to one month (21 trading days) is implemented to resample paired returns of each strategy and the benchmark. After 1000 bootstrap iterations, none of the DRL strategies shows a significant Sharpe-ratio difference from 1/N at the 5% or even 10% level. For A2C vs. 1/N the p-value is ≈ 0.4670. In this context, the p-value represents the proportion of the 1000 resampled time-series in which a random difference in Sharpe ratios was at least as large as the one originally observed, under the assumption that no true performance difference exists between the strategies. PPO vs. 1/N was even less significant $p \approx 0.5010$ and DDPG vs. 1/N yields $p \approx 0.4740$. Pairwise tests among the DRL agents likewise fail to reach significance at 5 %. Hence, it cannot be concluded that any DRL method out- or underperformed the benchmark in

terms of Sharpe ratio.

The lack of statistically significant differences implies that there is not strong evidence that any DRL strategy truly beats the 1/N portfolio on a risk adjusted basis. This echoes previous literature by (DeMiguel et al., 2009) that finds sophisticated methods often fail to outperform naive diversification out of sample. One reason is sample length: test window covers only a few years, far too short for mean-variance or DRL models to show decisive superiority. Moreover, the agents were trained offline and then tested without further learning causing regime shifts in rates and volatility likely eroded their static policies. In an online learning or meta learning setup, sustained adaptation might enlarge any performance gap, but that lies beyond the current scope.

Finally, it is acknowledged that significance testing in finance has its pitfalls: return distributions are non-normal, and with limited realized path of returns, the power to detect differences is limited. Using block bootstrapping to at least account for time dependence, which is an improvement over naive i.i.d. assumptions. The results from Ledoit and Wolf (2008) method, in particular, gives confidence that if there were a true difference in Sharpe of moderate size, it likely would have detected it. Since it did not, the safest conclusion is that the DRL strategies and the 1/N strategy performed comparably in a statistical sense.

5.4 Interpretation and Critical Discussion

The empirical results provide a rich ground for interpretation. On one hand, the DRL agents demonstrated the ability to adjust and find some profitable opportunities (they all achieved positive returns and average Sharpe ratios near 1, which is respectable in absolute terms). On the other hand, none clearly dominated the simple equal-weight strategy. When considering why the DRL didn't significantly beat 1/N, there are several potential reasons. First, the efficiency of the markets for these large-cap stocks might simply be too high that is consistent with the Efficient Market Hypothesis (EMH) which posits that in a competitive market, it's hard to consistently attain excess risk-adjusted returns. The agents tried to time the market, but any patterns they exploited may have been fleeting or not sufficiently profitable after costs. It's telling that the 1/N, which doesn't attempt any prediction, did as well or better. This suggests there were no easy arbitrages available. Second, estimation error and overfitting likely played a role. The DRL models have many parameters and were trained on a finite sample. They might have "learned" spurious correlations that did not hold up in the test period. For example, A2C's aggressive trades that led to underperformance could be due to it chasing a pattern that turned out to be noise. Overfitting is a notorious issue in quantitative strategies: without extremely robust validation, complex models can fool themselves with in-sample

noise. The true market is always somewhat different from historical data.

The DRL strategies frequently rebalanced in response to market changes, effectively performing a form of dynamic hedging or trend following. For instance, the agents might have been selling during high volatility and buying in calm periods, effectively performing a volatility-timing strategy. This could yield a positive average return, but in a sudden volatility spike it could backfire (some of the big drawdowns). In short, the DRL strategies may be taking on "hidden" risks like tail risk or liquidity risk. This would explain why their Sharpe ratios didn't surpass 1/N whilst overperforming in some time periods, any extra return was compensation for those hidden risks. While 1/N was chosen as the benchmark, one might wonder if the DRL strategies would fare better against other baselines. For example, compared to a marketcap weighted index of these stocks (which would be dominated by the largest companies like AAPL and GOOGL), maybe the DRL performance would look different. If the 1/N portfolio happened to do exceptionally well (it often outperforms cap-weighted indices in diversified sets (DeMiguel et al., 2009, 1930-1933)), then beating 1/N is a high bar. However, the goal was specifically to test against the naive diversification as that is often a tough benchmark. Indeed, the findings reinforce the notion that 1/N is hard to beat consistently.

The practical implementation of these DRL strategies would be challenging due to their high turnover and instability. The weight plots make clear that positions were changing frequently and sometimes dramatically. In reality, such frequent trading could incur market impact costs (which was was not modeled) in addition to the fixed 10 basis point fees. It could also raise questions of whether the strategy could be executed with large capital, would there be enough liquidity to support constant rebalancing among these stocks without slippage. The equal-weight portfolio is straightforward to implement and scale. This highlights a classic trade-off: a complex strategy might promise higher returns on paper, but simplicity often wins in net terms when execution costs and constraints are considered.

It's informative to consider how each DRL algorithm's nature might have influenced its performance. PPO, with its more stable policy updates (via clipping), perhaps learned a more robust strategy: this correlates with its relatively moderate trading count, making big updates to portfolio less frequently. A2C, which updated more aggressively (every 5 steps) and might converge to a local optimum faster, perhaps overfit and ended up with a even more generalizable strategy. DDPG, being an off-policy algorithm, had the ability to learn from replay and might have found a strategy that was a bit different from the on-policy ones. Its performance being intermediate could be due to a combination of factors: it may have benefitted from replay (learning from more data), but as a deterministic policy it might have struggled with exploration, getting stuck in a suboptimal trading

pattern at times. Indeed, the noisiness of DDPG's weights suggests it was tweaking continuously, possibly reflecting the inherent instability of training a continuous control agent in a non-stationary financial environment.

This outcome has several implications for financial theory. The results lend some support to the EMH, particularly its assertion that finding excess risk-adjusted returns in liquid markets is extremely difficult. If markets were highly inefficient, it would be expected that a complex algorithm finds patterns and consistently beats a simplistic strategy. Instead, the DRL agents struggled to surpass 1/N, suggesting that the market pricing of these assets was efficient enough that even nonlinear, dynamic strategies could not easily exploit mispricings. One could argue this is consistent with at least a weak-form EMH: past price patterns alone were not sufficient to guarantee better performance, as the DRL (which has access to historical data and technical features) did not create a winning strategy out-of-sample. While EMH provides one lens, the Adaptive Market Hypothesis offers another perspective that is quite relevant. The results can be interpreted through AMH as follows: The DRL agents possibly did find some inefficiencies during training (patterns that worked in that period), but once deployed in the test period, those patterns may have dissipated or new conditions emerged. Markets in the test period were different, for example the regime shift in interest rates. In an AMH sense, the market adapted, and the agents needed to adapt further. They were not online updated (rolling), so they were somewhat "static" in their learned behavior, which could not fully capitalize on new anomalies. Nevertheless, there was short episodes of outperformance, for example A2C during a certain rally which could hint at temporary inefficiencies that the agent exploited until they vanished. The mixed performance aligns with an AMH view that profit opportunities are episodic: sometimes present, sometimes not. The fact that the agents didn't consistently win might indicate that any edge they had was quickly nullified by market changes or other participants. If one looks at the high turnover and occasional suboptimal trades by the DRL agents, it raises an analogy to human behavioral biases. At times the agents overreacted to short-term fluctuations (not unlike a human trader who chases noise). This wasn't programmed, but emerged from the learning process essentially, the agents can exhibit "behavioral" traits too (like trend chasing or fear based selling) if the reward signal fools them. In a way, the DRL strategies underperformance could hint that they sometimes fell for patterns that weren't truly there, drawing a parallel to how investors fall for hot streaks or panic sell. This is speculative but an interesting cross-link between AI behavior and behavioral finance.

The study also highlights many practical challenges in applying DRL to real-world portfolio management. There was a transaction cost of 0.1 % which is reasonable for retail trading of large-cap stocks, but for institutional scale or for assets with less liquidity, costs can be higher. The DRL strategies, especially with their high turnover, would face signi-

ficant performance erosion from trading costs. Indeed, even that small cost was enough to take away their potential edge. Moreover, the simulation did not include market impact (the price slippage caused by one's own trades), which could be substantial given how often the agents trade. In reality, a strategy that reallocates 20 % of a portfolio daily in large stocks could move the market if done with enough capital.

Study showed that results can vary with hyperparameter choices and due to stochasticity. Even with chosen hyperparameters, these algorithms have inherent variability: two training runs might yield somewhat different policies due to different weight initializations or sampling randomness. This stochasticity can be problematic for a practitioner: one cannot be sure if this particular trained agent is the optimal one or if another training run would do better. In contrast, a fixed strategy like 1/N has no such uncertainty. Additionally, the non-convex optimization in DRL means there's a chance the agent finds a locally optimal but globally suboptimal strategy. Financial data, especially if restricted to a single market or a handful of assets, is limited in quantity and rife with regime changes. Non-stationarity (the fact that the statistical properties of returns change over time) is a deep challenge. A strategy trained on 2010–2018 data might not be ready for 2020's pandemic crash or 2022's rate hikes. The test showed some cracks in the strategies when facing such new events. To use DRL in practice, one might need to continuously retrain or adapt the models, which introduces its own difficulties. This highlights how data limitations constrain what DRL can achieve in finance. With more data or an ability to simulate realistic scenarios, perhaps the agents could learn more robust strategies. Another practical issue is that DRL policies are largely black-box. If a portfolio manager wants to know "Why is the agent investing 50% in JPM today?", it's hard to get a straight answer from the network's weights. This opacity can reduce trust. These insights will be carried into the concluding chapter, summarizing this thesis and proposing what future improvements might be pursued.

6 CONCLUSIONS

This thesis was set out to investigate whether advanced Deep Reinforcement Learning (DRL) techniques can improve upon a simple equal-weight (1/N) portfolio strategy in a quantitative finance setting. Portfolio management was framed as a sequential decision problem with a reward that emphasizes risk adjusted returns under realistic trading frictions. The empirical evidence did not show statistically significant outperformance of the DRL agents over 1/N in the test period. Final wealth and Sharpe ratios were very close across strategies, with the benchmark marginally ahead, and the differences were not significant at conventional levels.

In this setting there is no decisive evidence that DRL improves upon 1/N. These findings are consistent with the view that extracting stable excess risk adjusted returns in liquid equities is difficult. Under a weak form EMH reading, historical and technical features alone were insufficient to deliver persistent gains out of sample. The agents may have exploited patterns during training that did not survive the regime changes in the test period, and the fixed policies used here were not updated online to track those shifts. Practical factors likely contributed to the outcome. Turnover amplifies the effect of transaction costs, even at modest assumed fees. The return generating process is nonstationary, and the quantity of informative data is limited relative to model capacity. Training is stochastic and nonconvex, which complicates stability and reproducibility. The opacity of neural policies also poses challenges for oversight and governance compared with transparent rules such as 1/N.

The main limitations of the study are a small asset universe, static rather than online policies, a restricted feature set, and simplified execution that does not model market impact or slippage. These choices support clarity and comparability but they constrain external validity. Answering the research question, DRL did not deliver superior risk adjusted performance to 1/N in the experiments. Equal weighting remains a very demanding benchmark, consistent with prior literature such as DeMiguel et al. (2009).

Future work could prioritize continual or online adaptation to regime change, explicit risk objectives and constraints including drawdown and CVaR, richer and economically motivated features with broader universes and controlled use of shorting and leverage, realistic execution with impact and turnover regularization during training, and stronger interpretability and benchmarking against factor and rule based strategies. Naive diversification remains robust, and whether DRL can consistently surpass it likely depends on progress along these fronts.

References

Almahdi, S. – Yang, S. Y. (2017) An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, vol. 87, 267–279.

Bai, Y., Gao, Y., Wan, R., Zhang, S. – Song, R. (2024) A review of reinforcement learning in financial applications. *Annual Review of Statistics and Its Application*, vol. 12.

Barberis, N. – Thaler, R. (2003) A survey of behavioral finance. *Handbook of the Economics of Finance*, vol. 1, 1053–1128.

Betancourt, C. – Chen, W.-H. (2021) Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. *Expert Systems with Applications*, vol. 164, 114002.

Beysolow II, T. (2019) Market making via reinforcement learning. In *Applied Reinforcement Learning with Python: With OpenAI Gym, Tensorflow, and Keras*, 77–94, Springer.

Black, F. (1986) Noise. *The journal of finance*, vol. 41 (3), 528–543.

Bouchaud, J.-P., Ciliberti, S., Lempérière, Y., Majewski, A., Seager, P. – Ronia, K. S. (2017) Black was right: Price is within a factor 2 of value. *arXiv preprint arXiv:1711.04717*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. – Zaremba, W. (2016) Openai gym. In *Proceedings of Deep Reinforcement Learning Workshop, NIPS*.

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

Carhart, M. M. (1997) On persistence in mutual fund performance. *The Journal of finance*, vol. 52 (1), 57–82.

DeMiguel, V., Garlappi, L. – Uppal, R. (2009) Optimal versus naive diversification: How inefficient is the 1-n portfolio strategy? *The Review of Financial Studies*, vol. 22 (5), 1915–1953, URL: `https://EconPapers.repec.org/RePEc:oup:rfinst:v:22:y:2009:i:5:p:1915-1953`.

Fama, E. F. (1965) The behavior of stock-market prices. *The journal of Business*, vol. 38 (1), 34–105.

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

Fujimoto, S., Hoof, H. – Meger, D. (2018) Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*.

Glorot, X., Bordes, A. – Bengio, Y. (2011) Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323, JMLR Workshop and Conference Proceedings.

Gu, S., Kelly, B. – Xiu, D. (2020) Empirical asset pricing via machine learning. *The Review of Financial Studies*, vol. 33 (5), 2223–2273, URL: `https://doi.org/10.1093/rfs/hhaa009`.

Guan, M. – Liu, X.-Y. (2021) Explainable deep reinforcement learning for portfolio management: an empirical approach. In *Proceedings of the second ACM international conference on AI in finance*, 1–9.

Huang, G., Zhou, X. – Song, Q. (2020) Deep reinforcement learning for long-short portfolio optimization. *arXiv e-prints*, arXiv–2012.

Huang, G., Zhou, X. – Song, Q. (2022) Deep reinforcement learning for portfolio management. URL: `https://arxiv.org/abs/2012.13773`.

Jeong, G. – Kim, H. Y. (2019) Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, vol. 117, 125–138.

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

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

Jones, S. L. – Netter, J. M. (2008) Efficient capital markets. *The Concise Encyclopedia of Economic*, vol. 15 (14), 87–98.

Lecun, Y., Bengio, Y. – Hinton, G. (2015) Deep learning. *Nature*, vol. 521 (7553), 436–444, publisher Copyright: © 2015 Macmillan Publishers Limited. All rights

reserved.

Ledoit, O. – Wolf, M. (2008) Robust performance hypothesis testing with the sharpe ratio. *Journal of Empirical Finance*, vol. 15 (5), 850–859.

Lee, J., Kim, R., Yi, S.-W. – Kang, J. (2020) Maps: Multi-agent reinforcement learning-based portfolio management system. *arXiv preprint arXiv:2007.05402*.

Liang, Z., Chen, H., Zhu, J., Jiang, K. – Li, Y. (2018) Adversarial deep reinforcement learning in portfolio management. URL: `https://arxiv.org/abs/1808.09940`.

Lillicrap, T. P. et al. (2015) Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lindner, T., Puck, J. – Verbeke, A. (2022) Beyond addressing multicollinearity: Robust quantitative analysis and machine learning in international business research. *Journal of International Business Studies*, vol. 53 (7), 1307–1314.

Liu, X.-Y., Xiong, Z., Zhong, S., Yang, H. – Walid, A. (2018) Practical deep reinforcement learning approach for stock trading. *arXiv preprint arXiv:1811.07522*.

Lo, A. W. (2004) The adaptive markets hypothesis: Market efficiency from an evolutionary perspective. *Journal of Portfolio Management, Forthcoming*.

Malkiel, B. G. (2003) The efficient market hypothesis and its critics. *Journal of economic perspectives*, vol. 17 (1), 59–82.

Markowitz, H. (1952) Portfolio selection. *The Journal of Finance*, vol. 7 (1), 77–91, URL: `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1952.tb01525.x`.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. et al. (2015) Human-level control through deep reinforcement learning. *nature*, vol. 518 (7540), 529–533.

Mohammed, S., Bealer, R. – Cohen, J. (2021) Embracing advanced ai/ml to help investors achieve success: Vanguard reinforcement learning for financial goal planning. *arXiv preprint arXiv:2110.12003*.

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

Musiol, M. (2016) Speeding up deep learning computational aspects of machine learning.

Nevmyvaka, Y., Feng, Y. – Kearns, M. (2006) Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, 673–680.

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

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. – Chintala, S. (2019) Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Pendharkar, P. C. – Cusatis, P. (2018) Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, vol. 103, 1–13.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, N. – Dormann, P. (2021) Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, https://github.com/DLR-RM/stable-baselines3.

Safari, S. A. – Schmidhuber, C. (2025) Trends and reversion in financial markets on time scales from minutes to decades. *arXiv preprint arXiv:2501.16772*.

Schulman, J., Moritz, P., Levine, S., Jordan, M. – Abbeel, P. (2015) High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. – Klimov, O. (2017) Proximal policy optimization algorithms. URL: `https://arxiv.org/abs/1707.06347`.

Schwert, G. W. (2003) Anomalies and market efficiency. *Handbook of the Economics of Finance*, vol. 1, 939–974.

Sharpe, W. F. (1964) Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, vol. 19 (3), 425–442.

Shen, W., Wang, J., Jiang, Y.-G. – Zha, H. (2015) Portfolio choices with orthogonal bandit learning. In *IJCAI*, vol. 15, 974–980.

Shiller, R. J. (2017) Narrative economics. *The American Economic Review*, vol. 107 (4), 967–1004, URL: `http://www.jstor.org/stable/44251584`.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. – Riedmiller, M. (2014) Deterministic policy gradient algorithms. In *International conference on machine learning*, 387–395, Pmlr.

Sola, J. – Sevilla, J. (1997) Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, vol. 44 (3), 1464–1468.

Soleymani, F. – Paquet, E. (2021) Deep graph convolutional reinforcement learning for financial portfolio management–deeppocket. *Expert Systems with Applications*, vol. 182, 115127.

Sutton, R. S., Barto, A. G. et al. (1998) *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge.

Sutton, R. S., Barto, A. G. et al. (2018) Reinforcement learning: An introduction 2nd ed. *MIT press Cambridge*, vol. 1 (2), 25.

Tian, Y., Gao, M., Gao, Q. – Peng, X.-H. (2024) Trading in fast-changing markets with meta-reinforcement learning. *Intelligent Automation & Soft Computing*, vol. 39 (2).

Uhlenbeck, G. E. – Ornstein, L. S. (1930) On the theory of the brownian motion. *Phys. Rev.*, vol. 36, 823–841, URL: `https://link.aps.org/doi/10.1103/PhysRev.36.823`.

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

Wang, Z., Huang, B., Tu, S., Zhang, K. – Xu, L. (2021) Deeptrader: a deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 643–650.

Watkins, C. J. – Dayan, P. (1992) Q-learning. *Machine learning*, vol. 8, 279–292.

Williams, R. J. (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, vol. 8, 229–256.

Wu, M.-E., Syu, J.-H., Lin, J. C.-W. – Ho, J.-M. (2021) Portfolio management system in equity market neutral using reinforcement learning. *Applied Intelligence*, vol. 51 (11), 8119–8131.

Xiong, Z., Liu, X.-Y., Zhong, S., Yang, H. – Walid, A. (2018) Practical deep reinforcement learning approach for stock trading. *arXiv preprint arXiv:1811.07522*, vol. 25, 27.

Yang, H., Liu, X.-Y., Zhong, S. – Walid, A. (2020) Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance*, 1–8.

Ye, Y., Pei, H., Wang, B., Chen, P.-Y., Zhu, Y., Xiao, J. – Li, B. (2020) Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 1112–1119.

Zhang, Z., Zohren, S. – Roberts, S. (2019) Deep reinforcement learning for trading. *arXiv preprint arXiv:1911.10107*.

Zou, J., Lou, J., Wang, B. – Liu, S. (2024) A novel deep reinforcement learning based automated stock trading system using cascaded lstm networks. *Expert Systems with Applications*, vol. 242, 122801.

Appendices

APPENDIX I

Algorithm 1: Advantage Actor-Critic (A2C)

Initialize policy network $\pi_\theta(s)$ and value network $V_\phi(s)$ with random weights.
for iteration = 1 to M (M updates) **do**
  for each environment worker $k = 1$ to K (in parallel) **do**
    Run policy π_θ for n steps or until episode ends:
    Collect states s_t, actions a_t, rewards r_t for $t = 1, \ldots, n$.
    if episode ended at step n **then**
      Set $R = 0$
    else
      Set $R = V_\phi(s_{n+1})$ {bootstrap from last state}
    end if
    for $t = n$ down to 1 **do**
      $R \leftarrow r_t + \gamma R$
      Calculate advantage: $A_t \leftarrow R - V_\phi(s_t)$
      Accumulate gradients: $g_\theta \leftarrow g_\theta + \nabla_\theta[-\log \pi_\theta(a_t|s_t)A_t]$ {policy gradient (maximize advantage)}
      Accumulate gradients: $g_\phi \leftarrow g_\phi + \nabla_\phi[A_t^2]$ {value gradient (MSE loss)}
    end for
  end for
  $\theta \leftarrow \theta + \alpha_\theta \frac{g_\theta}{K}$ {update policy weights; ascent on advantage}
  $\phi \leftarrow \phi - \alpha_\phi \frac{g_\phi}{K}$ {update value weights; descent on loss}
end for

APPENDIX II

Algorithm 2: Deep Deterministic Policy Gradient (DDPG)

Initialize actor network $\mu_\theta(s)$, critic network $Q_\phi(s, a)$ with random weights
Initialize target networks: $\theta' \leftarrow \theta$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{D}$
for episode = 1 to E **do**
  Receive initial state s_0
  for $t = 0$ to $T - 1$ (until end of episode or max steps) **do**
    // Actor selects action with exploration noise
    $a_t = \mu_\theta(s_t) + \text{noise}_t$ {noise $\sim \mathcal{N}(0, \sigma)$ or Ornstein-Uhlenbeck}
    Execute action a_t, observe reward r_t and new state s_{t+1}
    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$
    $s_t \leftarrow s_{t+1}$
    // Update step (every step or every few steps)
    Sample minibatch of B experiences (s_i, a_i, r_i, s'_i) from $\mathcal{D}$
    for each sample i in batch **do**
      $y_i = r_i + \gamma Q_{\phi'}(s'_i, \mu_{\theta'}(s'_i))$ {compute target Q}
    end for
    Update critic by minimizing loss:

$$L = \frac{1}{B} \sum_i (Q_\phi(s_i, a_i) - y_i)^2$$

    Update actor using deterministic policy gradient:

$$\nabla_\theta J \approx \frac{1}{B} \sum_i \nabla_\theta \mu_\theta(s_i) \nabla_a Q_\phi(s_i, a)|_{a=\mu_\theta(s_i)}$$

    Perform gradient ascent on θ (e.g., Adam optimizer)
    Soft update target networks:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta', \quad \phi' \leftarrow \tau\phi + (1 - \tau)\phi'$$

  end for
end for

APPENDIX III

Algorithm 3: Proximal Policy Optimization (PPO)

Initialize policy network $\pi_\theta(s)$ and value network $V_\phi(s)$
for iteration $= 1$ to N_{iter} **do**
  Collect set of trajectories $\mathcal{D} = \{\tau\}$ by running policy π_θ in environment (with T steps, possibly parallel environments)
  Compute rewards-to-go R_t and advantage estimates $\hat{A}_t$ for each timestep in $\mathcal{D}$ (using current V_ϕ, e.g., via GAE)
  Save current policy parameters as $\theta_{\text{old}} \leftarrow \theta$
  for epoch $= 1$ to K (multiple epochs per batch) **do**
    for each minibatch $M \subset \mathcal{D}$ **do**
      Compute ratios:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, \quad \forall (s_t, a_t) \in M$$

      Compute policy loss:

$$L_{\text{pg}} = -E_M \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

      Compute value loss:

$$L_{\text{vf}} = E_M \left[(V_\phi(s_t) - R_t)^2 \right]$$

      Compute entropy bonus (encourage exploration):

$$L_{\text{ent}} = -E_M[\beta\, H(\pi_\theta(s_t))]$$

      Compute total loss:

$$L = L_{\text{pg}} + c_1 L_{\text{vf}} + c_2 L_{\text{ent}}$$

      Update θ, ϕ by gradient descent/ascent on total loss
    end for
  end for
end for

APPENDIX IV

Algorithm 4: Simplified Pseudocode for the StockTradingEnv

Input: Price data, feature data, macro/fundamental data, and config parameters.

// — Environment Initialization —
Store data arrays and configuration parameters.
Define continuous action space $\mathcal{A}$ for portfolio weights.
Define observation space $\mathcal{S}$ based on the concatenated dimensions of all input data over the lookback window.

// — Reset Function Logic —
function Reset()
    Set ‘current_step‘ to its initial value (‘window_size‘ - 1).
    Set ‘portfolio_value‘ to 1.0.
    Set ‘current_weights‘ to the initial distribution (e.g., $1/N$).
    ‘observation‘ ← Construct observation for the initial state.
    return ‘observation‘.
end function

// — Step Function Logic —
function Step(‘action‘)
    ‘previous_weights‘ ← ‘current_weights‘.
    ‘target_weights‘ ← Normalize ‘action‘ so that weights sum to 1.
    ‘turnover‘ ← $\sum$|target_weights − previous_weights|.
    ‘transaction_cost‘ ← ‘portfolio_value‘ × ‘turnover‘ × ‘cost_rate‘.

    // Calculate return based on weights held during the period
    ‘asset_returns‘ ← Get returns for the current step from price data.
    ‘portfolio_return‘ ← ‘previous_weights‘ · ‘asset_returns‘.

    // Update value and compute reward
    ‘previous_value‘ ← ‘portfolio_value‘.
    ‘portfolio_value‘ ← ‘portfolio_value‘ × (1 + ‘portfolio_return‘) -‘cost‘.
    ‘reward‘ ← log(portfolio_value/previous_value).

    // Transition to the next state
    ‘current_weights‘ ← ‘target_weights‘.
    ‘current_step‘ ← ‘current_step‘ + 1.
    ‘done‘ ← (‘current_step‘ ≥ ‘max_steps‘).
    ‘next_observation‘ ← Construct observation for the new state.
    return ‘next_observation‘, ‘reward‘, ‘done‘.
end function