Optimizing Quantitative Trading: An Experimental Study of DQN Trading Strategies and Utility Functions

Mohammad Bagheri
Student number: 2044174

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Quantitative Finance & Actuarial Sciences
School of Economics and Management
Tilburg University

Thesis committee:

dr. Nikolaus Schweizer
prof dr. Bertrand Melenberg

Tilburg University
Tilburg, The Netherlands
August 2024

Abstract

Machine learning has revolutionized financial trading by enhancing traditional methods and introducing new strategies. This thesis explores the application of Deep Q-Networks (DQNs), a reinforcement learning model, for quantitative trading. Developed by Google DeepMind, DQNs have demonstrated human-expert level performance in Atari games by combining Q-learning with deep convolutional neural networks, effectively learning complex policies from high-dimensional inputs. This research focuses on developing a DQN-based trading system to optimize trading performance across 20 highly liquid futures contracts in four asset classes: commodities, equities, fixed income, and foreign exchange. The study evaluates DQN agents using different utility functions (linear and logarithmic) and position sizing rules (volatility targeting and a novel volatility targeting with momentum scaling). Performance is assessed based on risk-adjusted returns, profitability, and drawdown management. Key findings indicate that DQN agents significantly outperform traditional baseline models, including passive buy-and-hold strategies aligned with the Efficient Market Hypothesis and active technical trading strategies. DQN agents exhibit superior resilience during market disruptions, such as the COVID-19 pandemic and the Russian-Saudi oil price war, with logarithmic utility functions generally yielding better results. This thesis contributes to quantitative finance by showcasing the potential of reinforcement learning based strategies in enhancing trading performance. It provides insights into effective utility functions and position sizing rules, and highlights the operational practicality of these models, considering transaction costs. The results could serve as a stepping stone toward building more robust trading systems using reinforcement learning, paving the way for future research in this field.

Contents

1 Introduction

In recent decades, advances in machine learning have revolutionized many fields, including financial trading [Bahoo et al., 2024; Singh et al., 2023]. Machine learning models are used to analyze large datasets, identify patterns, and make predictions, which are highly valuable in financial trading. Traditionally, financial trading has utilized various theories and methods, including short-term technical analysis, time-series forecasting, cross-sectional strategies, and fundamental analysis, which assesses a company's financial health [Fama and French, 1992, 1996; Baz et al., 2015; Nti et al., 2020]. These approaches are essential for both individual passive income generation and broader financial applications such as pension fund management, company investment portfolios, and risk hedging. With the evolution of technology and data science, financial trading has increasingly incorporated quantitative methods, leading to the rise of quantitative trading, often referred to as algorithmic trading. Quantitative trading uses advanced decision-making systems to perform trades with a higher frequency and more systematically than human traders. Human trading can be biased and emotional [Lad and Tailor, 2016; Kusev et al., 2017], leading to irrational decisions, whereas algorithmic trading is designed to be more objective and consistent. Quantitative trading now accounts for a substantial portion of trading volume in various markets, including stocks and commodities, and is a crucial tool for hedge funds and asset managers seeking to outperform the market [Kunz and Martin, 2015]. Conversely, some believe in market efficiency, arguing that consistent outperformance of the market is not feasible because all available information is reflected in prices [Fama, 1965; Malkiel, 2003].

Machine learning, especially supervised learning, has pushed the boundaries of what was once considered impossible. It has achieved notable success in fields such as image classification and natural language processing [Krizhevsky et al., 2012; Vaswani et al., 2017; Devlin et al., 2019]. For financial trading, supervised learning is increasingly used to optimize traditional methods and predict market movements [Lim et al., 2019]. These models can analyze large volumes of financial data to identify patterns and trends that may not be noticeable to human traders. However, translating these predictions into effective trading strategies that take into account risks, preferences, and transaction costs remains challenging. This is mainly because supervised learning-based trading follows a two-step approach: first, making predictions about the price movements, and then inputting the predictions into a trading system that executes trades. The prediction models often do not consider market liquidity and frictions, which can significantly impact the effectiveness of the trading strategies.

Reinforcement learning (RL), another subset of machine learning, could potentially address some of these challenges [Moody and Saffell, 1998; Shakya et al., 2023]. Unlike supervised learning, RL involves training an agent through interactions with an environment, learning to maximize rewards through trial and error, inspired by human learning. This approach can be particularly useful in financial trading, where the agent can learn to make trade decisions that maximize long-term rewards while considering various market conditions and risks. Deep Q-Networks (DQNs), an RL model developed by researchers at Google DeepMind, notably by Mnih et al. [2013], gained fame in 2014 for their ability to achieve human-expert level performance in Atari games. They researched the application of deep learning to reinforcement learning, with the aim of creating agents capable of learning intricate policies directly from high-dimensional (sensory) inputs such as images. Their experiments demonstrated that DQN could learn to play various Atari games at a human-expert level, achieving high scores and mastering diverse game mechanics without any game-specific knowledge. This breakthrough showcased the potential of deep reinforcement learning in solving complex tasks. Financial trading, in essence, can be seen as a game, where the objective is to make decisions that lead to the highest possible rewards, much like an agent in an Atari game optimizing its play to achieve the highest score.

Despite the potential of RL, its application in quantitative finance is still limited and under-researched compared to supervised learning. Most existing RL models for financial trading assume risk-neutral agents, as the reward system of the agents is primarily based on the profits made [Zhang et al., 2019; Théate and Ernst, 2021]. However, this assumption can lead to behavioral flaws. Humans are typically risk-averse and require a risk premium for taking on additional risk. Risk-neutral agents, on the other hand, tend to take excessive risks and often disregard potential losses. This disregard for potential losses arises because risk-neutral agents evaluate outcomes purely based on expected returns, without accounting for the variability or uncertainty of those returns. They are indifferent to the distribution of returns, focusing solely on maximizing expected profit. This behavior is undesirable for investors, who prioritize stability and downside protection. Therefore, it is crucial to penalize high volatility, particularly downside risk, in the reward structure of RL models to better align with investor preferences. Furthermore, the importance of money management, such as determining position sizes, is often overlooked, although it significantly impacts trading performance and portfolio risk profiles [Moody and Saffell, 1998; Scholz, 2012]. Incorporating these considerations into RL models can lead to more robust and desirable trading strategies.

This thesis aims to explore and extend the application of RL in quantitative finance, focusing on Deep Q-Networks (DQNs), for quantitative trading. The main objectives are:

1. **Build a Reinforcement Learning Trading System:** Develop a DQN-based trading system for quantitative trading.
2. **Investigate Reward Functions:** Compare DQN agents using reward designs based on both linear and logarithmic utility functions to account for risk preferences.
3. **Extend Position Sizing Techniques:** Combine volatility scaling with a trend-based position sizing rule to create a novel ensemble position sizing strategy.
4. **Perform Market Comparisons:** Evaluate the DQN agents across various markets (commodities, equities, fixed income, foreign exchange) against passive strategies and technical trading benchmarks.

This research involves developing a DQN network for trading single assets and evaluating it using different utility functions and position sizing rules. The strategies are applied to 20 futures contracts across four asset classes. Performance is assessed based on three components: risk-adjusted returns, profitability, and drawdown management, and compared to traditional baseline models.

The findings reveal that the constructed DQN agents significantly outperform traditional baseline models, including the passive buy-and-hold strategies and active technical trading strategies, across most asset classes except fixed income. DQN agents demonstrate better resilience during market disruptions, such as the COVID-19 pandemic and the Russian-Saudi oil price war. Agents using logarithmic utility functions generally outperform those using linear utility functions. Position sizing strategies significantly impact performance, and trading cost analysis highlights the operational practicality and limitations of these models. This research contributes to quantitative finance by demonstrating the potential of DQN-based strategies in optimizing trading performance. It provides insights into the impact and effectiveness of different utility functions and position sizing rules, offering practical implications for traders. The study also highlights the resilience of DQN agents during market disruptions and their sensitivity to transaction costs.

This thesis is organized as follows: the background and related work section discusses theoretical foundations and relevant studies, serving as the motivation for the methodology. The methodology section details the development of the DQN model, including aspects of the RL agent

such as its state-space, action-space, reward design, and the architecture of the reinforcement learning model. The experimental setup section defines the data and systematic approach for performing experiments and evaluating the performance of RL agents versus baseline models. The results section presents empirical findings, including comparisons between utility functions and position sizing rules, crisis analysis, and transaction cost sensitivity analysis. Finally, the conclusion summarizes key insights and is followed by a discussion of limitations and future research directions.

2 Background & Related Work

In this section, we discuss the background and related work pertinent to the objectives and scope of this thesis. We begin by introducing the Efficient Market Hypothesis (EMH) and the various types of analyses employed in financial trading. Following this, we explain the concept of quantitative trading and the principles of supervised learning. This leads into a discussion on reinforcement learning, highlighting its relevance and motivation for our focus within the context of quantitative trading. Subsequently, we review the literature related to this thesis and provide the rationale behind the methodology we will use. Additionally, we discuss utility theory and position sizing, as these are core aspects that underpin our development of reinforcement learning models. Based on our review of the literature, we aim to develop a reinforcement learning model and conduct experiments that, to the best of our knowledge, offer new insights not currently found in existing literature.

2.1 Efficient Market Hypothesis

The Efficient Market Hypothesis (EMH), formulated by Fama [1965], claims that financial markets are 'efficient', entailing that asset prices incorporate and reflect all the relevant information available at any given time. Consequently, it is nearly impossible to consistently profit from mispriced assets through either fundamental or technical analysis without taking on additional risk, as all pertinent information is already blended into the prices. EMH applies to nearly all financial assets, including stocks, commodities, fixed income, and foreign exchange. EMH has three forms: weak, semi-strong, and strong. The weak form entails that all past trading information is reflected in the prices. The semi-strong form hypothesizes that all publicly available information is reflected in the prices. The strong form claims that all information, both public and private, is incorporated into prices, meaning that no type of information can give investors an edge in outperforming the market.

EMH is a central theory in financial economics and has received substantial support as well as its share of criticism. Malkiel [2003] argues that a long-term passive buy-and-hold strategy is optimal for average investors. According to Malkiel, attempting to time the market or pick individual stocks tends to be ineffective in the long run, aligning with the notion of EMH. Empirical evidence suggests that stock prices follow a random walk, indicating that changes in prices are unpredictable and consistent with the weak form of EMH. Studies by Kendall and Hill [1953] and subsequent research by Fama [1970] demonstrated empirically that stock prices move randomly, reinforcing the idea that historical price data cannot predict future prices. Additionally, in developed markets like Europe and the U.S., the semi-strong form of EMH is well-supported [Solnik, 1973]. Event studies have shown that newly publicly available information, such as firm earnings statements, is rapidly incorporated into stock prices, preventing investors from achieving abnormal (excess) returns through fundamental analysis [Patell and Wolfson, 1984]. Moreover, another convincing support for EMH is the fact that active fund managers generally perform poorly compared to passive index funds over the long term [Carhart, 1997; Malkiel, 1995]. This aligns with the EMH, suggesting that markets are efficient enough that even active fund man-

agers with access to more private data cannot effectively achieve abnormal gains.

However, EMH has faced significant criticism, particularly from the field of behavioral finance, which challenges the assumption of EMH that market participants are rational, which is also a crucial assumption of modern portfolio theory [Markowitz, 1952]. Critics argue that EMH fails to account for irrational behavior and cognitive biases that can influence investor decisions and market prices. Economists have provided evidence of anomalies and market inefficiencies, such as the January effect (a seasonal increase in equity prices at the beginning of the year), momentum (the tendency of well-performing assets to carry on performing well and poorly performing assets to continue underperforming), and financial crises like the dotcom bubble and the 2008 financial crisis. These phenomena stem from irrational investor behavior driven by emotions and cognitive biases, which contradict the notion of EMH [Shiller, 2003; Kahneman and Tversky, 1979]. Even Malkiel, an advocate of EMH, acknowledges market inefficiencies and anomalies but argues that they are difficult to exploit systematically [Malkiel, 2003]. Furthermore, Fama himself has published multiple articles addressing behavioral finance critiques, presenting a balanced view on the debate between EMH and behavioral finance [Fama, 1998]. In his work with Kenneth French, they empirically found that stocks with small market capitalizations and high book-to-market ratios consistently outperformed the market, suggesting these factors were not fully accounted for by the market [Fama and French, 1992]. Another famous argument against EMH is the presence of noise trader risk [Long et al., 1990]. Noise traders, who make decisions based on irrelevant information, can cause prices to diverge from their fundamental values and lead to market inefficiencies.

2.2 Fundamental and Technical Analysis

To further explore the foundations of financial trading, it is essential to understand two types of analysis: fundamental and technical analysis, which form the basis of many trading strategies.

Fundamental analysis is a methodology for evaluating the intrinsic value of a financial asset, such as a stock, by examining relevant macroeconomic factors, industry of a company, financial, and non-financial factors of a company [Abarbanell and Bushee, 1997]. Economic factors include macroeconomic conditions such as GDP growth rate, interest rates, inflation levels, and unemployment rates, which help assess the overall health of the economy. Industry factors involve assessing the dynamics within the industry of the given company, including the regulatory environment, market trends, and competition. Financial factors involve evaluating the company's financial stability by analyzing its balance sheet, cash flow statement, growth potential, and dividend payments, which provide insights into its profitability, liquidity, and solvency. Non-financial factors focus on the company's reputation and Environmental, Social, and Governance (ESG) status, which refers to how its products and services contribute to sustainable development. The primary goal of fundamental analysis is to determine the fundamental value of an asset, which can then be compared to its market value to decide whether the asset is correctly priced. If the stock is undervalued, one might go long, and if it is overvalued, one might short the stock to make a profit. Fundamental analysis is mostly used for long-term investment strategies, providing a deep understanding of the underlying value of an asset, but it is considered less useful for short-term trading due to market inefficiencies in the short term. Lev and Thiagarajan [1993] reveal that fundamental data of a company adds approximately 70% to the predictive power of earnings with respect tot excess returns.

Alternatively, technical analysis uses historical price and volume data to forecast future market movements. Technical analysts believe that past trading activity and price changes have predictive power over future price movements, challenging the notion of the Efficient Market Hypothesis, which states that markets are informationally efficient and that prices are unpre-

dictable. Technical analysis strategies rely on technical indicators such as moving averages to smooth noisy price data and identify trends, or relative strength indices to determine when an asset is overbought or oversold. They also analyze volume levels to measure the strength of a certain movement and find support and resistance levels to mitigate risks by identifying price ranges within which an asset is likely to stay in the short term. In general, the primary objective is to identify trends and patterns that occur regularly to make profitable trades. Technical analysis is mostly used by short-term traders and market participants who seek to capitalize on market inefficiencies in the short term.

A study by Schulmeister [2005] examined the profitability of more than 1000 technical trading strategies and found that all of them would have been lucrative in the dollar market between 1973 and 1999, and 91.7% remained profitable in the out-of-sample period from 2000 until 2004. Interestingly, they found that the profitability of the trading models was due to their ability to exploit lasting trends and the aggregate trading behavior of the technical models exercised significant demand pressure, with most models taking the same long or short positions, which in turn made the trends last longer. Moreover, Baz et al. [2015] conducted an empirical study in which they looked at both fundamental value and momentum-based technical trading strategies using the moving average convergence divergence (MACD) indicator, finding that the strategies were generally profitable and worked best when combined. Kwon and Kish [2002] found that technical trading rules based on moving averages and trading volume were able to produce greater profits than the Buy & Hold strategy from 1962 until 1996 on the New York Stock Exchange. Lastly, Park and Irwin [2007] conducted a survey of various studies to explore the effectiveness of technical trading strategies and found that 56 out of 95 studies reported positive findings regarding technical trading strategies in various markets such as the stock market, foreign exchange, and commodity markets.

2.3 Quantitative Trading and Supervised Learning

Building on the principles of market analysis, quantitative trading leverages statistical models and algorithms to make trading decisions, integrating insights from either fundamental or technical analysis or both. Fundamental analysis evaluates a security's intrinsic value, while technical analysis examines past market data to identify patterns and trends. Quantitative trading systematically exploits market inefficiencies by utilizing large datasets and algorithmic strategies, enhancing the accuracy and speed of trading decisions. Traditionally, time-series strategies for trading relied on statistical methods such as the Autoregressive Moving Average (ARMA) model and the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model, which were used for quantitative trading. However, due to the non-linear, non-stationary, and noisy nature of financial market time-series data, these statistical methods suffer from their linearity assumptions [Huang et al., 2018].

Nowadays, most quantitative trading models are based on modern supervised learning algorithms, which do not have the same linearity limitations as traditional statistical methods. The goal of supervised learning is to train models on historical data to make predictions about a financial asset [Wang and Yan, 2021]. For example, a supervised learning model might predict the future price of a stock based on historical prices and other variables such as macroeconomic conditions, technical indicators, and company fundamentals. Once trained by minimizing the forecasting error with respect to the training data, the model makes predictions on new unseen data. These predictions are then fed into a trading regimen, where they are translated into trading decisions such as longing, shorting, or holding the asset. Common supervised learning models include linear regression, support vector machines (SVM), tree-based models, and neural networks (NN). An example of an advanced supervised learning application in trading is provided by Lim et al. [2019], who introduced an innovative approach to traditional momentum time series strategies

where they developed a model that leverages deep neural networks for generating trading signals. This model simultaneously learns trends and determines position sizing in a data-driven manner, focusing on optimizing risk-adjusted returns. The results showed that their model significantly outperformed both traditional technical trading benchmarks and passive buy-and-hold strategies in terms of risk-adjusted returns across various asset classes.

Supervised learning, while powerful, has its limitations in quantitative trading. One limitation is the two-step approach: the model first outputs a prediction about future price movements, which then needs to be translated into trade decisions. However, minimizing forecast error does not necessarily align with the main objective of the investor, which is maximizing risk-adjusted returns [Moody and Saffell, 1998]. Additionally, predictions from the model serve as inputs to the trading system, but factors such as liquidity and transaction costs are not directly considered by the learning model and should be tackled in the trading system itself.

2.4 Reinforcement Learning

Reinforcement learning (RL) addresses some of the limitations of supervised learning for quantitative trading by enabling trading algorithms to learn optimal strategies through either direct interaction with the market environment (model-free RL) or through a model of the environment (model-based RL). These algorithms continuously improve their performance over time through trial and error [Fischer, 2018]. RL is a type of sequential learning in which the agent learns to make decisions by taking actions in an environment and receiving numerical rewards, with the objective of maximizing cumulative rewards. This method is particularly useful in trading because of its dynamic decision-making capabilities. The two-step approach of supervised learning is combined into one system in RL, often relying on supervised learning to predict actions that maximize numerical rewards [Sutton and Barto, 2018]. These rewards can be designed to align with investor objectives; for example, the risk appetite of an investor can be taken into account by using, for example, utility theory. Pioneering research by Moody and Saffell [1998] highlighted the construction of trading systems using reinforcement learning to optimize financial objectives. Their results showed that these proposed systems, one based on Q-learning and the other on maximizing the Sharpe differential — outperformed the S&P 500 over a 25-year period. This finding suggests a predictable structure in US stock prices, contrasting with the Efficient Market Hypothesis (EMH). In addition, transaction costs can be directly implemented in the RL agent's rewards, allowing it to learn a trade-off between incurring transaction costs and the strength of a trading signal. This desirably would lead to net positive returns, as the agent should learn to trade only — or change the direction of its current position — when the signal is strong enough. This flexibility in altering the reward function and making sequential decisions while considering more aspects than just price prediction makes RL highly useful for quantitative trading. There exist three categories of RL frameworks, namely:

- **Critic-Only Approach**: This method focuses on learning a value function that guides the agent by evaluating the expected outcomes of different actions given the state representation. During decision-making, the agent receives the current state of the environment, such as the current price, volatility, and volume, and selects the action with the best expected outcome according to the value function, hence the name 'critic'. This value function is typically updated using the Bellman Equation, which provides a recursive decomposition of the expected rewards and includes a discount factor to balance the significance of immediate versus future rewards.
- **Actor-Only Approach**: In this approach, the agent receives the state of the environment as input and acts based on that without evaluating the expected rewards of different actions. The agent learns a direct correspondence from states to actions. Actor-only approaches

typically involve a continuous action space and exhibit faster convergence during training compared to the critic-only approach.

- **Actor-Critic Approach**: This relatively new method combines the strengths of the critic-only and actor-only approaches. The actor determines the agent's action given the state representation, and the critic evaluates the selected action of the actor. Through trial and error, the actor learns to choose actions that the critic finds best, while the critic improves its evaluation accuracy.

According to the survey conducted by Fischer [2018] in which he explored 50 different publications on these RL approaches for financial markets, all three approaches have their own advantages and weaknesses and there is no clear winner. The main advantage of the critic-only approach is the fact that the reward design of such models does not need to be differentiable, and for that reason, an intricate reward function can be made to align the objective of the RL agent with the preferences of an investor; for example, it is quite simple to include frictions such as transaction costs or drawdown penalties to the reward function of the RL agent. Moreover, due to the fact that critic-only relies on the Bellman Equation, the discounting of future rewards can be easily managed. Alternatively, the main disadvantage of the critic-only is the fact that it suffers from the Bellman curse of dimensionality [Bellman, 1957], and increasing the number of discrete actions leads to a significant growth in computational complexity and memory requirements, making it challenging to efficiently store and update the value function for all state-action pairs. Then, the actor-only approach has the advantage that unlike critic-only approach it has a continuous action space, making it more robust when the number of actions is relatively large, for example, it would be much more tractable to design an asset allocation framework that assigns fractions of wealth to assets which would be more involved with a critic-only approach. However, the main disadvantage of actor-only models is the fact that they require differentiable reward functions, and for that reason, it is harder to manipulate the reward function to match the preferences of an investor. Lastly, the actor-critic models combine the advantages of both the critic-only and actor-only models. However, they have not been extensively investigated empirically and, according to the existing literature, do not perform as well as the critic-only and actor-only models [Li et al., 2007; Bekiros, 2010].

Hence, the best reinforcement learning model for financial use depends on the specific scope. If the goal is asset allocation or portfolio management, the critic-only approach would be much harder to implement, whereas the actor-only and actor-critic approaches are better suited for such problems. However, in cases where the action space is not very large, such as in a trading framework where the agent is used for single-asset trading with a limited number of actions, the advantage of the critic-only approach in handling non-differentiable reward functions becomes more desirable. For that reason, for the scope of this thesis, we will consider the critic-only approach, more specifically, the Deep Q-Networks framework.

2.5 Deep Q-Networks

Deep Q-Networks (DQN) is a reinforcement learning algorithm that blends traditional Q-learning with deep neural networks. DQN approximates the Q-value function, which estimates the expected future rewards for each action in a given state, using a neural network architecture. In the original paper by the developers of DQN, they employed a Convolutional Neural Network (CNN) architecture to process images [Mnih et al., 2013]. However, various types of neural network architecture can be used as function approximators, as well as other supervised machine learning models, depending on the task at hand. For example, in financial trading, architectures such as Long Short-Term Memory (LSTM) or Multi-Layer Perceptron (MLP) would be more fitting because they are more effective at handling sequential data and understanding patterns over time, unlike CNNs which are designed to process spatial data [Théate and Ernst, 2021;

Abbasimehr and Paki, 2022; He et al., 2023]. Furthermore, DQN employs experience replay, storing past experiences and sampling them randomly to break correlation and improve learning stability. This approach allows DQN to handle high-dimensional state spaces and has achieved superhuman performance in playing Atari games.

DQN has also been researched for trading purposes. Zhang et al. [2019] constructed three types of Deep Reinforcement Learning (DRL) models for trading financial assets and tested them on 50 futures contracts across four asset classes based on daily data. They implemented a critic-only DQN and two actor-critic approaches, namely, Proximal Policy Optimization (PPO) and Advantage Actor Critic (A2C) models. They used a Long Short-Term Memory (LSTM) neural network architecture because of its superior capability with time-series data. Furthermore, they normalized the exposure of trading contracts to maintain constant risk exposure by performing volatility scaling. As a reward function, they used the net trade profits after transaction costs. The authors then constructed equally weighted portfolios for each asset class and evaluated the performance of these portfolios in terms of profitability, risk-adjusted returns, and drawdown management. They found that the DQN method gave the best results in terms of profitability, risk-adjusted returns, and drawdown management. A2C came second, and PPO third, with all three methods outperforming the Buy & Hold and technical trading benchmark strategies at high transaction costs of up to 50 basis points with respect to each taken position.

Additionally, Théate and Ernst [2021] constructed a modified DQN algorithm for intraday algorithmic trading purposes. They primarily considered stocks from various regions of the world, including the US, European and Asian markets, across different industries. The auhors developed a DQN agent capable of trading individual assets. The function approximator used was a deep neural network with an MLP architecture. The reward for the agents was based on net trade returns after transaction costs, although he acknowledges that this is a simplification of the Sharpe ratio, which is the ultimate objective for a trader. He suggests that it would be interesting to research how to better align daily returns as rewards with the Sharpe ratio, as the current approach maximizes cumulative daily returns rather than directly optimizing for the Sharpe ratio. Furthermore, the authors compared the performance of the DQN agents across different stocks with passive strategies like Buy & Hold and Short & Hold, as well as active trading strategies such as trend following and mean reversion using moving averages. He evaluated the performance of these strategies using metrics such as Sharpe ratio, Sortino ratio, profit & loss, annualized return, volatility and maximum drawdown, similar to Zhang et al. [2019]. The results indicated that the DQN agent consistently outperformed the benchmark strategies on average and demonstrated robustness across various trading costs.

Similarly, Carapuço et al. [2018] designed a reinforcement learning framework for trading in the foreign exchange market, utilizing a critic-only DQN approach with a three-layer neural network as a function approximator. The authors' primary aim was to construct a practical DQN agent tailored for forex trading. There are few important differences with the previous two DQN based papers. Firstly, the agent adjusts its position direction bi-hourly instead of engaging in intraday trading. Secondly, unlike Zhang et al. [2019] and Théate and Ernst [2021], where the agent was rewarded based on net returns or net profit, Carapuço et al. [2018] used a variation of the Sortino ratio — a measure that penalizes downside risk — as the agent's reward. This approach is similar to Moody and Saffell [1998], which used the Sharpe differential, a variation in the Sharpe ratio. This adjustment aligns the DQN agent's objective with the real-life goal of maximizing risk-adjusted returns for investors. The authors observed a stable learning process and noted that the agent successfully identified relationships in out-of-sample test data, despite the non-stationary and noisy nature of financial data. These findings underscore the potential of DQN in algorithmic trading.

2.6 Reward Design & Utility Theory

As mentioned earlier, one of the strengths of using an critic-only approach, such as Deep Q-Networks, in quantitative trading is the ability to align the RL agents' objectives with those of investors. Investors typically aim to maximize risk-adjusted returns, often measured by ratios like the Sharpe ratio, the Sortino ratio (which considers only the downside deviation of returns), and the Calmar ratio (which considers returns in relation to the maximum drawdown over a period). Therefore, it is crucial to design the reward function for the agent to either directly correspond to these ratios or to produce quantities highly correlated with such ratios.

In economics, utility theory models how individuals or organizations make choices under uncertainty, aiming to maximize their gratification or utility [Poon, 2018]. Utility theory measures the benefit an individual gains from a specific choice, considering their risk preferences and aversion. Thus, it balances risk and return, similar to what the Sharpe ratio measures, return per unit of risk. This theoretical framework can guide the design of reward functions that our DQN agent should maximize in its environment. We will present the family of power utility functions, propose two opposing types of utility functions, and construct reward functions based on these types.

2.6.1 Power Utility

The power utility function expresses the utility of an agent concerning their degree of risk aversion, often denoted by the parameter γ, towards changes in their wealth [Poon, 2018]. Denoting wealth by W, the power utility function is defined as:

$$U(W) = \frac{W^{1-\gamma} - 1}{1-\gamma} \quad \text{with } \gamma \neq 1 \tag{1}$$

For $\gamma = 0$, the agent is risk-neutral towards changes in their wealth level, resulting in a utility function where utility increases linearly with wealth. A risk-neutral agent values gains and losses equally, meaning that a one-dollar gain impacts the utility as much as a one-dollar loss. When $\gamma > 0$, the agent exhibits risk aversion towards changes in their wealth, resulting in a concave function. Here, the agent shows diminishing marginal utility — each additional unit of wealth provides less satisfaction. Unlike the linear case ($\gamma = 0$), a loss impacts the agent's utility more significantly than a gain of the same magnitude. As γ approaches 1, the power utility function becomes undefined. However, by taking the limit and using L'Hôpital's rule, we obtain:

$$\begin{aligned} \lim_{\gamma \to 1} U(W) &= \lim_{\gamma \to 1} \frac{W^{1-\gamma} - 1}{1-\gamma} \\ &= \lim_{\gamma \to 1} \frac{(-1)W^{1-\gamma}\log(W)}{-1} \\ &= \log(W). \end{aligned} \tag{2}$$

Thus, the logarithmic utility is achieved when the risk-aversion parameter $\gamma = 1$, which is a special case of the power utility. In addition, one of the key properties of the power utility function is constant relative risk aversion (CRRA). Using the Arrow-Pratt measure of relative risk aversion Pratt [1964]:

$$\text{RRA} = -\frac{U''(W)}{U'(W)} W = \gamma. \tag{3}$$

This means that the agent has constant relative risk aversion, indicating that regardless of the wealth level, the agent demonstrates the same degree of risk aversion. If $\gamma > 0$, the agent is risk-averse; if $\gamma < 0$, the agent is risk-seeking, meaning the utility function is convex, and as wealth increases, marginal utility increases, leading the agent to prefer riskier choices with higher potential returns.

2.6.2 Reward Design

Hence, with respect to the previous section, we need to determine the reward for the DQN agent after each action it takes within the environment. In the studies by Zhang et al. [2019] and Théate and Ernst [2021], linear utility functions were applied, since net profits minus transaction costs and net returns served as agent rewards, respectively. However, this approach might be problematic because an agent with a linear utility function could exhibit degeneracy when optimizing such an objective. It would be indifferent between a certain return and a uncertain return with the same expected value, potentially leading the agent to consistently seek more exposure as long as the expected value is positive, without considering the catastrophic impacts it could have on the portfolio.

In contrast, concave utility functions are more appropriate and align better with an investor's preferences, as the risk taken in a trade should be tolerable for the investor, and as the wealth grows, preservation of wealth becomes significantly more important which translates into diminshing marginal utility. Therefore, it theoretically makes more sense to use a power utility function that incorporates a risk aversion parameter γ greater than 0. To test this, we will construct a DQN agent with a linear utility function and another DQN agent with a logarithmic utility function. This approach will help us evaluate the impact of these reward designs on the performance of DQN agents and whether they lead to the expected behavior as dictated by utility theory.

2.7 Position Size

Before we delve into the methodology, we introduce one important aspect of trading strategies: the position sizing rule.

As Scholz [2012] claims, the use of technical trading rules, which only offer signals for long or short positions, necessitates that the trader determines the exposure in each transaction. Even though position sizing crucially affects the risk and return of a portfolio, recent academic literature has largely neglected this aspect. The author explains that there are a few types of position sizing rules, also referred to as money management rules, that can complement a trading rule. The first type involves always buying or selling one unit of the given security. He refers to such positions as erratic position sizes because the position size only depends on the share price of the asset. Alternatively, the other type of position sizing involves relative position sizing, which can be implemented by always investing fixed absolute amounts or by always investing a static proportion of the remaining capital. He also notes that it is surprising that current literature does not take this into account, as experienced traders are generally aware of the trade-off in money management. Specifically, undersized positions cannot maximize a timing strategy's potential, and even robust timing strategies might lead to catastrophic losses if they involve too much exposure. Scholz empirically demonstrates this by testing erratic, absolute, and relative position sizing strategies with identical technical (timing) strategies using simple moving averages. He concludes that using relative money management significantly reduces the risks in a financial portfolio, and only in some specific cases can erratic position sizing lead to robust performance. Lastly, he concludes that smaller fractions of relative position sizes deliver the highest risk-adjusted returns in most cases.

The phenomenon that Scholz describes is also true for the DQN studies that we have presented regarding the use of reinforcement learning in trading. Note that the RL agent is essentially learning to optimize a timing strategy, given that the action space is reduced to only the direction of trades to take, not how much to trade. This is because the DQN agent has a discrete action space, and the Bellman Equation's curse of dimensionality prevents an increase in the number of actions such that they map to a range of position sizes. Yet, position sizing is not really

considered in these studies. Moody and Saffell [1998] discusses taking a constant magnitude of exposure without explicitly defining it but acknowledges that relaxing the 'constant' magnitude would provide better risk control. Théate and Ernst [2021] attempts to maximize each position size given their budget and liquidity constraints, where they need to withhold cash for already open short positions; they take the maximum cash available and buy shares for a position if buying, and when shorting, they maximize their available cash such that they can pay back the broker they borrow shares from. Carapuço et al. [2018] maintains the position size at a constant level, though they do not specify whether this is in an absolute or relative sense. Zhang et al. [2019] uses the most sophisticated position sizing rule based on volatility targeting, where they always scale a static fraction of the number of units of a contract with a scaler. The scaler is based on a fixed volatility target, where the annualized volatility target is divided by the ex-ante estimate of market volatility. This way, the exposure is reduced when market volatility is high and increased when market volatility is low. There is significant empirical evidence that volatility targeting leads to better risk-adjusted returns and reduces the impacts of left-tailed events such as crises [Moreira and Muir, 2015; Dreyer and Hubrich, 2017; Lim et al., 2019; Mylnikov, 2021].

For this reason, we will also adopt volatility scaling as one of the position sizing strategies for our DQN agents. Furthermore, we will extend this by considering an ensemble momentum-volatility scaling strategy, which combines the position sizing strategy of Baz et al. [2015] based on the strength of the trend signal and volatility scaling. By incorporating this ensemble approach, when a trend signal is strong and volatility is not too high, the position size increases, potentially leading to higher returns. In this way, the size of the position is a function of both the volatility and the strength of the ongoing trends.

3 Methodology

We have now introduced the important concepts and their backgrounds, and presented a summary of the related works. This section outlines the structure of our dynamic trading models based on Deep Q-Networks (DQN) developed by Mnih et al. [2013]. We discuss the model's components, including the environment, the agent, and the reward functions we will experiment with.

3.1 Markov Process

The reinforcement learning trading problem is framed as a Markov Decision Process (MDP) where the RL agent engages with the environment in discrete time intervals. At each time step t, the agent receives a representation of the environment, called the state S_t. Based on the state, the agent decides on a certain action A_t, following which a numerical reward R_{t+1} is assigned based on the action taken. The sequence of rewards $\mathbf{R} = \{R_1, R_2, R_3, \dots\}$ forms a controlled Markov process influenced by the agent's actions.

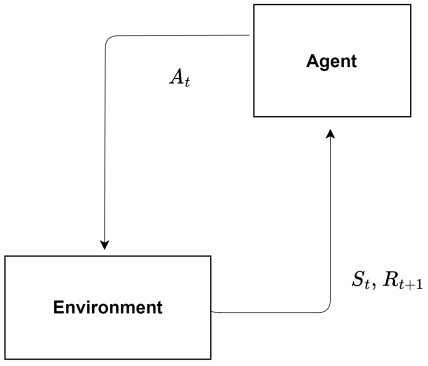


Figure 1: The process of a reinforcement learning trading problem at each time step t.

This sequence can occur over a finite but sufficiently large number of steps T. The objective at each step t is to maximize the expected discounted cumulative rewards, with $\gamma \in (0, 1)$ as the discount factor for future rewards. Even though the sequence has a finite number of steps, when T is sufficiently large, the influence of future rewards diminishes due to discounting, and the problem can be approximated using the infinite time horizon assumption. This approximation simplifies the theoretical analysis and is commonly used in reinforcement learning algorithms like DQN, as it ensures the value function remains stationary i.e., independent of time.

The sum of discounted future rewards at time t is then defined as:

$$G_t = \sum_{k=t+1}^{\infty} \gamma^{k-t-1} R_k. \tag{4}$$

For sufficiently large T, this can be approximated as:

$$G_t \approx \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k. \tag{5}$$

The value function $V(S_t)$ represents the expected sum of discounted future rewards starting from state S_t and following a particular policy π:

$$V(S_t) = \mathbb{E}_\pi \left[G_t \mid S_t \right]. \tag{6}$$

Using the definition of G_t, and assuming an infinite time horizon, the value function can be expressed as:

$$V(S_t) = \mathbb{E}_\pi \left[\sum_{k=t+1}^{\infty} \gamma^{k-t-1} R_k \mid S_t \right]. \tag{7}$$

This can be split into the immediate reward and the sum of discounted future rewards from the next time step:

$$V(S_t) = \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t \right]. \tag{8}$$

By using the recursive definition of the value function, we arrive at the Bellman Equation:

$$V(S_t) = \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) \mid S_t \right]. \tag{9}$$

To find the optimal value function $V^*(S_t)$, which represents the maximum sum of discounted future rewards, we solve the Bellman Optimality Equation:

$$V^*(S_t) = \max_{A_t} \mathbb{E} \left[R_{t+1} + \gamma V^*(S_{t+1}) \mid S_t, A_t \right]. \tag{10}$$

Hence, this optimization problem involves finding the optimal policy, π^*, that maximizes the expected sum of discounted forthcoming rewards for each state S_t. The Bellman Optimality Equation characterizes this optimization, providing a recursive way to compute the optimal value function and hence determine the best actions to take in each state to maximize long-term rewards.

This section sets the foundation for understanding how we will use the MDP framework to apply reinforcement learning techniques for building and training the DQN agents. Next, we will present our state-space, action-space, and reward functions before delving into the specifics of the DQN architecture and the training process.

3.2 State Space

The environment's representations, or state space, are crucial for the agent to make informed trading decisions. As we focus on intraday trading, we select relevant independent variables for this purpose — such as prices, returns, volatility, and technical indicators — appropriate for short-term price predictions. To effectively capture the temporal dynamics of the time series data, we also incorporate lags of these variables. Although we will elaborate on using neural networks (MLP) in the following sections, it is important to note that any supervised learning algorithm can be employed. Furthermore, this thesis does not explore potential new features but relies on indicators commonly used for predictive analysis in research on financial markets [Htun et al., 2023]. The dependent variable in our model is the value function $V(S_t)$. While the value function $V(S_t)$ depends on the state S_t, it ultimately represents the expected return based on the agent's actions under a specific policy. Thus, $V(S_t)$ implicitly accounts for the actions A_t that the agent takes from state S_t. This establishes our starting point to present the chosen independent variables and features.

We include normalized past prices ($\bar{p}_t$) and returns ($\bar{r}_t$) over different horizons, trend reversal indicators like Moving Average Convergence Divergence (MACD) over varied periods, and the Relative Strength Index (RSI) to identify overbought or oversold price ranges. The list of independent variables representing the state space for an agent trading asset i is presented below.

1. The daily close prices which are normalized in the following way

$$\bar{p}_t^{(i)} = \frac{p_t^{(i)} - \widehat{\mathbb{E}(p_t^{(i)})}}{\widehat{\sigma(p_t^{(i)})}} \tag{11}$$

where $\widehat{\mathbb{E}(p_t^{(i)})} = \frac{1}{T}\sum_{k=1}^{T} p_k^{(i)}$ and $\widehat{\sigma(p_t^{(i)})} = \sqrt{\frac{1}{T-1}\sum_{k=1}^{T}\left[p_k^{(i)} - \mathbb{E}\left(p_k^{(i)}\right)\right]^2}$ are the empirical mean and standard deviation of the close prices in the training set, respectively. Note that we specify the training set in Section 4.1.

2. Simple past returns over multiple horizons namely: one month, two months, six months, and one year i.e.,

$$r_{t-j,t}^{(i)} = \frac{p_t^{(i)}}{p_{t-j}^{(i)}} - 1 \quad \text{for } j \in \{20,\ 40,\ 60,\ 252\} \tag{12}$$

Note that we could also have taken logarithmic returns. However, in the current literature on DQN agents for short-term trading, simple returns are used. For the sake of consistency with the literature [Moody and Saffell, 1998; Zhang et al., 2019; Théate and Ernst, 2021], and given that the difference between simple returns and logarithmic returns for short intervals is negligible, we use simple returns. Moreover, following Lim et al. [2019], we normalize these returns by the daily volatility of a specific time period. Thus, the normalized returns are defined as follows:

$$\bar{r}_{t-j,t}^{(i)} = \frac{r_{t-j,t}^{(i)}}{\sigma_t^{(i)}\sqrt{j}} \quad \text{for } j \in \{20,\ 40,\ 60,\ 252\} \tag{13}$$

where $\sigma_t^{(i)}$ is the 3-month exponentially moving average standard deviation of the daily returns. Notice that three months of financial trading days corresponds to a span of 63 days which we can translate into a smoothing factor to calculate the exponentially moving average (EMA) of the daily returns, denote ψ as the span, then:

$$\alpha(\psi) = \frac{2}{\psi + 1} \text{ with } \psi \geqslant 1. \tag{14}$$

The EMA is calculated recursively using the following equations:

$$\begin{aligned} \text{EMA}_{t-\psi}^{(i)} &= r_{t-\psi-1,t-\psi}^{(i)} \\ \text{EMA}_{t-\psi+1}^{(i)} &= \alpha(\psi)\cdot r_{t-\psi,t-\psi+1}^{(i)} + (1-\alpha(\psi))\cdot \text{EMA}_{t-\psi}^{(i)} \\ &\vdots \\ \text{EMA}_{t}^{(i)} &= \alpha(\psi)\cdot r_{t-1,t} + (1-\alpha(\psi))\cdot \text{EMA}_{t-1}^{(i)} \end{aligned} \tag{15}$$

and the $\sigma_t^{(i)}$ is then calculated recursively using the following equations:

$$\begin{aligned} {\sigma^2_{t-\psi}}^{(i)} &= 0 \\ {\sigma^2_{t-\psi+1}}^{(i)} &= \alpha(\psi)\cdot(r_{t-\psi,t-\psi+1} - \text{EMA}_{t-\psi+1}^{(i)})^2 + (1-\alpha(\psi))\cdot {\sigma^2_{t-\psi}}^{(i)} \\ &\vdots \\ \sigma_t^{2\,(i)} &= \alpha(\psi)\cdot(r_{t-1,t} - \text{EMA}_{t}^{(i)})^2 + (1-\alpha(\psi))\cdot {\sigma^2_{t-1}}^{(i)}. \end{aligned} \tag{16}$$

Finally, by taking $\sqrt{\sigma_t^{2\,(i)}}$, we attain the 3-month exponentially moving average standard deviation of the daily returns for $\psi = 63$.

3. The trend-following momentum indicator, MACD, is a technical indicator which relies on the difference between two exponential moving averages (EMA) of short and long-term periods to identify potential buy and sell signals. Furthermore, it can be used to recognize trend reversals, for if the MACD line crosses above or below the zero line, it may indicate a shift

in the prevailing trend direction. We will use the volatility normalized MACD indicator as proposed by Baz et al. [2015], which is given by:

$$\text{MACD}^{(i)}_{t,S:L} = \frac{q^{(i)}_{t,S:L}}{\text{std}\left(p^{(i)}_{t-252:t}\right)} \tag{17}$$

with

$$q^{(i)}_{t,S:L} = \left(\frac{m_i(S) - m_i(L)}{\text{std}\left(p^{(i)}_{t-63:t}\right)}\right) \tag{18}$$

where the normalization factors $\text{std}(p^{(i)}_{t-63:t})$ and $\text{std}(p^{(i)}_{t-252:t})$ are the 3-month and 1-year rolling standard deviation of the prices $p^{(i)}_t$, respectively. These rolling standard deviations of the prices can be calculated using the following equations:

$$\begin{aligned} \tilde{p}^{(i)}_t &= \tfrac{1}{n}\textstyle\sum_{k=t-n+1}^{t} p^{(i)}_k \\ \text{std}(p^{(i)}_{t-n:t}) &= \sqrt{\tfrac{1}{n-1}\textstyle\sum_{k=t-n+1}^{t}(p^{(i)}_k - \tilde{p}^{(i)}_t)^2}. \end{aligned} \tag{19}$$

Furthermore, $m_i(S)$ and $m_i(L)$ are the exponentially moving averages of asset prices on the short- and long-term time spans, respectively. To obtain the volatility normalized MACD values for various time scales, we use $S \in \{8, 16, 32\}$ and $L \in \{24, 48, 96\}$ as in the original paper of the authors. For the calculation of these moving averages, the spans are translated to their corresponding smoothing factors using Equation 14. Then like before with the EMA's for the daily returns, the EMA's for the prices are calculated for the different time spans in a recursive fashion:

$$\begin{aligned} \text{EMA}^{(i)}_{t-\psi} &= p^{(i)}_{t-\psi} \\ \text{EMA}^{(i)}_{t-\psi+1} &= \alpha(\psi)\cdot p^{(i)}_{t-\psi+1} + (1-\alpha(\psi))\cdot \text{EMA}^{(i)}_{t-\psi} \\ &\vdots \\ \text{EMA}^{(i)}_{t} &= \alpha(\psi)\cdot p_t + (1-\alpha(\psi))\cdot \text{EMA}^{(i)}_{t-1} = \underline{m_i(\psi)} \end{aligned} \tag{20}$$

4. RSI is a momentum oscillator between 0 and 100 that measures the speed of changes in prices which is indicative of when an asset is overbought (high RSI) or oversold (low RSI) [Wilder, 1978]. Moreover, in order to calculate the RSI for asset i in period t, we first need to separate the positive returns $r^{(i)+}_{t,t+1}$ and the negative returns $r^{(i)-}_{t,t+1}$:

$$r^{(i)+}_{t,t+1} = \begin{cases} r^{(i)}_{t,t+1} & \text{if } r^{(i)}_{t,t+1} > 0, \\ 0 & \text{if otherwise} \end{cases} \quad \text{and} \quad r^{(i)-}_{t,t+1} = \begin{cases} -r^{(i)}_{t,t+1} & \text{if } r^{(i)}_{t,t+1} < 0, \\ 0 & \text{if otherwise} \end{cases}$$

Then, we calculate the rolling mean of the average gain ($\overline{G}$) and average loss ($\overline{L}$) over a specified period T:

$$\overline{G} = \frac{1}{T}\sum_{j=1}^{T} r^{(i)+}_{t-j,t-j+1} \quad \text{and} \quad \overline{L} = \frac{1}{T}\sum_{j=1}^{T} r^{(i)-}_{t-j,t-j+1}$$

Finally, the RSI is then defined as:

$$\text{RSI}_t^{(i)} = 100 - \frac{100}{1 + \frac{\overline{G}}{\overline{L}}} = \frac{100\overline{G}}{\overline{G} + \overline{L}} \tag{21}$$

Note again that we will be using a rolling window of 3 months, i.e. $T = 63$.

Ultimately, for each feature, the past five observations (lags) are used to form a single state for asset i at time t i.e. denoting $S_t^{(i)}$ as the vector containing the features:

$$\begin{bmatrix} \overline{p}_{t-5}^{(i)}, \ \overline{p}_{t-4}^{(i)}, \ \cdots, \ \overline{p}_t^{(i)}, \overline{r}_{t-20,t}^{(i)}, \ \overline{r}_{t-40,t}^{(i)}, \ \cdots, \ \overline{r}_{t,252}^{(i)}, \text{MACD}_{t-5,8:24}^{(i)}, \ \text{MACD}_{t-5,16:48}^{(i)}, \\ \text{MACD}_{t-5,32:98}^{(i)}, \ \cdots, \ \text{MACD}_{t,16:48}^{(i)}, \ \text{MACD}_{t,32:98}^{(i)}, \text{RSI}_{t-5}^{(i)}, \ \text{RSI}_{t-4}^{(i)}, \ \cdots, \ \text{RSI}_t^{(i)} \end{bmatrix}. \tag{22}$$

3.3 Action Space

We consider an agent that trades a single asset with a discrete action space, effectively deciding on the type of position to bet on: long, short, or hold. As discussed in Section 2.7, this is done because the Deep Q-Network algorithm has a discrete action space and relies on the Bellman Equation, which suffers from Bellman's curse of dimensionality. This constraint prevents us from increasing the actions to correspond to a range of positions. Therefore, the agent only needs to focus on the direction and timing of trades, and we will use sophisticated position sizing rules to complement the agent's actions. The discrete action space is defined as follows:

$$A_t = \begin{cases} 1: & \text{long}, \\ -1: & \text{short}, \\ 0: & \text{hold}. \end{cases}$$

3.4 Position Sizing

The size of the positions corresponding to the agent's actions — specifically, how much the agent should trade in a period t given action A_t — is determined using two different dynamic position sizing rules. A robust position sizing rule enhances the agent's risk management and prevents catastrophic losses to the portfolio [Scholz, 2012].

3.4.1 Volatility Targeting

The first position sizing rule will be based on volatility targeting, where we inversely size our position with respect to the ex-ante estimate of the volatility of the given asset i. It has been empirically shown that volatility targeting can reduce the impact of black swan events, improve returns, and overall stabilize the risk profile of an investment portfolio [Dreyer and Hubrich, 2017; Mylnikov, 2021]. We consider an annualized volatility target σ_{target} of 15% in accordance with Lim et al. [2019], and define a position scalar $\beta_t^{(i)}$ for asset i at time t, which we will refer to as the 'volatility factor':

$$\beta_t^{(i)} = \frac{\sigma_{target}}{\sigma_{t-1}^{(i)}\sqrt{252}} \tag{23}$$

where $\sigma_{t-1}^{(i)}$ is the ex-ante volatility estimate and corresponds to the 3-month exponentially moving average standard deviation of the daily returns, calculated in the same fashion as before using Equation 16. This volatility factor, $\beta_t^{(i)}$, is then multiplied by a fixed fraction of the

available wealth, $\hat{X}_t^{(i)} = \mu_V W_t^{(i)}$ for the asset i in period t, with $\mu_V \in (0, 1)$, which generates the position size, $X_t^{(i)}$:

$$X_t^{(i)} = \frac{\sigma_{target}}{\sigma_{t-1}^{(i)}\sqrt{252}} \hat{X}_t^{(i)} \tag{24}$$

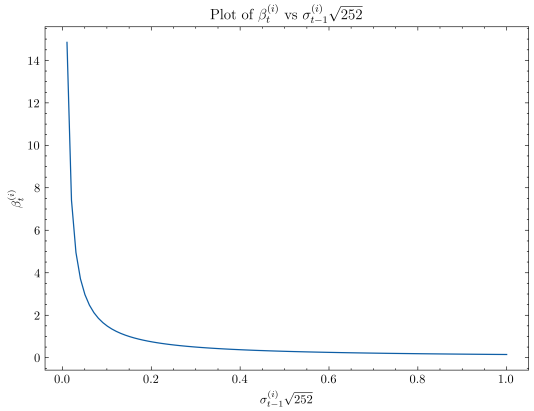


Figure 2: The range of the volatility factor, $\beta_t^{(i)}$ when using a volatility target of 15%.

In Figure 2, we illustrate the relationship between the volatility factor $\beta_t^{(i)}$ and the annualized ex-ante volatility estimate. When the annualized volatility estimate equals the volatility target, the size of the position will be equal to $\hat{X}_t^{(i)}$. If the estimated volatility is lower than the volatility target, the position size will grow exponentially and inversely. Conversely, if the volatility target exceeds the volatility estimate, the position size will become smaller than the fixed fraction of wealth $\hat{X}_t^{(i)}$, resulting in reduced exposure to the high-risk state of the market.

3.4.2 MACD Signal

The second position sizing approach combines volatility targeting with the volatility normalized MACD trading rule proposed by Baz et al. [2015]. This trading rule, is a trend-following momentum method which quantifies the strength and direction of a trend at time t. Similarly to our MACD features, the trend estimate is the weighted average of the MACD signals of different horizons as defined in Equation 17:

$$\text{MACD}_t^{(i)} = \sum_{k=1}^{3} \text{MACD}_{t,S_k:L_k}^{(i)} \tag{25}$$

This trend signal is then translated into a position scalar, $\eta_t^{(i)}$, which we will refer to as the 'trend factor'.

$$\eta_t^{(i)} = \frac{\text{MACD}_t^{(i)} \exp\left(\frac{\text{MACD}_t^{(i)2}}{4}\right)}{0.89}. \tag{26}$$

In Figure 3, we show the relationship between the trend signal and the trend factor. We can observe that the volatility-normalized MACD signal ranges between $-\sqrt{2}$ and $\sqrt{2}$, while the trend factor ranges between -1 and 1, with -1 indicating minimal volatility and a downtrend, and 1 indicating minimal volatility and an uptrend. The trend factor approaches zero when the trend strength is low or when the volatility is at its peak.

Since our agent decides the direction – whether to take a short or long position – we will use the absolute value of the trend factor, ensuring that it ranges from 0 to 1. Then, we can formulate the ensemble position sizing rule using both the trend factor, $\eta_t^{(i)}$ as well as the volatility factor $\beta_t^{(i)}$:

$$\begin{aligned} X_t^{(i)} &= \left|\eta_t^{(i)}\right| \beta_t^{(i)} \hat{X}_t^{(i)} \\ &= \left| \frac{\text{MACD}_t^{(i)} \exp\left(\frac{\text{MACD}_t^{(i)2}}{4}\right)}{0.89} \right| \frac{\sigma_{target}}{\sigma_{t-1}^{(i)}} \hat{X}_t^{(i)} \end{aligned} \tag{27}$$

where, as before, $\hat{X}_t^{(i)} = \mu_M W_t^{(i)}$ is a fixed fraction of the available wealth with $\mu_M \in (0, 1)$ for the asset i in period t.

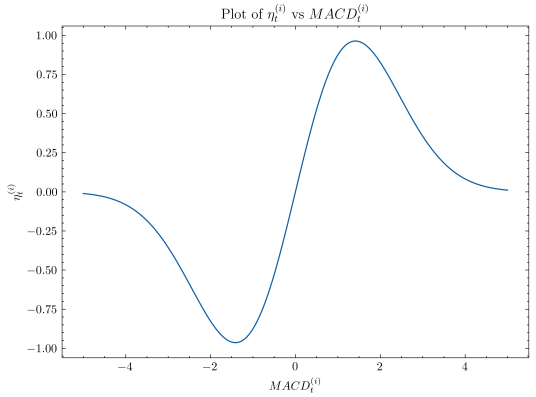


Figure 3: The range of the trend factor, $\eta_t^{(i)}$, when using MACD signal proposed by Baz et al. [2015].

The motivation for this ensemble position sizing rule is to refine our position sizing strategy by incorporating both volatility targeting and market trends. This approach allows for the adjustment of position sizes based on the strength of market trends. When significant trends are detected, we can increase our position sizes to capitalize on these opportunities. Conversely, during periods of low volatility and weak trend signals, we reduce our positions more than volatility targeting alone would suggest. This method aims to enhance returns by strategically adjusting our exposure according to market conditions. For this reason, we examine the effectiveness of this ensemble position sizing rule.

Finally, to conclude this section on position sizing, note that for both position sizing rules, for instances where the wealth $W_t^{(i)}$ corresponding to asset i is lower than the determined position size $X_t^{(i)}$, the position size will be equal to $W_t^{(i)}$ i.e. the actual position size is given by:

$$\bar{X}_t^{(i)} = \min\left(X_t^{(i)}, W_t^{(i)}\right). \tag{28}$$

3.5 Integration of Actions and Position Sizing Rules

When the agent decides to take a long or short position, transaction costs will incur as a static percentage of the trade position, while holding does not incur transaction costs. The position sizing rule determines the optimal position size for each period. The trade position $\bar{X}_t^{(i)}$ is

established at the end of each time period t and is maintained until $t+1$, at which point the position is evaluated in terms of profits and closed unless the action A_{t+1} is holding. If the action A_{t+1} is not 'to hold', the position-sizing rule produces a new position size $\bar{X}^{(i)}_{t+1}$ and transaction costs are incurred with respect to this new position size. If A_{t+1} is to hold the last position, the position sizing rule will not be used, as the exposure has already been determined in the previous periods, and the position size $\bar{X}^{(i)}_{t+1}$ will be equal to the last net position size adjusted for the change in price, $(1+r^{(i)}_{t,t+1})\bar{X}^{(i)}_{t}$. To facilitate this, we introduce an auxiliary parameter $X^{(i)}_{LT}$, which keeps track of the position size of the last period for asset i. This parameter allows for straightforward calculation of the position size for holding positions. For example, if the agent decides to hold, the net position in step $t+1$ is calculated as:

$$\bar{X}^{(i)}_{t+1} = (1+r^{(i)}_{t,t+1})X^{(i)}_{LT}$$

and the parameter $X^{(i)}_{LT}$ is updated as follows:

$$X^{(i)}_{LT} = \bar{X}^{(i)}_{t+1}.$$

This update occurs after every action, including buying, selling, and holding. Furthermore, in the initial periods, if there have not yet been any instances of buying or selling actions, the hold action serves as a neutral position since no exposure has been determined. In this case, $X^{(i)}_{LT} = 0$, indicating that there is no position to hold. In summary, the size of the position $\bar{X}_{t+1}$ is determined as follows with respect to the action A_{t+1}:

$$\bar{X}^{(i)}_{t+1} = \begin{cases} \bar{X}^{(i)}_{t+1} - \lambda\bar{X}^{(i)}_{t+1} & \text{if } A_{t+1} \in \{-1, 1\}, \\ (1+r^{(i)}_{t,t+1})X^{(i)}_{LT} & \text{if } A_{t+1} = 0. \end{cases}$$

As an illustration, consider two periods of trading:

- **Period 1:** The agent chooses to buy, and the position sizing rule allocates €100 to invest. With a transaction cost rate λ of 10 basis points (bp), the net position in period 1 will be €100 - 0.10 = €99.90.
- **Period 2:** The realized profit/loss is calculated. Assume the price has increased to 101, the return is 1%. If the agent holds, the net position will be €1.01 * €99.90 = €100.899, and no new transaction costs will occur. However, if the agent chooses to sell or buy, the position size will again be determined by the position sizing rules, and transaction costs will incur.

Using this framework, the agent can effectively integrate its discrete action space with dynamic position sizing rules to optimize trading strategies while managing transaction costs.

3.6 Reward Function

A benefit of using a model-free reinforcement learning algorithm like DQN is that its reward functions do not need to be differentiable and for that reason it gives us the flexibility to construct more complex reward functions and incorporate frictions such as transaction costs directly into the reward function [Fischer, 2018]. Reward functions are crucial because they define the utility of the agent by assigning values to each action's outcome, guiding the agent to learn which actions lead to the most beneficial results. We explore the impact of two different reward function designs that correspond to distinct types of utility functions for the agent aiming to optimize its (risk-adjusted) returns. This dual-approach allows for a comprehensive analysis of how different reward structures influence decision-making strategies of the DQN agent.

3.6.1 Linear Utility

An straightforward approach is to give the realized net returns of a trade i.e. the net profits to the agent as its reward, and examine whether it leads to desirable results in terms of risk adjusted performance. This approach aligns with the preferences of a risk-neutral investor and corresponds to a linear utility function where gains and losses weigh equally and the agent maximizes the expected value of the cumulative returns. As our position sizes $\bar{X}_t^{(i)}$ are dynamic and vary within each period, we normalize the net profits by dividing them by $\bar{X}_t^{(i)}$ to attain the reward function of the risk-neutral agent, which can be formulated as follows:

$$R_{t+1}^{(i)} = A_t \frac{p_{t+1}^{(i)} - p_t^{(i)}}{p_t^{(i)}} - \lambda 1_{A_t \in \{-1,1\}}. \tag{29}$$

Consequently, the Bellman Equation corresponding to the linear utility maximization problem equals:

$$V^*\left(S_t^{(i)}\right) = \max_{A_t} \mathbb{E}\left[A_t \frac{p_{t+1}^{(i)} - p_t^{(i)}}{p_t^{(i)}} - \lambda 1_{A_t \in \{-1,1\}} + \gamma V^*\left(S_{t+1}^{(i)}\right) \mid S_t^{(i)}, A_t\right] \tag{30}$$

where γ is the discount factor of the future rewards. Note that the reward function defined in Equation 29 considers the transaction cost rate (λ) of short or long actions performed at time t.

3.6.2 Logarithmic Utility

However, maximizing pure profits could lead to a risk profile which is not desired due to the fact that most investors exhibit some level of risk aversion and for that reason they strive to maximize their risk-adjusted returns which aligns with the principle of mean-variance theory [Markowitz, 1952]. Therefore, we will focus on an agent with logarithmic utility, where the agent has constant relative risk aversion (CRRA), which means the agent is equally averse to proportional changes in wealth regardless of the initial wealth level [Bodnar et al., 2020]. Moreover, due to the concavity of the logarithmic function, the agent has diminishing marginal utility, which implies that as the wealth level of the agent grows, the utility gained from an additional unit of wealth decreases. This also means that, unlike the case of linear utility, losses will affect the utility of an agent more than gains of the same magnitude and small incremental gains are preferred to (sudden) large increases in wealth. Hence, instead of rewarding the agent net returns made on a trade, we will reward the logarithm of the net returns to the agent, which translates into:

$$R_{t+1}^{\log,(i)} = \log\left(1 + \left(A_t \frac{p_{t+1}^{(i)} - p_t^{(i)}}{p_t^{(i)}} - \lambda 1_{A_t \in \{-1,1\}}\right)\right). \tag{31}$$

Note that we transformed the net profits by dividing them by $\bar{X}_t^{(i)}$ and adding 1 such that the agent's reward is still defined for negative trade returns - effectively, we are using logarithmic returns.

Remark: There is one case where the logarithmic return would not be defined: when the net return

$$\frac{p_{t+1}^{(i)} - p_t^{(i)}}{p_t^{(i)}} - \lambda 1_{A_t \in \{-1,1\}}$$

is less than or equal to -1. Assuming λ is between 0 and 1000 basis points (bps), where 1000 bps corresponds to a transaction cost rate of 10% of a trade position, the price of asset i would need to increase (or decrease) by more than 90% (for 1000 bps) or 100% (for 0 bps) in a single day with the agent shorting (longing) for this issue to arise. Since we focus on highly liquid futures

contracts with underlying assets that have high market capitalizations, where the transaction cost rate typically ranges from 0 to 50 bps, the probability of an asset changing by 90% or 100% in a single day is extremely small and should not cause problems. However, for less liquid markets like penny stocks or cryptocurrencies, or with larger intervals between steps, such as yearly steps, the probability of such drastic price changes is much higher, and hence it is crucial to note the frequency of the trading steps.

In these cases, the logarithmic reward design needs adjustment. One naive approach is to set a fallback value for the logarithmic reward to handle instances where it becomes undefined. For example, setting this fallback value of the reward to -1 reflects the negative impact of such occurrences, while setting it to zero avoids a negative penalty but still imposes an opportunity cost by not providing a positive reward. Setting the fallback value of the reward to zero takes into account the small chance of these events occurring and prevents them from disproportionately impacting the agent's long-term rewards. This approach effectively sets a threshold for penalization, ensuring that extreme negative returns do not disproportionately affect the agent's learning process. While this approach does introduce a discrepancy where, for example, a loss of -99% (for λ of 0 bps) is penalized more than a loss of -100% or -200%, the reasoning is that the probability of such extreme losses, in our setting, is very low. Thus, the agent should not be overly penalized for these rare occurrences.

Alternatively, one could consider using exponential utility, which is always defined for all real numbers and avoids issues with non-positive values. Note that the risk aversion in exponential utility is absolute rather than relative. This means that the agent's level of risk aversion remains constant regardless of returns, leading to different trading behaviors compared to logarithmic utility, where risk aversion is constant but applied proportionally to returns.

We will implement the naive approach with the fallback value set to zero to ensure that even if such situations arise with their small probability, it will not affect the agent's behavior too much and ensures that the logarithmic reward is still defined.

$$R_{t+1}^{\log,(i)} = \begin{cases} \log\left(1 + \left(A_t \frac{p_{t+1}^{(i)} - p_t^{(i)}}{p_t^{(i)}} - \lambda \mathbb{1}_{A_t \in \{-1,1\}}\right)\right) & \text{if } A_t \frac{p_{t+1}^{(i)} - p_t^{(i)}}{p_t^{(i)}} - \lambda \mathbb{1}_{A_t \in \{-1,1\}}) > -1 \\ 0 & \text{if } A_t \frac{p_{t+1}^{(i)} - p_t^{(i)}}{p_t^{(i)}} - \lambda \mathbb{1}_{A_t \in \{-1,1\}} \leqslant -1 \end{cases} \tag{32}$$

Thus, the Bellman Equation corresponding to the logarithmic utility maximization problem of the risk-averse agent then becomes as in Equation 33.

$$V^*\left(S_t^{(i)}\right) = \max_{A_t} \mathbb{E}\left[R_{t+1}^{\log,(i)} + \gamma V^*\left(S_{t+1}^{(i)}\right) \mid S_t^{(i)}, A_t\right] \tag{33}$$

3.6.3 Budget Constraint and Stop Loss Policy

We introduce a stop rule within the episodes in which the agent interacts with the environment, acting as a budget constraint or stop-loss mechanism. If the wealth allocated to asset i falls below ω% of the initial wealth allocated $W_0^{(i)}$ with $\omega \in (0, 1)$, all trading activities cease, the agent receives no further rewards, and the episode is concluded. By halting trading if the wealth level drops below ω% of the initial wealth, the agent is incentivized to not only maximize its utility but also to preserve capital such that it does not exceed its allocated budget and manage volatility, potentially mitigating risk. The underlying idea is that $T_{earlystopping}$ will be smaller than T in Equation 5. When early stopping is triggered, the agent achieves a smaller V^* because potential future rewards are lost. Note that it is important to use a stop loss threshold ω that

effectively exposes poor policies without being overly restrictive, ensuring that the agent can still converge and complete a full trading cycle after being trained.

Consequently, a criterion for the convergence of the agent's policy is to ensure that this budget constraint is applied during training while still aiming to maximize terminal wealth. To assess whether the stopping rule is enforced, we measure how many trading days the agent participates in, denoted by treward. For instance, if the training dataset consists of T days, each episode should ideally last T days, resulting in treward equaling T. If treward is less than T, it indicates that the agent's policy is not fully aligned with the stopping rule. Therefore, when treward consistently reaches T during training, this means that one of the convergence criteria has been met. This approach ensures that the agent learns to operate within constraints while still seeking to maximize its utility.

Additionally, this stopping rule also enhances the training process efficiency by terminating episodes early when the rule is triggered due to the policy being infeasible, thus conserving computational resources.

3.7 Benchmark Models

In order to assess the performance of the built DQN models with different trading rules and distinct reward functions, we consider multiple baseline models.

3.7.1 Buy & Hold

The first baseline model is the buy-and-hold strategy, commonly used as a benchmark for trading algorithms aiming to profit from short-term price fluctuations [Kampouridis and Otero, 2017; Tran et al., 2023; Kumar et al., 2023]. This strategy aligns with the Efficient Market Hypothesis, as noted by Malkiel [2003], who argues that the buy-and-hold strategy is optimal for average investors. In this strategy, assets are bought at the beginning of the investment horizon and held until the end, potentially resulting in profit due to price differences. Additionally, we assume that there are no transaction costs at the time of buying or selling for this strategy.

3.7.2 Momentum Strategy

The second baseline model is based on the classical time series momentum strategy, which involves analyzing past price movements to predict future trends. This strategy assumes that assets that have performed well in the past will continue to do so and those that have performed poorly will continue to perform poorly. Moskowitz et al. [2012] empirically demonstrated that this time series momentum strategy provided additional returns over a passive long position for most instruments they tested. In our implementation, we look at the 1-year return and generate trading signals based on these returns. If the yearly return is positive, the strategy takes a long position; if negative, it takes a short position. The yearly return for the asset i at time t is calculated as:

$$r^{(i)}_{t-252,t} = \frac{p^{(i)}_t - p^{(i)}_{t-252}}{p^{(i)}_{t-252}} \tag{34}$$

Therefore, the direction of a trade will be as follows:

$$A_t = \text{sign}(r_{t-252,t}) \tag{35}$$

To manage risk, we apply volatility scaling to our position sizes, ensuring that portfolio volatility remains within a target range. Thus, a trade position at time t for asset i will be:

$$\bar{X}_t^{(i)} = \frac{\sigma_{target}}{\sigma_{t-1}^{(i)}\sqrt{252}} \tag{36}$$

where $\sigma_{t-1}^{(i)}$ is the 3-month exponentially moving average standard deviation of the daily returns.

3.7.3 MACD Signal

The last baseline model is based on technical analysis, specifically the volatility-normalized MACD strategy discussed in Section 3.4.2 and proposed by Baz et al. [2015]. This trend-following momentum indicator relies on the difference between two exponential moving averages to identify the signal direction and is normalized by an estimate of the volatility of returns to determine the signal strength. To enhance the model, we use the equal-weighted average of multiple long- and short-term moving averages to create the final signal, as proposed by Baz et al. [2015]:

$$A_t = \text{sign}\left(\text{MACD}_t^{(i)}\right) = \text{sign}\left(\sum_{k=1}^{3} \text{MACD}_{t,S_k:L_k}^{(i)}\right) \tag{37}$$

with $S_k \in \{8, 16, 32\}$ and $L_k \in \{24, 48, 96\}$. The position size is then determined by the formula:

$$\bar{X}_t^{(i)} = \left| \frac{\text{MACD}_t^{(i)} \exp\left(\frac{\text{MACD}_t^{(i)2}}{4}\right)}{0.89} \right| \frac{\sigma_{target}}{\sigma_{t-1}^{(i)}\sqrt{252}} \tag{38}$$

As explained in Section 3.4.2, the first term is the trend factor and the second term is the volatility targeting term, also called the volatility factor.

Note that for the latter two strategies, we do not scale the size of the positions with a fixed fraction of the wealth level at time t, unlike the position sizes of the DQN agent. Instead, we assume the fixed fraction of the wealth to be equal to one. This will not affect our analysis of cumulative rewards, relative returns, or performance metrics like the Sharpe ratio, as these metrics compare relative rather than absolute differences. Furthermore, similarly to the volatility target of the position sizing rule of the DQN agents, we use an annualized volatility target of 15%.

3.8 Deep Q-Network

Deep Q-Network (DQN) merges traditional Q-learning with deep neural networks to manage continuous, high-dimensional state spaces [Mnih et al., 2013]. DQN employs a deep neural network to estimate the optimal action-value function which predicts the expected returns or utility after taking an action (A_t) in a given state (S_t), based on a time-independent trading strategy (π). The agent selects actions aimed at maximizing the estimated Q-values using an epsilon-greedy approach, highlighting the model-free characteristic of DQN, where learning occurs through direct interaction with the environment.

3.8.1 Neural Network

The neural network serves as the approximator of the state-action value, often called the Q-function, in this framework. The network receives as input the state of the environment and outputs normalized expected rewards for each possible action, often called the Q-values. The

neural network is trained using a loss function that determines the difference between the estimated Q-values using the current Q-function and the target Q-values. Denoting the parameters of the neural network including its layers and biases as θ, the target Q-value is computed using the Bellman Equation:

$$Q'_{\theta}(S_t, A_t) = r + \gamma \max_{A_{t+1}} Q_{\theta}(S_{t+1}, A_{t+1}) \tag{39}$$

where r is the immediate reward for taking action A_t in state S_t, γ is the discount factor of future rewards, and $\max_{A_{t+1}} Q_{\theta}(S_{t+1}, A_{t+1})$ is the maximum expected future reward of the next state S_{t+1} considering all available actions A_{t+1}. The loss function for our model is defined as the mean squared error between the estimated Q-value and the target Q-value:

$$L(\theta) = \mathbb{E}\left[\left(Q_{\theta}(S, A) - Q'_{\theta}(S, A)\right)^2\right]. \tag{40}$$

Through the loss function, the Q-function is updated by tuning θ to minimize the loss function using an optimizer. This process continues until the Q-values stabilize or a predetermined number of episodes are completed.

3.8.2 Multi-Layer Perceptron

In the existing literature, such as Lim et al. [2019] and Lindemann et al. [2021], it is demonstrated that LSTM and Recurrent Neural Networks (RNNs) exhibit superior performance when handling time-series data due to their ability to maintain memory over long sequences. This memory capability allows them to capture complex temporal dependencies, making them more flexible for sequential data tasks. However, there is also evidence [He et al., 2023; Oukhouya and El Himdi, 2023] suggesting that Multi-Layer Perceptrons (MLPs) can effectively capture useful signal features if the appropriate features are selected and a reasonable number of lags are included. This allows MLPs to effectively represent and capture the temporal dynamics of time-series data. Moreover, MLPs are significantly easier to train in terms of computational power, which can outweigh the benefits of the memory capabilities of LSTMs and RNNs. The idea is that if we can develop a Deep Q-Network (DQN) using an MLP that performs desirably in terms of risk management and profitability, we know that it can be improved using LSTMs or RNNs architectures, which are more flexible due to their ability to handle long-term dependencies in time series data.

In this paper, the neural network implemented is a multi-layer perceptron (MLP), where the network consists of an input layer, which takes in features derived from the state of the environment that accommodate the total number of lags and features per lag of the learning environment [Rumelhart et al., 1986]. It has two hidden layers, each consisting of 28 units, which are meant for capturing the nonlinear relationships in the data. The network output layer produces three values, corresponding to the estimate of the action value function for each of the three possible actions of the agent. Moreover, the optimizer that minimizes the loss function is the Adam optimizer based on stochastic gradients [Kingma and Ba, 2014]. To avoid overfitting, we use dropout [Srivastava et al., 2014] and L2 regularization [Cortes et al., 2012] methods. Dropout involves randomly dropping a fraction of neurons during training to prevent them from co-adapting too much, while L2 regularization adds a penalty to the loss function based on the squared magnitude of the model parameters, encouraging smaller and more generalized weights.

3.8.3 Learning Algorithm Overview

- **Action Selection**: At each time step, the agent selects an action from the action space using an epsilon-greedy policy. During the learning phase the agent exploits the accumulated knowledge with probability $1 - \epsilon$ and explores with probability ϵ by selecting a

random action. ϵ starts with a value 1 and diminishes gradually with a certain decay rate in each episode, allowing for more exploitation, until it reaches a specified minimum value ϵ_{min}.

- **Experience Storage**: The agent stores experiences as tuples (S_t, A_t, R_{t+1}, S_{t+1}, $done$) in a replay memory, with $done$ either indicating the set off of a stopping rule or the end of an episode. An episode in this context refers to a complete sequence of actions from the initial state to a terminal state, encompassing a full cycle of trading activities within the model. For example, if the training data spans from January 1, 2011, to January 1, 2012, each episode would contain 252 time steps, corresponding to the number of financial trading days within the training data.

- **Learning from Memory**: The agent periodically randomly samples batches of experiences from the memory to update the Q-function. This random sampling enhances learning stability and prevents catastrophic forgetting i.e. bias towards recent trends and sequences. Thus, improving the generalization ability of the agent.

3.8.4 Validation and Preventing Overfitting

The effectiveness of the trained policy is evaluated by setting the exploration rate (ϵ) to zero, ensuring that the agent strictly adheres to the policy based on the learned Q-values. This evaluation is performed on a new, previously unseen dataset. The agent's performance is assessed by its risk-adjusted returns, reflecting the learning progression and the effectiveness of the policy. We validate the model at the end of each training episode. The training process is stopped when the training score continues to increase, but the validation score decreases for more than 10 consecutive episodes. This early stopping method helps to avoid overfitting. Additionally, during training, we limit the reduction of ϵ to a minimum of 0.10 to further prevent overfitting to the training data. Random sampling of the agent's experiences, as mentioned earlier, also helps mitigate overfitting. We present the pseudocode for the Deep Q-Network (DQN) algorithm in Appendix A.1.

The policy is trained on the training set, while hyperparameter tuning and early stopping are performed with respect to the performance on the validation set. We also use a separate testing set, which serves as the out-of-sample dataset. This testing set is not used for tuning parameters or early stopping but is reserved solely for evaluating the final performance of the model. To tune the hyperparameters of the DQN model, we use a combination of trial and error and an informal random search of different parameters. This approach helps us find the optimal hyperparameters that lead to the best performance on the validation set.

4 Experimental Setup

In this section, we will introduce the data used to train and test the Deep Q-Network models with distinct reward functions and position sizing rules, specify the parameters and hyperparameters of the models, and outline the approach for analyzing the results of the different models in comparison to the benchmark models.

4.1 Data Definition

We examine the daily closing prices for 20 different futures contracts across four asset classes: Commodities, Equity, Fixed Income, and Foreign Exchange. This data, obtained from the Bloomberg Terminal, spans from January 1, 2006, to April 22, 2024. Below is an overview of how we have divided the data into training, validation, and testing sets.

- **Training Set:** January 1, 2006 $\rightarrow$ April 1, 2018
- **Validation Set:** April 2, 2018 $\rightarrow$ January 22, 2019
- **Testing Set:** January 23, 2019 $\rightarrow$ April 22, 2024

Naturally, the training set is the largest of the three datasets, as the DQN agent requires an extensive dataset to interact with the environment and learn an optimal policy. The validation and testing sets are primarily used for evaluating the trained models, which necessitates fewer interactions. The validation set is used to tune the hyperparameters of the models and perform early stopping when convergence has been reached (Section 3.8.4). The testing set, which serves as the out-of-sample data and is used for testing the DQN agents, was chosen because it includes multiple market disruptions such as the Covid-19 crisis, the Russian-Saudi oil price war in March 2020, and the Russian invasion of Ukraine in 2022. This selection allows for a more in-depth analysis of the performance of the reinforcement learning agents during these crisis moments.

Bloomberg Ticker	Description	Category
CC1 Comdty	COCOA Futures	Commodity
KW1 Comdty	Wheat Futures	Commodity
GI1 Index	ICE Commodity Futures Index	Commodity
SI1 Comdty	Silver Futures	Commodity
GC1 Comdty	Gold, 100 oz Futures	Commodity
CL1 Comdty	Crude Oil Futures	Commodity
C 1 Comdty	Corn Future	Commodity
MXEF Index	MSCI Emerging Markets Index	Equity
MXUS Index	MSCI USA Index	Equity
MXEU Index	MSCI Europe Index	Equity
NDX Index	NASDAQ 100 Stock Index	Equity
SPX Index	S&P 500 INDEX	Equity
ES1 Index	SP500 Mini Index	Equity
RX1 Comdty	Euro-Bund (5Y) Futures	Fixed Income
OE1 Comdty	Euro-Bobl (10Y) Futures	Fixed Income
TY1 Comdty	US Treasury Note 10Y Futures	Fixed Income
EUR BGN Curncy	Euro	Foreign Exchange
JPY BGN Curncy	Japanase Yen	Foreign Exchange
CAD BGN Curncy	Canadian Dollar	Foreign Exchange
AUD BGN Curncy	Australian Dollar	Foreign Exchange

Table 1: List of all considered future contracts and their corresponding asset class categories.

4.2 Training, Testing, and Relevant Parameters

Following our defined methodology, we trained an agent for each asset using each of the strategies and then tested the performance of these portfolios on an out-of-sample test set. Each asset was assigned an initial wealth of \$1000, with a stop loss threshold set at 40%. Furthermore, we used the following fixed fractions of wealth for each position sizing rule: $\mu_V = 0.5$ and $\mu_M = 1$, in order to bring the average positions of the different rules to approximately the same magnitude for comparison purposes. An overview of the parameters introduced is presented in Table 2. Similarly, the DQN algorithm involves a number of hyper-parameters, as mentioned in Section 3.8. We performed an informal random search for these parameters with respect to the validation set to find the most optimal values. Due to our computational capacity constraints, it was not feasible to perform a more extensive hyper-parameter optimization, such as a grid search, often used for supervised learning models. An overview of the hyper-parameters used is given in Table 3.

Parameter	μ_V	μ_M	W_0	λ	ω
Value	0.5	1	1000	10bp	0.40

Table 2: Overview of the parameters: μ_V and μ_M are the fractions of wealth used when applying volatility targeting and the ensemble MACD-Volatility targeting position sizing rules, respectively. W_0 is the initial wealth allocated to a DQN agent for each asset. λ is the static transaction cost rate when the agent performs a long or short action relative to the position size, $\bar{X}_t^{(i)}$. Lastly, ω is the stop-loss threshold relative to the initial wealth; if the wealth level of a DQN agent for a given asset falls below ωW_0, an episode is stopped (Section 3.6.3).

Hyperparameter	α_{LR}	Optimizer	Batch Size	γ	Memory Size	ϵ_{min}	ϵ_{decay}
Value	0.0001	Adam	512	0.95	5,000	0.10	0.995

Table 3: Hyperparameters used in the DQN model. α_{LR} represents the learning rate of the neural network, which controls the adjustment of model weights during training. The optimizer refers to the stochastic optimization algorithm applied to minimize the loss function. Batch Size denotes the number of samples used in each iteration of random memory replay, stabilizing training, and preventing overfitting. The discount factor, γ, determines the importance of future rewards and is chosen such that the agent is relatively long-term oriented. Note that a lower γ would place more importance on immediate rewards. Memory Size is the size of the replay memory to store the agent's experiences. ϵ_{min} is the minimum value of the epsilon-greedy approach, where the agent decides on actions based on accumulated knowledge/experience with probability ϵ, exploiting current knowledge, and alternatively randomly chooses an action with probability $1 - \epsilon$, allowing exploration. We do not allow ϵ to go to zero to avoid overfitting on the training data. Lastly, ϵ_{decay} is the rate at which ϵ decreases with each episode of training.

4.3 Portfolio Construction

Furthermore, we create simple equally weighted portfolios consisting of all the assets in each asset class and also an aggregate portfolio consisting of all the assets. For a portfolio of N assets, the daily return of that portfolio is then given by:

$$R_t^{(P)} = \frac{1}{N} \sum_{i=1}^{N} R_t^{(i)}. \tag{41}$$

The same procedure will be also performed for benchmark models of Section 3.7. Note that these $R_t^{(i)}$ are the actual simple returns net of transaction costs, as defined in Equation 29. Therefore, irrespective of the underlying reward design, including the DQN agent based on logarithmic utility, the returns used in Equation 41 are derived from Equation 29 and then averaged.

4.4 Performance Metrics

We compute various types of metrics for the constructed portfolios to measure the performance of these portfolio's corresponding to different agents or benchmark models, similar to how the current literature does this [Théate and Ernst, 2021; Zhang et al., 2019; Lim et al., 2019].

4.4.1 Cumulative Return

The total cumulative daily returns measures the overall gain or loss of an investment over a period by compounding the daily returns. The formula is as follows:

$$\text{Total Cumulative Daily Return} = \left[\prod_{t=1}^{T}\left(1 + R_t^{(P)}\right)\right] - 1 \tag{42}$$

where $R_t^{(P)}$ is the daily return of the given portfolio for day t and T is the total number of return periods.

4.4.2 Sharpe Ratio

The Sharpe ratio quantifies the risk-adjusted returns of an investment portfolio by dividing the excess return of the portfolio by its volatility. A general guideline is that a portfolio with a Sharpe ratio greater than 1 is considered satisfactory, while a Sharpe ratio above 2 is regarded as very good. The annualized Sharpe ratio for a portfolio is given by:

$$\text{Sharpe Ratio} = \frac{\sqrt{252} \times \left(\frac{1}{T}\sum_{t=1}^{T}(R_t^{(P)} - R_f)\right)}{\sigma_p}$$

where:

- $\frac{1}{T}\sum_{t=1}^{T}(R_t^{(P)} - R_f)$ is the mean excess daily return of the portfolio.
- σ_p is the standard deviation of the portfolio returns.
- T is the total number of return periods.

The standard deviation of the portfolio returns is calculated as:

$$\sigma_p = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(R_t^{(P)} - \overline{R})^2}$$

where $\overline{R}$ is the average portfolio return over T trading days.

4.4.3 Sortino Ratio

The Sortino ratio quantifies the risk-adjusted return of a portfolio like the Sharpe ratio but only offsets the downside volatility instead of both upside and downside volatility like the Sharpe ratio. The annualized Sortino ratio is given by:

$$\text{Sortino Ratio} = \frac{\sqrt{252} \times \left(\frac{1}{T}\sum_{t=1}^{T}(R_t^{(P)} - R_f)\right)}{\sigma_d}$$

where:

- $\frac{1}{T}\sum_{t=1}^{T}(R_t^{(P)} - R_f)$ is the mean excess daily return of the portfolio.
- σ_d is the downside deviation of the portfolio returns.

- T is the total number of return periods.

The downside deviation is calculated as:

$$\sigma_d = \sqrt{\frac{1}{T}\sum_{t=1}^{T} \min(R_t^{(P)} - R_f, 0)^2}$$

4.4.4 Maximum Drawdown (MDD)

The MDD represents the maximum percentage drop in the value of the portfolio from a peak to the trough before a new peak is reached. It is an indicator of the downside risk throughout the history of the portfolio up until time T. The MDD of the portfolio is given by the following:

$$\text{MDD} = \min\left(\frac{C_t - P_t}{C_t}\right) \quad \text{for } t = 1, 2, \ldots, T$$

where:

- C_t is the peak value before the drawdown.
- P_t is the trough value during the drawdown period.
- T is the total number of return periods.

Similarly, the annual MDD is an indicator of the downside yearly risk and is given by:

$$\text{Annual MDD} = \min\left(\frac{C_{t,y} - P_{t,y}}{C_{t,y}}\right) \quad \text{for } t = 1, 2, \ldots, T_y$$

where:

- $C_{t,y}$ is the peak value before the drawdown within year y.
- $P_{t,y}$ is the trough value during the drawdown period within year y.
- T_y is the total number of trading days within year y.

4.4.5 Calmar Ratio

The Calmar ratio quantifies the risk-adjusted return of an investment with respect to its maximum drawdown over a given interval. The annualized Calmar ratio is given by:

$$\text{Calmar Ratio} = \frac{252 \times \left(\frac{1}{T}\sum_{t=1}^{T}(R_t^{(P)} - R_f)\right)}{|\text{MDD}|}$$

where:

- $\frac{1}{T}\sum_{t=1}^{T}(R_t^{(P)} - R_f)$ is the mean excess daily return of the portfolio.
- MDD is the maximum drawdown as defined above.
- T is the total number of return periods.

4.4.6 P&L Ratio

The P&L Ratio measures the performance of a portfolio by dividing the average profit by the average loss over a given period. It provides an indication of the profitability of the portfolio's trading strategy by comparing the average gains on profitable trades to the average losses on losing trades. The P&L ratio is given by:

$$\text{P\&L Ratio} = \frac{\text{Average Profit}}{|\text{Average Loss}|}.$$

The Average Profit and Average Loss of the portfolio are calculated as follows:

$$\text{Average Profit} = \frac{\sum_{t=1}^{T} R_t^{(P)} \cdot \mathbb{I}(R_t^{(P)} > 0)}{\sum_{t=1}^{T} \mathbb{I}(R_t^{(P)} > 0)}$$

$$\text{Average Loss} = \frac{\sum_{t=1}^{T} R_t^{(P)} \cdot \mathbb{I}(R_t^{(P)} < 0)}{\sum_{t=1}^{T} \mathbb{I}(R_t^{(P)} < 0)}$$

where:

- $R_t^{(P)}$ is the portfolio return on day t.
- $\mathbb{I}(R_t^{(P)} > 0)$ is an indicator function that is 1 if $R_t^{(P)}$ is positive and 0 otherwise.
- $\mathbb{I}(R_t^{(P)} < 0)$ is an indicator function that is 1 if $R_t^{(P)}$ is negative and 0 otherwise.
- T is the total number of return periods.

4.4.7 Percentage of Positive Returns

Similar to the P&L Ratio, the percentage of positive returns is indicative of the performance of a portfolio over a given period. The percentage of positive returns is given by:

$$\text{Percentage of Positive Returns} = \frac{\sum_{t=1}^{T} \mathbb{I}(R_t^{(P)} > 0)}{T} \times 100\%$$

where:

- $\mathbb{I}(R_t^{(P)} > 0)$ is an indicator function that is 1 if $R_t^{(P)}$ is positive and 0 otherwise.
- T is the total number of return periods.

Additionally, we also analyze the annualized expected returns, and annualized standard deviation of the portfolio's to gain insight into the profitability of the investment portfolios. Note, we will assume R_f is zero in this paper.

5 Experimental Results

In this section, we examine the performance of the previously described reinforcement learning model and apply sensitivity analysis to answer several key questions. First, we investigate the training stability of DQN agents and determine whether overfitting is adequately prevented. Next, we explore the appropriate type of position sizing for a DQN approach to ensure robust risk management, considering that DQN outputs only discrete actions (long, hold, or short) rather than actual sizes. We present two cases: one where position sizes are determined by volatility targeting, and another using the volatility-normalized momentum indicator MACD in combination with volatility targeting. Furthermore, we assess the model's effectiveness across different risk profiles by evaluating two distinct types of traders: a risk-neutral trader and a risk-averse trader. Each type of trader is applied to various futures contracts within four separate asset classes, with a separate model trained for each asset. We also create simple equally weighted portfolios for each asset class and one comprehensive portfolio comprising all the considered futures. A passive buy-and-hold strategy, a momentum-based strategy, and a technical trading strategy using MACD serve as baseline models for comparison purposes. Following this, we present the results and discuss the findings in terms of the efficacy of the risk profiles, the position size strategies, and the overall performance of the reinforcement learning model for each asset class. Additionally, we analyze the stability of the optimal DQN strategies during several market disruptions. Lastly, we apply sensitivity analysis on trading costs to assess how the models perform under increased transaction costs.

5.1 Training Stability and Overfitting Analysis

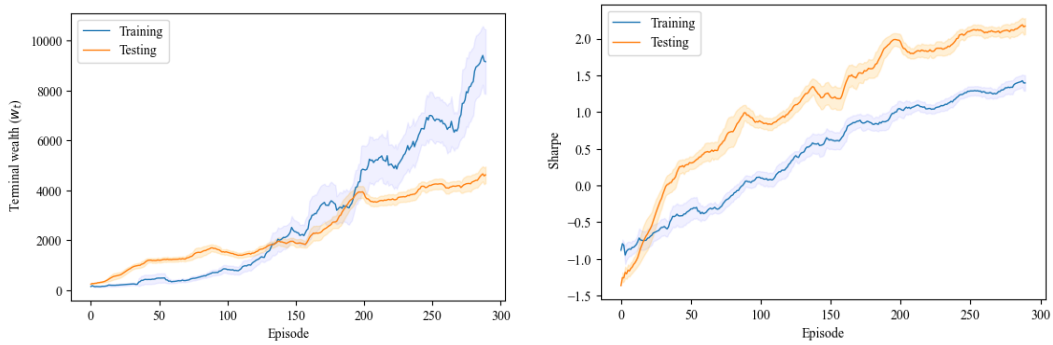
Before delving into the analysis of the performance of the various DQN agents and comparing them with the benchmark models, we evaluate the training stability of our DQN agents to ensure that the models are learning effectively and do not overfit the training data. To illustrate this, we present the terminal wealth level and the Sharpe ratio per episode for both the training and testing phases for one representative asset and one of the DQN strategies. Figure 4 depicts the rolling average of performance measures over 20 iterations of the DQN agent, which is trained using a logarithmic reward design and volatility targeting as the position sizing rule, with respect to the number of training episodes for the MSCI Emerging Markets Index futures. This is comparable to a typical run of the DQN algorithm for the various asset types and reward functions. The blue line represents the training phase, and the orange line represents the testing phase. The shaded areas around the lines represent the 95% confidence interval of these performance metrics, indicating the variability in performance. Recall that each asset is allocated an initial wealth of \$1000.

Firstly, the figure demonstrates that terminal wealth increases gradually during both the training and testing phases, indicating that the DQN agents are learning effectively over time. The close proximity and similar trends of the training and testing lines, especially with respect to the Sharpe ratio, suggest that the overfitting tendency of the reinforcement learning agent is managed properly. If overfitting were present, there would typically be a noticeable gap between training and testing performance, which serves as the expected performance when the model is tested on unseen data. In the case of overfitting, we would expect to see training performance being much better in terms of Sharpe ratio, or a scenario where training performance with respect to terminal wealth consistently increases while testing performance decreases. This is not the case here. Additionally, note that the performance of the test set being temporarily superior to that of the training set is not an error in both charts. This phenomenon illustrates a major obstacle in quantitative trading: the constantly changing distribution of financial returns [Mertzanis, 2014]. The testing and training sets do not share the same distributions, and in this case, it seems the testing period is an easier to trade environment and more profitable for the

given asset. Additionally, the relatively stable and narrow confidence interval, especially for the testing set, indicates consistent generalization ability across different episodes.

Lastly, recall that we introduced a budget constraint that served as a stop-loss policy, whereby the agent was not allowed to fall below a certain threshold, ω, of its initial wealth as otherwise all trading activity would cease (Section 3.6.3). One of the convergence criteria was that the agent had to learn to comply with this stop-loss threshold and prevent the termination of its trading activity due to insolvency. We measured this by introducing a variable, $\mathrm{treward}$, which should reach the terminal period T of the training set (in our case $\mathrm{T} = 3405$). As shown in Figure 5, $\mathrm{treward}$ consistently reaches the terminal period in the final phase of training, indicating that the budget constraint criteria is met.

Overall, the training stability suggests that the DQN agents are robust and do not exhibit signs of overfitting.



(a) Terminal Wealth W_T (b) Sharpe ratio

Figure 4: Terminal wealth level and Sharpe ratio per episode during training and testing phases for the MSCI Emerging Markets Index future contract. The shaded areas represent the 95% confidence interval of the performance metrics per episode.

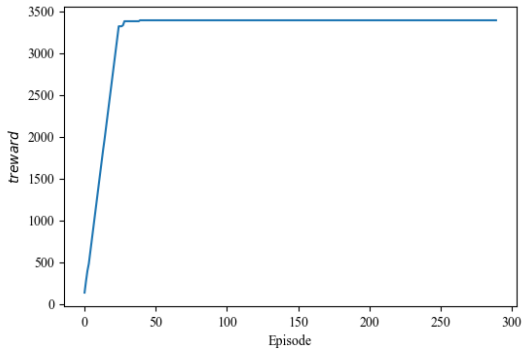


Figure 5: The rolling average of $\mathrm{treward}$ with respect to episodes during the training phase for the MSCI Emerging Markets Index futures contract.

5.2 Performance

In order to assess the performance of the different DQN agents and position sizing rules across various asset classes, compared to the benchmark models introduced in the methodology, we present key metrics that explore three performance components: risk-adjusted returns, profitability, and drawdown management, as shown in Table 4. Additionally, the cumulative trade returns of the different strategies are depicted in Figure 6.

		Sharpe	Sortino	P&L	MDD	AADD	Annualised Return	Annualised Std Dev	Calmar Ratio	Positive Percentage
Commodity	**Logarithmic**									
	Volatility Target	**3.10**	**3.84**	1.21	-0.70	-0.47	**0.48**	**0.13**	0.68	51.86
	Ensemble	2.90	3.49	1.19	-0.63	-0.46	0.45	**0.13**	0.71	52.15
	Linear									
	Volatility Target	3.01	3.60	**1.23**	**-0.57**	**-0.40**	0.45	**0.13**	**0.79**	**54.63**
	Ensemble	2.65	3.14	1.21	-0.59	-0.48	0.39	**0.13**	0.66	53.17
	Benchmarks									
	Buy & Hold Benchmark	0.18	0.19	0.92	-1.57	-0.62	0.13	0.31	0.08	52.12
	Sign Benchmark	0.31	0.59	1.05	-0.94	-0.79	-0.02	0.29	-0.03	48.90
	MACD Benchmark	0.83	1.65	1.05	-0.85	-0.71	0.14	0.30	0.16	51.10
Equity	**Logarithmic**									
	Volatility Target	**2.30**	**3.31**	**1.31**	**-0.27**	**-0.20**	**0.36**	0.14	**1.34**	53.10
	Ensemble	2.09	3.04	1.30	-0.33	-0.22	0.31	**0.13**	0.93	53.10
	Linear									
	Volatility Target	1.89	2.54	1.29	-0.31	-0.22	0.26	0.13	0.86	51.20
	Ensemble	1.91	2.79	1.31	-0.31	-0.21	0.28	0.14	0.89	52.59
	Benchmarks									
	Buy & Hold Benchmark	0.65	0.88	0.97	-0.42	-0.29	0.11	0.18	0.25	**54.03**
	Sign Benchmark	-0.54	-0.65	0.90	-0.73	-0.57	-0.10	0.16	-0.14	50.29
	MACD Benchmark	0.25	0.34	1.03	-0.30	-0.24	0.02	0.17	0.08	50.66
Fixed Income	**Logarithmic**									
	Volatility Target	0.58	0.75	1.11	-0.15	-0.11	0.01	**0.02**	0.09	47.48
	Ensemble	0.19	0.23	**1.23**	-0.14	-0.12	0.00	**0.02**	0.03	46.17
	Linear									
	Volatility Target	0.61	0.80	**1.23**	**-0.11**	**-0.08**	0.01	**0.02**	0.12	47.70
	Ensemble	0.55	0.64	1.14	-0.17	-0.12	0.01	**0.02**	0.07	46.68
	Benchmarks									
	Buy & Hold Benchmark	-0.52	-0.73	0.97	-0.29	-0.17	-0.03	0.05	-0.10	47.29
	Sign Benchmark	**2.86**	**4.35**	1.11	-0.24	-0.12	0.16	0.05	**0.69**	**57.98**
	MACD Benchmark	1.65	2.29	1.12	-0.22	-0.14	0.08	0.05	0.38	54.61
Foreign Exchange	**Logarithmic**									
	Volatility Target	1.78	2.22	1.20	-0.15	**-0.12**	0.09	0.05	0.62	51.64
	Ensemble	1.76	**2.35**	1.20	-0.19	**-0.12**	**0.10**	0.06	0.55	49.16
	Linear									
	Volatility Target	1.46	1.95	1.18	-0.21	-0.16	0.09	0.06	0.42	49.02
	Ensemble	**1.87**	**2.35**	**1.22**	**-0.14**	**-0.12**	**0.10**	0.05	**0.70**	49.67
	Benchmarks									
	Buy & Hold Benchmark	0.47	0.33	0.98	-0.22	-0.18	0.01	**0.03**	0.04	50.95
	Sign Benchmark	0.89	1.10	1.01	-0.39	-0.28	0.05	0.06	0.13	**55.42**
	MACD Benchmark	0.96	1.19	1.02	-0.30	-0.21	0.05	0.06	0.18	53.00
Total	**Logarithmic**									
	Volatility Target	**3.86**	**3.94**	1.22	-0.70	-0.47	**0.28**	**0.06**	0.39	**56.75**
	Ensemble	3.64	3.65	1.24	-0.63	-0.46	0.26	**0.06**	0.40	55.07
	Linear									
	Volatility Target	3.61	3.47	**1.26**	**-0.57**	**-0.40**	0.24	**0.06**	**0.43**	56.09
	Ensemble	3.42	3.36	1.24	-0.59	-0.48	0.23	**0.06**	0.38	55.58
	Benchmarks									
	Buy & Hold Benchmark	0.36	0.37	1.00	-1.57	-0.62	0.07	0.13	0.04	55.56
	Sign Benchmark	0.44	0.59	0.97	-0.94	-0.79	0.01	0.12	0.01	51.10
	MACD Benchmark	1.03	1.49	1.04	-0.85	-0.71	0.08	0.12	0.10	52.64

Table 4: The performance metrics of DQN Agents with logarithmic and linear utility functions, along with the position sizing rule types (Volatility Targeting, Ensemble) across four asset classes: commodities, equity, fixed income, and foreign exchange. This includes metrics for a total portfolio consisting of all futures contracts corresponding to these asset classes. Additionally, the performance metrics of the passive Buy & Hold strategy, Momentum Strategy, and Technical MACD Strategy, serving as the benchmarks, are provided.

5.2.1 Risk-Adjusted Returns

To evaluate the risk-adjusted performance of the strategies, we examine the Sharpe and Sortino ratios. All DQN strategies significantly outperform the benchmark models across all distinct asset classes, except in the fixed income class. This underscores the effectiveness of reinforcement learning models for quantitative trading compared to passive buy-and-hold strategies and technical analysis-based strategies. For fixed income, momentum-based time series strategies Sign(R) and the MACD technical trading benchmark strategies outperform the DQN agents. This discrepancy may be due to the DQN models' state representation not capturing the dy-

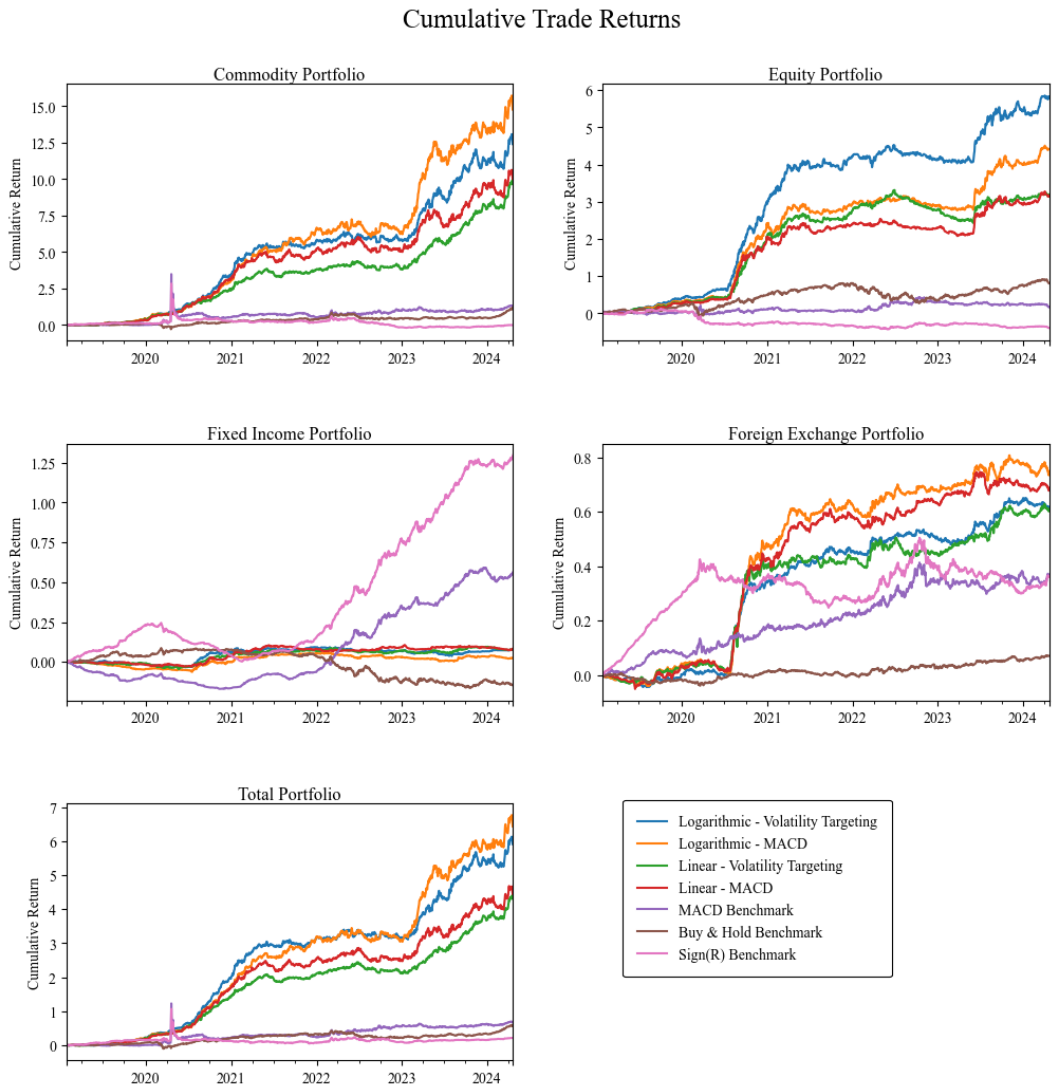


Figure 6: Overview of the cumulative returns of the different asset class portfolio's and the different type of strategies.

namics of these assets as effectively as it does for other asset classes. Further investigation is needed, possibly by incorporating more relevant features for the fixed income market, such as interest rates of the underlying fixed income assets.

Regarding utility functions, logarithmic utility performs better in terms of risk-adjusted returns for commodities and equities, while for fixed income and foreign exchange, linear utility demonstrates slightly better performance. This is particularly evident in fixed income, where the logarithmic utility using the ensemble position sizing rule has a Sharpe ratio of 0.19 and a Sortino ratio of 0.23, whereas the linear counterpart has a Sharpe ratio of 0.55 and a Sortino ratio of 0.64. For foreign exchange, linear utility with ensemble position sizing leads to the best risk-adjusted ratios. However, when using logarithmic utility supplemented with the volatility targeting rule, it still results in better risk-adjusted performance than linear utility with volatility targeting. This suggests that the risk-neutral trader corresponding to the linear utility has some favorable char-

acteristics for these specific asset classes, possibly due to their mean-reverting nature.

For the total portfolio, logarithmic utility with volatility targeting achieves the highest Sharpe and Sortino ratios. All DQN agents outperform benchmarks in the total portfolio, with the best-performing benchmark (MACD) having a Sharpe ratio of 1.03, while the worst-performing DQN agent (linear utility plus ensemble position sizing) achieves a Sharpe ratio of 3.42.

5.2.2 Profitability

To assess profitability, we analyze the cumulative returns, P&L, percentage of trades with positive returns, and annualized return vs. annualized volatility. All DQN strategies are profitable across all asset classes and outperform the benchmarks, except in fixed income, similar to our observations with the risk-adjusted performance. This is particularly evident when looking at the cumulative returns, where DQN strategies excel in capitalizing on trending markets such as commodities and equities. However, they encounter more challenges with mean-reverting assets, especially in fixed income, where they are outperformed by the momentum time series strategy and the MACD signal benchmarks. In foreign exchange, the DQN agents do outperform the benchmarks but not by as much as they do in commodities or equities.

When comparing utility types, logarithmic utility leads to more optimal results for commodities and equities across all trading rules. Furthermore, it also yields the best outcomes in fixed income and foreign exchange when combined with volatility targeting despite linear utility strategies identifying more winning trades in equities (higher percentage of positive trade returns), logarithmic utility plus volatility targeting leads to the best cumulative returns overall. Alternatively, the volatility-trend ensemble position sizing rule performs better in commodities and foreign exchange. A notable insight into the position sizing rules is that although the logarithmic utility with volatility targeting has a higher number of trades with positive returns for fixed income and foreign exchange, the P&L of logarithmic utility with ensemble position sizing is higher for these asset classes. This suggests that when the agent of the latter makes correct predictions, potentially due to a stronger trend signal, the position sizes are larger, and when the agent makes incorrect predictions, due to a weaker trend signal, the position sizes are smaller. This highlights the efficacy of adding a trend strength scalar to the scalar of volatility targeting.

5.2.3 Drawdown Management

In terms of drawdown management, measured by maximum drawdown (MDD), annualized average drawdown (AADD), and the Calmar ratio, all DQN strategies outperform the benchmarks in both maximum drawdown and annualized average drawdown. Linear utility generally outperforms logarithmic utility in drawdown metrics, except for equities, where logarithmic utility performs better. This is somewhat unexpected, as losses weigh more with logarithmic utility due to its concavity compared to a linear utility function. Therefore, one would expect that the logarithmic reward design would penalize the DQN agent more in scenarios with significant drawdowns, leading to better drawdown management. However, this does not seem to be the case based on the observed behavior of the risk-averse agent in the out-of-sample test compared to the risk-neutral agent. Lastly, with respect to position sizing, volatility targeting leads to slightly better drawdown management overall.

5.2.4 Performance Across Asset Classes

For commodities and equities, both utility functions outperform benchmarks, with logarithmic utility capturing greater gains and providing more robust risk management. Volatility targeting is more robust in terms of risk-adjusted returns, while the ensemble method achieves the highest cumulative returns. In fixed income, both utility types are best combined with volatility targeting.

However, as aforementioned, two of the benchmarks outperform the DQN agents, indicating that the constructed DQN agents are less effective for fixed income assets. For foreign exchange, linear utility slightly outperforms logarithmic utility and the ensemble position sizing rule is most robust in terms of both risk management and profitability, unlike for the other asset classes where volatility targeting is superior.

5.2.5 Total Portfolio

When examining the total portfolio, all DQN agents perform significantly better than the given benchmark strategies. The risk-averse agents demonstrate superior performance in terms of risk-adjusted returns and profitability, while risk-neutral agents exhibit better drawdown management, especially when supplemented with volatility targeting.

5.2.6 Hierarchy of Position Sizing Rules

Then, if we look at only the logarithmic utility that yields the best results on the total portfolio, we can deduce that volatility targeting, which adjusts position sizes based on market volatility to maintain a consistent level of risk, seems more robust in terms of risk-adjusted returns, whilst the ensemble position rule, which adds a trend factor to the volatility scaling, seems to be best for profitability.

To summarize, our analysis reveals a nuanced ranking in terms of profitability and risk mangement for the various risk profiles and position sizing rules across the asset classes. Key findings are:

- All DQN strategies outperform the passive Buy & Hold strategy, providing evidence against the Efficient Market Hypothesis.
- DQN trading strategies demonstrate robust risk-adjusted performance across all asset classes except fixed income.
- A risk-averse DQN agent with logarithmic utility achieves optimal risk management and profitability when supplemented with the appropriate position sizing rule with respect to the given market.
- The ensemble position sizing rule increases profitability but reduces risk-adjusted performance for commodities, equities, and fixed income, compared to regular volatility targeting.
- The ensemble position sizing rule is optimal for risk management and profitability in the foreign exchange market.
- Position sizing rules significantly influence the risk management effectiveness of a DQN agent. Notably, The risk-neutral agent exhibits better drawdown mangament when supplemented with the right position sizing rule than the risk-averse agent.

5.3 Crisis Analysis of Crude Oil

The ability to manage risks effectively during black swan events and crises is crucial for portfolio management and quantitative trading due to the profound impact these events can have on performance and risk management strategies. Extreme market turbulence can lead to substantial losses if risks are not adequately mitigated. Consequently, this section analyzes the performance of DQN strategies under such conditions to assess their robustness and resilience, ensuring they can withstand and recover from severe market disruptions. Specifically, the focus is on the performance of these strategies on crude oil futures during the COVID-19 pandemic and the Russian-Saudi price war. Crude oil is highly sensitive to geopolitical events and economic disruptions, affecting related markets including currencies, equities, and other commodities. Figure 7

illustrates the cumulative returns of optimal DQN trading strategies, while Table 5 presents the performance metrics for the crude oil contract during the given market disruptions. All strategies, except Buy & Hold, employ volatility scaling to maintain a constant risk exposure. We compared the efficacy of optimal DQN strategies based on logarithmic utility with the two position sizing rules against our benchmark models.

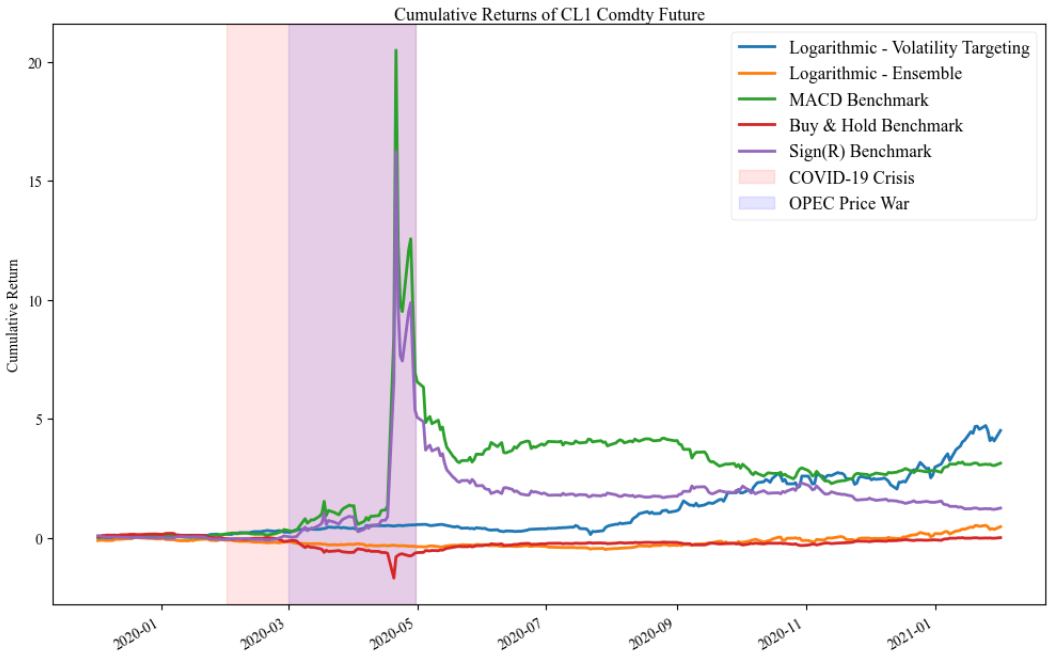


Figure 7: Cumulative returns of DQN strategies and benchmarks for crude oil futures contracts during the COVID-19 pandemic and the Russian-Saudi price war market disruptions.

Strategy	Sharpe	Sortino	P&L	MDD	AADD	Annualized Return	Annualized Std Dev	Calmar Ratio	Positive Percentage
Logarithmic									
Volatility Targeting	2.82	4.96	1.32	-0.27	-0.15	2.98	0.54	10.88	56.21
Ensemble	1.05	1.55	1.17	-0.51	-0.23	0.53	0.55	1.05	50.65
Benchmarks									
MACD Benchmark	0.55	2.11	1.28	-0.85	-0.70	0.36	1.52	0.42	51.10
Sign(R) Benchmark	0.37	1.37	1.23	-0.94	-0.79	0.02	1.52	0.02	49.56
Buy & Hold Benchmark	-0.31	-0.32	0.71	-1.57	-0.57	0.08	1.52	0.05	52.12

Table 5: Performance metrics of the optimal DQN strategies and the corresponding benchmark models for the crude oil futures.

Analyzing the cumulative returns and performance metrics during this period, we observe that the DQN strategies experienced varied performance during these disruptions, with notable differences in volatility and recovery patterns. Specifically, during these disruptions, the DQN agent using the volatility position sizing rule exhibited strong risk-adjusted returns, with a Sharpe ratio of 2.82 and a Sortino ratio of 4.96. It managed to limit drawdowns efficiently, as indicated by a MDD of -0.27, which is significantly lower than for the other strategies. Furthermore, the strategy achieved a high annualized return of 2.98 (with volatility of 0.54) and a Calmar ratio of 10.88, reflecting robust performance. The percentage of positive returns of 56.21% indicates that the strategy generated positive returns more than half the time. In contrast, the DQN agent relying on the ensemble position sizing rule had a lower Sharpe ratio of 1.05 and a Sortino ratio of 1.55, indicating moderate risk-adjusted returns. Its MDD of -0.51 suggest greater drawdowns compared to the volatility targeting strategy. Nevertheless, with an annualized return of 0.53

(with volatility of 0.55) and a Calmar ratio of 1.05, the strategy demonstrated suboptimal yet tolerable performance during a period of high market turbulence, with a percentage of positive returns equal to 50.65%.

On the other hand, the benchmarks exhibited higher volatility and larger drawdowns. The MACD benchmark, although it showed growth during the disruptions, had a Sharpe ratio of 0.55 and a Sortino ratio of 2.11. While the Sharpe ratio indicates lower risk-adjusted returns compared to the DQN strategies, the relatively high Sortino ratio suggests that the MACD benchmark managed downside risk reasonably well. However, its overall performance remains ambiguous and not as strong as the DQN strategy with volatility targeting. The MACD benchmark performed better than the DQN agent using ensemble position sizing in terms of downside risk, but it was worse in terms of drawdown management, as evidenced by its MDD of -0.85, which reflects significant risk exposure. The Sign(R) benchmark and Buy & Hold benchmark had even higher drawdowns and lower risk-adjusted returns, with the Buy & Hold benchmark showing negative performance metrics (Sharpe: -0.31, Sortino: -0.32).

To conclude this section, effective risk management during market disruptions involves not just achieving growth but also controlling volatility and drawdowns. The DQN strategies demonstrated varying degrees of success in this regard. The logarithmic - volatility targeting strategy, with its lower volatility and effective drawdown management, suggests better risk control compared to benchmarks. The logarithmic - ensemble strategy, while performing comparatively well concerning the benchmarks, showed higher drawdowns, indicating potential areas for improvement in risk management. In contrast, the benchmarks, particularly the Buy & Hold strategy, displayed significant volatility and drawdowns, indicating higher risk exposure. While the MACD benchmark showed some growth, its higher drawdowns and lower Sharpe ratio suggest less effective risk management compared to the DQN strategies.

5.4 Crisis Analysis of Total Portfolio

In this section, we investigate the performance of different trading strategies concerning the total portfolio and market disruptions during the entire out-of-sample period (24 January 2019 to 22 April 2024). We highlight the Russian invasion of Ukraine, which occurred during this period and significantly impacted the world economy. The graph in Figure 8 depicts the cumulative returns of the trading strategies, including benchmark strategies and the optimal DQN strategies based on logarithmic utility. The corresponding performance metrics are presented in the bottom panel of Table 4.

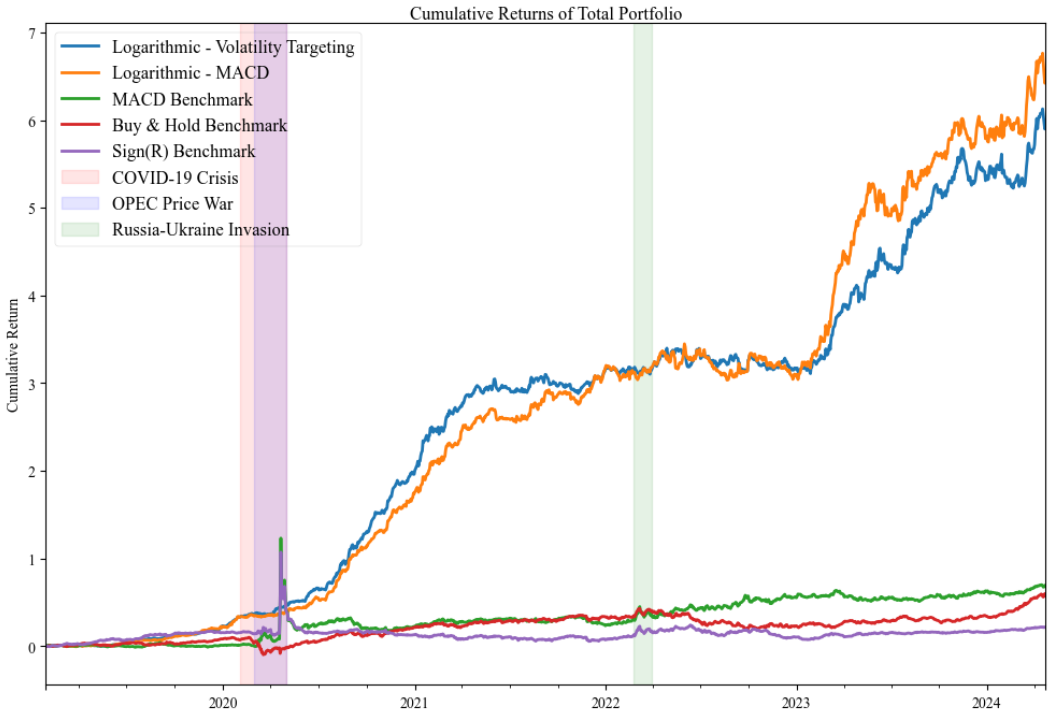


Figure 8: Cumulative returns of DQN strategies and benchmarks for the total portfolio during the entire out-of-sample period, with highlighted market disruptions: COVID-19, OPEC price war, and Russia-Ukraine invasion.

Analyzing the total portfolio reveals several key insights, similar to the crude oil analysis:

- The logarithmic-volatility targeting strategy exhibited the highest risk-adjusted returns and effective drawdown management, consistent with its performance in the crude oil analysis.
- Both logarithmic strategies (volatility targeting and ensemble) showed superior risk-adjusted returns compared to the benchmarks in terms of Sharpe and Sortino ratios, as well as in managing drawdowns more effectively.
- The benchmarks exhibited higher volatility and larger drawdowns, with the Buy & Hold strategy performing the worst in terms of risk-adjusted returns and drawdown management.

However, analyzing the total portfolio also provides additional insights:

- The inclusion of the Russian invasion of Ukraine as a disruption event offers a broader context for evaluating strategy resilience. Before the invasion, DQN strategies grew faster than benchmarks. During the invasion, cumulative returns across different strategies showed

minimal fluctuations. After the invasion, DQN strategies began to increase their cumulative returns, while benchmarks did not show significant improvement. The Buy & Hold strategy decreased post-crisis and recovered much later than the DQN strategies. Among benchmarks, the MACD strategy performed best, while the Sign(R) strategy showed the worst profitability.

- The MACD benchmark, despite showing some growth, continued to demonstrate lower risk-adjusted returns compared to the DQN strategies, reinforcing observations from the crude oil analysis.
- The Buy & Hold strategy consistently underperformed, highlighting its vulnerability to market disruptions due to its passive nature.
- The Sign(R) benchmark exhibited poor performance with substantial drawdowns and minimal profitability, consistent with the crude oil analysis.
- A notable difference in the total portfolio analysis is that the DQN agent based on logarithmic utility and the ensemble position sizing rule demonstrated better drawdown management compared to the DQN agent using logarithmic utility and volatility targeting. This is evident from its lower MDD and higher Calmar ratio, indicating more effective risk control during market disruptions.
- Additionally, the analysis reveals that both DQN strategies showed quicker recovery post-market shocks compared to the benchmarks, as evidenced by higher final cumulative returns and effective drawdown management. This suggests these strategies capitalized on market recoveries more efficiently.

Overall, DQN strategies, particularly the logarithmic-volatility targeting strategy, demonstrated superior performance compared to benchmarks. These strategies achieved higher risk-adjusted returns and effectively controlled drawdowns during significant market disruptions such as the COVID-19 pandemic, the Russian-Saudi price war, and the Russian invasion of Ukraine. This analysis underscores the value of DQN strategies for investors seeking stability and resilience in volatile markets across a diverse portfolio. The persistent patterns observed in both the crude oil and total portfolio analyses further validate the robustness of these strategies in managing risks and achieving returns during extreme market conditions. Additionally, the improved drawdown management of the logarithmic-ensemble strategy in the total portfolio highlights its potential advantage in broader market contexts compared to the case where we observed suboptimal drawdown management for crude oil futures.

5.5 Sensitivity Analysis of Transaction Costs

In this section, we perform a sensitivity analysis of our optimal DQN agents constructed with a logarithmic reward design with respect to increased transaction costs. The results of the previous sections were based on a transaction cost rate of 10 basis points (bp) per trade, which is not unreasonable for highly liquid markets such as foreign exchange or futures contracts of major commodities. However, for contracts with lower trading volumes, transaction costs could be significantly higher, especially for retail traders, impacting the profitability of trading strategies. Therefore, we investigate how quickly the profitability of the constructed DQN agents decreases with increasing transaction costs. We present the P/L ratio, Sharpe ratio, Sortino ratio, and Maximum Drawdown (MDD) of the individual assets through box plots in Figure 9, allowing us to observe differences between various futures contracts. We considered transaction rates of 10 bp, 20 bp and 30 bp with respect to each traded position.

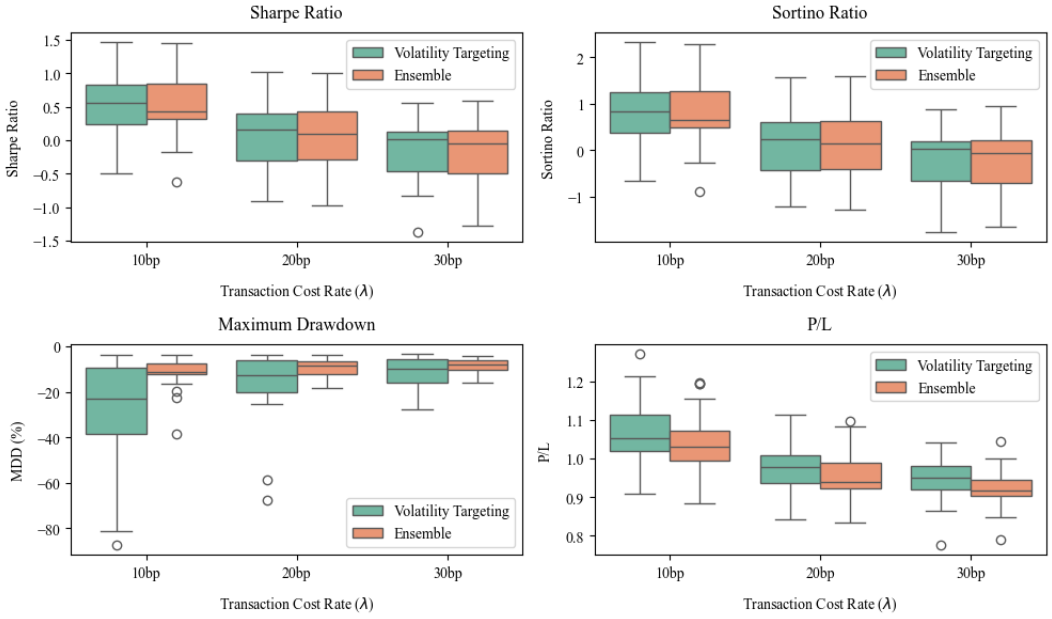
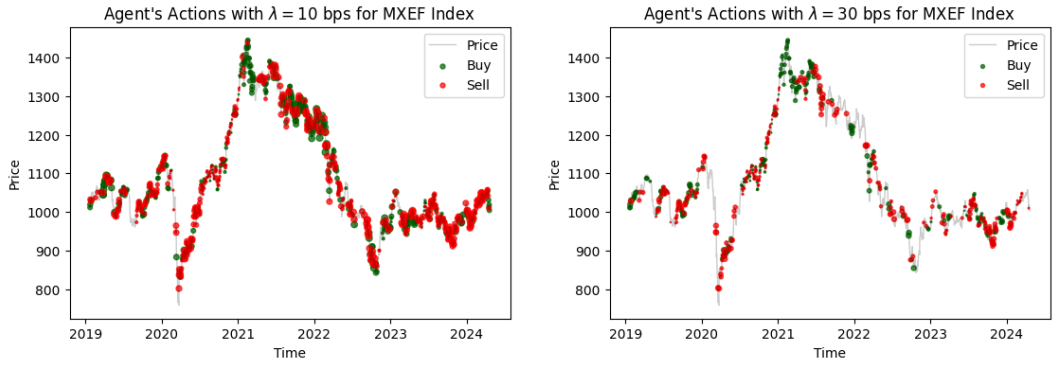


Figure 9: Sensitivity analysis of DQN strategies to transaction costs, showing the impact on Sharpe Ratio, Sortino Ratio, Maximum Drawdown (MDD), and P/L ratio for transaction cost rates of 10 bp, 20 bp, and 30 bp with respect to each traded position.

For the Sharpe Ratio, both the volatility targeting and ensemble position sizing rules experience a decrease as transaction costs increase, indicating a reduction in risk-adjusted returns. The volatility targeting strategy maintains a median Sharpe ratio above zero for transaction costs of 10 bp and 20 bp, but it approaches zero at 30 bp. The ensemble strategy shows a similar decreasing trend, starting with a lower median Sharpe ratio than the volatility targeting strategy and falling below zero by 30 bp. This implies that both strategies experience reduced risk-adjusted returns with higher transaction costs, with the volatility targeting strategy performing slightly better under increasing costs.

In terms of the Sortino Ratio, the pattern is similar. The Sortino ratio for both strategies declines as transaction costs rise. The variability of the volatility targeting strategy increases at higher costs, meaning more variability in managing the downside risk across the individual assets. The ensemble strategy follows a similar pattern, with a noticeable drop in the Sortino ratio as costs rise and higher variability and lower median values compared to the volatility targeting strategy.

Examining the Maximum Drawdown (MDD), an interesting trend arises. For the volatility targeting strategy, the MDD decreases as transaction costs increase, with the variability in drawdowns also decreasing. The smaller boxes and absence of outliers in the 30 bp case suggest improved drawdown management. Similarly, the ensemble strategy also shows a decrease in MDD with higher transaction costs, with the variability in drawdowns decreasing in a manner similar to the volatility targeting strategy. Furthermore, the ensemble strategy demonstrates better drawdown management overall compared to the volatility targeting strategy. This implies that the improved drawdown management for both strategies as transaction costs increase may be due to DQN agents becoming better at managing drawdowns when penalized (more) for excessive trading. The ensemble strategy, in particular, shows superior drawdown management. Higher transaction costs seem to encourage more prudent trading behavior, reducing the frequency of changing trade directions and potentially leading to better drawdown management. For example, if we consider the ensemble position sizing rule for the MXEF Index, which we examined for training stability, and look at the frequency of the "hold" action on the same testing set, the agent subject to higher transaction costs performs holding for 823 days out of 1371 days of trading. In contrast, the agent subject to lower transaction costs of 10 basis points performs holding for only 359 days out of the same 1371 days of trading, and thus the latter changes the direction of its trades more often (see Figure 10).



(a) Lower λ (b) Higher λ

Figure 10: This figure displays the actions and position sizes with respect to the future's price of the DQN agent using logarithmic utility and an ensemble position sizing rule for two different transaction cost rates for MSCI Emerging Markets Index futures contract. Buy actions are represented in green, while sell actions are depicted in red. The size of the markers indicates the position size. Days without a green or red marker imply a 'hold' action.

Lastly, the P/L Ratio declines significantly as transaction costs increase for both strategies. For the volatility targeting strategy, the median P/L ratio drops below 1.0 by 30 bp, indicating reduced profitability. The ensemble strategy shows a similar trend, with a more pronounced decrease in the P/L ratio. The median P/L ratio for the ensemble strategy falls below 1.0 at 20 bp and continues to decline at 30 bp. This indicates that both strategies' profitability is adversely impacted by higher transaction costs, with the ensemble strategy's profitability decreasing more rapidly, highlighting its greater sensitivity to transaction costs.

In conclusion, the sensitivity analysis indicates that the increase in transaction costs significantly affects the performance of both DQN strategies. The volatility targeting strategy generally exhibits better resilience in terms of risk-adjusted returns and profitability compared to the ensemble strategy. However, both strategies show improved drawdown management as transaction costs increase, suggesting that DQN agents may adapt by reducing excessive trading, thus improving drawdown management. The ensemble strategy, in particular, demonstrates better overall drawdown management compared to the volatility targeting strategy. These findings imply that while

DQN strategies can be robust under low transaction costs, their performance can deteriorate quickly as costs rise. This sensitivity should be considered when deploying these strategies in real-life trading environments, especially in markets with higher transaction costs. Additionally, analyzing individual contract metrics rather than portfolio-level metrics provides deeper insights into the performance and robustness of DQN strategies across different assets.

6 Conclusion

The primary objective of this thesis was to develop and evaluate Deep Q-Networks (DQN) for quantitative trading and to examine the impact of reward designs based on different utility functions, both with and without risk aversion. Additionally, we aimed to investigate the impact of position sizing rules to optimize these reinforcement learning algorithms, particularly for DQN, which have a limited action space and are less tractable for determining position size along with the direction. Position sizing is often overlooked in current trading strategy literature, both for traditional and RL-based financial trading research [Scholz, 2012]. We used volatility targeting, which scales position sizes inversely relative to the estimated volatility of returns, maintaining constant exposure. Various studies have shown that this approach leads to a more robust risk profile and reduces the impact of left-tailed events. Specifically, Zhang et al. [2019] showed the effectiveness of volatility targeting for various RL agents. Furthermore, we introduced a position sizing rule combining volatility targeting with a trend scalar based on the strength of the MACD signal as introduced by Baz et al. [2015]. This combination was assessed to determine if it results in more desirable characteristics than volatility targeting alone.

To achieve this, we developed DQN agents with various reward designs and experimented with them using 20 highly liquid futures contracts across four different asset classes. We evaluated the performance of these agents against three baseline models: the Buy & Hold strategy advocated by the Efficient Market Hypothesis, a momentum time-series strategy based on the sign of returns from a year prior, and a technical trading strategy using linear combinations of volatility-normalized moving average convergence divergence indicators across different time scales. For consistency with the existing literature, we analyzed several performance metrics related to risk-adjusted returns, profitability, and drawdown management throughout the out-of-sample period from the end of January 2019 to April 2024. Given that this period included market disturbances, we examined the impact of these disturbances on the performance of different strategies, assessing whether the DQN agents exhibited more desirable properties than the baseline models. Lastly, we addressed the transaction cost aspect of these RL trading strategies and their sensitivity to increasing transaction fees. This consideration is particularly relevant for retail traders or assets with less liquid markets which are subject to higher transaction costs.

With this approach, we were able to systematically assess the robustness of DQN for quantitative trading.

Firstly, we found that regarding the training of the Deep Q-Network-based agents:

- Training the DQN agents exhibited stable learning and effective prevention of overfitting through measures such as the dropout method, L2 regularization of the loss function of the Multi-Layer Perceptron, setting a minimum epsilon value for the epsilon-greedy approach of exploration versus exploitation, random sampling of past experiences to prevent catastrophic forgetting, and the implementation of early stopping when training performance increases but validation performance starts to decrease.

Then, regarding the performance of the trained DQN agents, we observed the following:

- The DQN agents were profitable and outperformed the benchmarks in all markets except fixed income, indicating that the DQN agents are less robust for such mean-reverting markets. This warrants further investigation. One potential reason is the different nature of the fixed income class. We hypothesize that using a more appropriate state space for the fixed income market or focusing on less frequent trading, such as every three months, which would decrease the amount of noise, could improve performance. Additionally, basing the rewards of the DQN agents on logarithmic utility generally proved better than using linear

utility for the analyzed period and assets. However, in some markets, linear utility showed favorable characteristics when combined with the right position sizing rule. For example, linear utility with volatility targeting led to the best performance for the fixed income class, while linear utility with the ensemble technique performed better for foreign exchange. This highlights the importance of considering the market type and investor characteristics, such as the degree of risk aversion, when determining the position sizing rule to complement the DQN architecture.

- For logarithmic utility, which generally led to better performance, we analyzed the effect of various position sizing rules. Volatility targeting was optimal in terms of risk-adjusted returns, while the ensemble position sizing technique yielded the best results in terms of profitability but not risk-adjusted returns. Our findings align with Zhang et al. [2019], who concluded that volatility targeting is optimal for their DQN agents. However, our ensemble volatility-trend position sizing rule led to better drawdown management over the analyzed horizon. During market disruptions, such as the COVID-19 pandemic and the Russian-Saudi oil price war, DQN agents with both position sizing rules displayed more desirable performance in terms of drawdown management and exhibited less variability in the total portfolio compared to the baseline models. Additionally, the ensemble position sizing rule showed better resilience overall, but for crude oil specifically, volatility targeting was superior in drawdown management. This further supports the notion that the risk profile and asset type are important determinants of the appropriate position sizing rule for DQN agents, significantly impacting their performance. It also underscores the versatility of reinforcement learning, particularly during market disruptions, compared to conventional trading strategies.

- Regarding the sensitivity analysis of transaction costs, we assumed a base case transaction cost rate of 10 basis points for each traded position. When costs increased to 20 and 30 basis points, there was a significant decrease in profitability and risk-adjusted returns. Volatility targeting performed slightly better in terms of risk-adjusted returns. Specifically, at 20 basis points, the median P/L of individual contracts fell below 1 for ensemble position sizing. In contrast, for volatility targeting, the median P/L fell below 1 only at 30 basis points, despite the ensemble rule being more profitable than volatility targeting at the base case of 10 basis points. Interestingly, higher transaction costs resulted in smaller drawdowns for both position sizing rules. This was possibly because agents changed exposures less frequently due to the increased penalty for changing position direction or size. This observation suggests that higher transaction costs may encourage more prudent trading behavior, reducing the frequency of changing position directions and hence decreasing drawdowns.

Thus, this study demonstrates the potential of using reinforcement learning, specifically Deep Q-Networks, for quantitative trading compared to conventional strategies, whether passive or based on technical trading. Our findings highlight significant improvements in risk management, profitability, and handling drawdowns. By utilizing utility theory to design the reward function, we observed enhanced risk-adjusted returns for risk-averse traders using a logarithmic utility function. Additionally, we identified a major shortcoming in the critic-only RL framework for trading, often overlooked in current literature. The DQN agent's optimal policy acts primarily as a timing strategy due to its limited action space, determining position direction but not size. It is crucial to complement this with an appropriate position sizing rule tailored to the asset's characteristics and the investor's risk profile. Our experiments demonstrated the impact of different position sizing rules on risk preferences. We introduced a novel position sizing rule, volatility targeting with a trend factor, which, in some cases, increased profitability compared to standard volatility targeting, although with additional risk. Our sensitivity analysis of transaction costs further reveal the DQN agent's applicability across different market types, showing profitability with up

to 20 basis points of transaction cost per traded position. These findings provide potentially a valuable stepping stone for developing more sophisticated and effective trading systems using reinforcement learning models.

7 Discussion

7.1 Limitations

There are certain limitations and aspects that need to be considered and require further investigation.

7.1.1 Constantly Changing Distribution of Financial Returns

One of the core challenges faced by the constructed Deep Q-Networks for financial trading, which is also a common challenge for most reinforcement learning models, is the constantly changing distribution of daily financial returns [Maheu and McCurdy, 2007; Mertzanis, 2014]. While our training set during the analyzed period was sufficiently representative of the out-of-sample set, leading to a stable learning process and robust performance in testing, an unrepresentative training set due to the continuously changing distribution of financial returns would impair the performance of the reinforcement learning agent. This is because the DQN relies on expected values of taking a given action with respect to the discounted cumulative rewards rather than the distribution.

Additionally, we observed that the testing set provided an easier trading environment, with the DQN agent recognizing more trading signals and achieving higher performance in the earlier episodes compared to the training set (Figure 4), a finding also noted by Théate and Ernst [2021]. In other scenarios, the opposite might occur, leading to worse performance on the testing set, which complicates both the training process and the assessment of the RL agents' generalization ability.

7.1.2 Neural Network Architecture

As mentioned in the methodology, we used a Multi-Layer Perceptron (MLP) neural architecture as the function approximator for the Q-values. Although MLPs have been shown to effectively capture the temporal dynamics of time-series data [He et al., 2023; Oukhouya and El Himdi, 2023], more advanced deep neural network architectures such as Long Short-Term Memory (LSTM) networks or Recurrent Neural Networks (RNNs) have demonstrated superior performance in terms of generalization ability for time-series data [Lim et al., 2019; Lindemann et al., 2021; Abbasimehr and Paki, 2022]. Hence, using these more appropriate neural network architectures for time-series data as the function approximator of the Q-values, could improve the generalization ability of the RL agents.

LSTMs and RNNs are specifically designed to handle sequences of data and possess memory capabilities that allow them to keep information over long sequences. This memory is achieved through recurrent connections, which enable the network to retain and utilize information from previous time steps, making them well-suited for tasks where context and temporal dependencies are crucial. However, these advanced architectures come with increased computational requirements. Utilizing such neural network architectures as the function approximator for the Deep Q-Network could potentially enhance its performance by better capturing the intricate temporal dependencies present in time-series data.

7.1.3 State Representation

The constructed DQN agents were less effective for fixed income assets, potentially due to inadequate state representation. Our current model showed limitations in this area, being outperformed by the baseline models. This contrasts with other asset classes, where the DQN

models were optimal. To improve performance for fixed income assets, more tailored state representations need to be developed. For example, incorporating features such as the underlying yield to maturity of the given fixed income futures or macroeconomic conditions such as unemployment rates or GDP growth could enhance the state representation of the DQN agents. Although DQN agents achieved satisfactory performance for other asset classes, such as foreign exchange, commodities, and equities, their performance could still be improved by refining the state representation specific to each asset class. Currently, we use the same features for each asset class. While this approach has been effective, tailoring state representations to the unique characteristics of each asset class could lead to even better performance.

7.1.4 Data Range and Timeframes

We only examined daily data from 2019 to 2024. Extending this analysis to include a broader date range and different timeframes, such as monthly or yearly data, would provide more robust testing under various market conditions. Furthermore, incorporating more fundamental analysis and macroeconomic conditions could be beneficial for larger time frames (e.g., monthly or quarterly). In contrast, analyzing very small timeframes, such as hourly data, could be beneficial for more intensive day trading. Ultimately, the appropriate timeframe depends on the use case. For example, pension funds might benefit more from monthly or quarterly trading data, while our focus on quantitative trading justified the use of daily data.

7.1.5 Logarithmic Reward Design

We considered the logarithmic utility function for highly liquid futures, noting that for less liquid futures, the effectiveness could be reduced due to higher transaction costs and reduced profitability, as indicated by our sensitivity analysis. However, the logarithmic reward design presents undefinedness problems if an asset price changes by more than 100% in one day (where $\lambda = 0$) where the agent had bet on the opposite direction. Therefore, we set zero as the fallback value of the logarithmic reward in Equation 32. In Appendix A.2, we provide descriptive statistics for the futures contracts considered across the entire dataset. The columns "min" and "max" show that daily returns have not grown to levels that would pose a problem for our models, with the exception of crude oil futures. There has been an instance where the price of crude oil declined by more than 300% in a day, during the market disruptions caused by COVID-19 and the Russian-Saudi oil price war (see Appendix A.3). This was also apparent with the baseline models, where this resulted in a sudden large increase in the cumulative returns for the Sign(R) and MACD baseline models in Section 5.4, as they likely shorted crude oil during that period.

This provides evidence against our assumption in the methodology section, where we stated that highly liquid assets with large market capitalizations are extremely unlikely to experience changes exceeding 100% in a single day. While this assumption holds true for most assets, it was not the case for crude oil futures during the past market disruptions. Although we set the reward to zero in such scenarios, it might have been more systematic to treat this lower bound as a hyperparameter and determine the optimal value through experimentation. In that case, it's essential to align the fallback value with the investor's objectives. For instance, for better suitability against black swan events, different values should be tested to identify which one leads to the best risk-adjusted returns or smallest drawdowns during such events.

Furthermore, this undefinedness issue could also arise with a higher probability with longer trading intervals or more volatile assets like penny stocks or cryptocurrencies. Adjusting the reward design by using utility functions such as exponential utility could mitigate this undefinedness problem of the reward design. In Appendix A.4, we provide a robustness check where we implement the exponential utility for crude oil, which was subject to such an occurrence, with various values for

the risk aversion parameter of the exponential utility. We observe that, in terms of risk-adjusted returns during the market disruption, the agent based on logarithmic utility remains one of the better-performing agents, though the differences compared to agents based on exponential utility are not substantial. To obtain a more accurate assessment, more disruptions should be considered.

7.1.6 Hyperparameter Tuning and Feature Selection

With access to more computational resources, implementing systematic hyperparameter tuning methods, such as grid search or heuristic-based techniques like the Genetic Algorithm [Holland, 1984] or Bayesian Optimization [Snoek et al., 2012], could significantly improve the performance of our model. However, it is crucial to carefully monitor for overfitting during this process. To effectively apply these methods, a larger validation set would be necessary to ensure the robustness of the tuning process. Currently, due to limited resources, we conducted an informal search for hyperparameters and allocated relatively more data to the testing and training sets rather than the validation set. Additionally, enhancing feature selection, such as carefully determining the number of lags of independent variables to include, could further improve the effectiveness of the reinforcement learning agents. However, this also requires additional computational power.

7.2 Future Research

Beyond these limitations, several promising areas for further research could help bridge the gap between the theory of reinforcement learning and its practical implementation in quantitative trading or asset management.

7.2.1 Alternative Reinforcement Learning Algorithms

Exploring other types of reinforcement learning algorithms or systems of RL agents that determine both position directions and sizes, or developing RL agents for portfolio optimization by determining asset fractions to own at each period, could streamline the implementation of such models. The use of actor-only or actor-critic approaches, which do not suffer from a limited action space, could also be beneficial. However, such architectures do possess other shortcomings; for instance, the actor-only framework requires differentiable reward functions, which limits flexibility in constructing reward functions. Furthermore, employing an ensemble of RL agents that perform different tasks, similar to the ensemble learning principle of random forests [Breiman, 2001], where a multitude of weak decision tree predictors collectively lead to a robust supervised learning algorithm, could enhance performance and robustness.

7.2.2 Distribution-Based Approaches

Exploring the idea of focusing on distributions rather than expected values within the DQN algorithm to handle uncertainty more effectively is an intriguing research path. As Théate and Ernst [2021] suggests, constructing reinforcement learning algorithms based on distributions rather than expected values, like the Deep Q-Network, could address the core challenge of the continuously changing distribution of financial returns, leading to better generalization ability. This approach involves predicting the entire distribution of possible returns rather than just their expected value. By focusing on the quantiles of these distributions, we can attain a more comprehensive understanding of the risks and uncertainties associated with different trading actions.

Take, for example, distributional reinforcement learning (DRL) techniques such as the C51 algorithm, which approximate the return distribution using a fixed number of discrete support points, also known as "atoms." These atoms represent different possible values that the return can take, and the algorithm predicts the probability of the return falling at each of these points.

This approach offers a more nuanced view of potential outcomes, capturing the variability and tail risks inherent in financial markets more effectively. According to Bellemare et al. [2017], such distribution-based approaches can significantly improve the stability and robustness of RL algorithms in environments with high stochasticity and non-stationarity, like financial markets.

Another example is the Quantile Regression DQN (QR-DQN), which directly estimates the quantiles of the return distribution. Unlike C51, which employs a fixed number of atoms to represent the distribution, QR-DQN uses quantile regression to predict multiple quantiles of the return distribution. This method allows for a more flexible and potentially more accurate approximation of the distribution. As demonstrated by Dabney et al. [2018], QR-DQN can lead to superior performance by providing a more detailed representation of uncertainty, which is particularly useful in the dynamic and unpredictable environment of financial markets.

A Appendix

A.1 Pseudo Code for DQN Algorithm

Algorithm 1: DQN Algorithm

Initialize the action-value function Q with arbitrary weights θ
Initialize the replay memory D with length N
Define epsilon ϵ, decay rate ϵ_{decay}, minimum epsilon ϵ_{min}
Define discount factor γ
Define number of episodes M
Define maximum time steps $T_{training}$, $T_{validation}$, $T_{testing}$
for *episode* $= 1$ **to** M **do**
    # **Training**
    Initialize trading environment and get initial state representation S_1;
    Set state $\phi_1 = \text{preprocess}(S_1)$;
    for $t = 1$ **to** $T_{training}$ **do**
        if *random number* $\leqslant \epsilon$ **then**
            Randomly pick an action a_t;
        else
            $a_t = \arg\max_a Q(\phi(S_t), a; \theta)$;
        Execute action a_t in environment;
        Observe reward r_t and next state S_{t+1};
        Update next state $\phi_{t+1} = \text{preprocess}(S_{t+1})$;
        Store tuple transition $(\phi_t, a_t, r_t, \phi_{t+1}, \text{done})$ in memory D;
        if *memory* D *has enough samples* **then**
            Randomly sample minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1}, \text{done})$ from D;
            for *each transition* $(\phi_j, a_j, r_j, \phi_{j+1}, \textit{done})$ **do**
                if *done* **then**
                    $y_j = r_j$;
                else
                    $y_j = r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta)$;
                Conduct a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to θ;
            if $\epsilon > \epsilon_{min}$ **then**
                $\epsilon = \epsilon \times \epsilon_{\text{decay}}$;
    Compute and store validation performance metrics;
    # **Validation**
    Initialize validation environment and get initial state representation S_1;
    Set state $\phi_1 = \text{preprocess}(S_1)$;
    for $t = 1$ **to** $T_{validation}$ **do**
        $a_t = \arg\max_a Q(\phi(S_t), a; \theta)$;
        Execute action a_t in validation environment;
        Observe reward r_t and next state S_{t+1};
        Update next state $\phi_{t+1} = \text{preprocess}(S_{t+1})$;
    Compute and store validation performance metrics;
    # **Testing**
    Initialize testing environment and get initial state representation S_1;
    Set state $\phi_1 = \text{preprocess}(S_1)$;
    for $t = 1$ **to** $T_{testing}$ **do**
        $a_t = \arg\max_a Q(\phi(S_t), a; \theta)$;
        Execute action a_t in testing environment;
        Observe reward r_t and next state S_{t+1};
        Update next state $\phi_{t+1} = \text{preprocess}(S_{t+1})$;
    Compute and store test performance metrics;
    if *validation score* $\downarrow$ *whilst training score* $\uparrow$ *for more than 10 consecutive episodes* **then**
        Stop training (early stopping);
        break;

A.2 Daily Market Return Statistics

	count	mean	std	min	25%	50%	75%	max
CC1 Comdty	4775.000	0.001	0.018	-0.092	-0.009	0.000	0.010	0.122
KW1 Comdty	4775.000	0.000	0.020	-0.086	-0.012	0.000	0.011	0.084
GI1 Index	4775.000	0.000	0.015	-0.120	-0.007	0.000	0.008	0.079
SI1 Comdty	4775.000	0.000	0.020	-0.178	-0.009	0.000	0.010	0.130
GC1 Comdty	4775.000	0.000	0.011	-0.094	-0.005	0.000	0.006	0.090
CL1 Comdty	4775.000	-0.000	0.055	-3.060	-0.012	0.000	0.012	0.377
C 1 Comdty	4775.000	0.000	0.019	-0.236	-0.009	0.000	0.010	0.136
MXEF Index	4775.000	0.000	0.012	-0.095	-0.006	0.001	0.006	0.106
MXUS Index	4775.000	0.000	0.012	-0.121	-0.004	0.000	0.006	0.117
MXEU Index	4775.000	0.000	0.012	-0.116	-0.005	0.001	0.006	0.101
NDX Index	4775.000	0.001	0.014	-0.122	-0.005	0.001	0.007	0.126
SPX Index	4775.000	0.000	0.012	-0.120	-0.004	0.000	0.006	0.116
ES1 Index	4775.000	0.000	0.012	-0.104	-0.004	0.000	0.006	0.141
RX1 Comdty	4775.000	0.000	0.004	-0.029	-0.002	0.000	0.002	0.027
OE1 Comdty	4775.000	0.000	0.002	-0.019	-0.001	0.000	0.001	0.019
TY1 Comdty	4775.000	0.000	0.004	-0.026	-0.002	0.000	0.002	0.036
EUR BGN Curncy	4775.000	-0.000	0.006	-0.024	-0.003	0.000	0.003	0.035
JPY BGN Curncy	4775.000	0.000	0.006	-0.038	-0.003	0.000	0.003	0.057
CAD BGN Curncy	4775.000	0.000	0.006	-0.039	-0.003	0.000	0.003	0.033
AUD BGN Curncy	4775.000	0.000	0.008	-0.070	-0.004	0.000	0.004	0.086

Table 6: Descriptive statistics (count, mean, standard deviation, and quantiles) of the daily returns for the given assets, including the training set, validation set, and out-of-sample testing set.

A.3 Crude Oil Returns

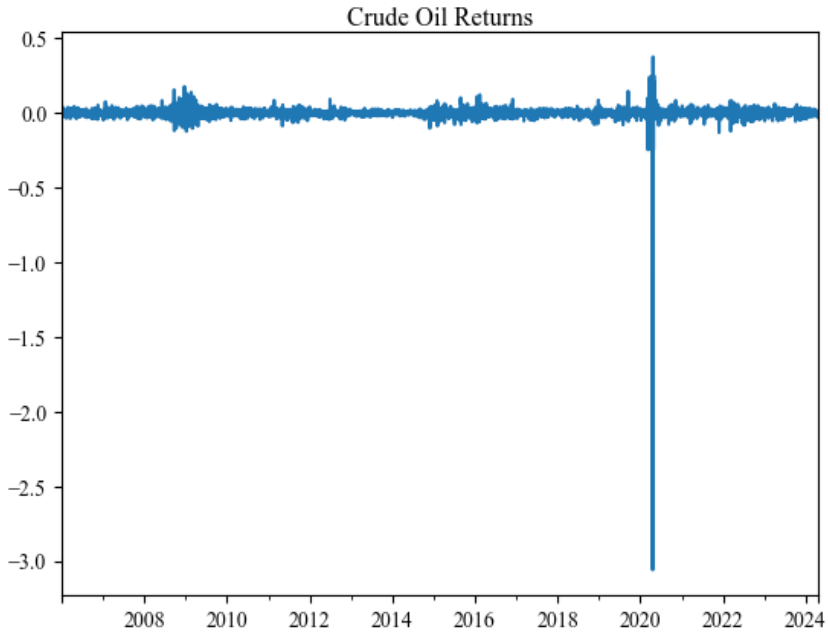


Figure 11: The changes in CL1 Comdty daily prices. Notably, there was a sudden decrease of more than 300% in crude oil futures during the first part of 2020.

A.4 Robustness Check: Logarithmic vs. Exponential Utility

In this section, we examine the robustness of the logarithmic utility design, particularly concerning the fallback value of zero defined in Equation 32 in Section 3.6.2. This fallback value is used when large losses are incurred due to extremely unlikely events where the logarithmic reward becomes undefined. When this happens, the reward is set to the fallback value of zero, effectively ensuring that we do not penalize losses over a certain threshold. The motivation behind this approach is to prevent the agent from being overly penalized for these rare events, given their low probability and magnitude, which could inappropriately affect its long-term rewards and learning ability.

We compare the logarithmic reward design with one based on exponential utility. The exponential utility does not have the same undefinedness problem as the logarithmic utility when large and sudden price changes lead to losses greater than 100% (for a transaction cost rate $\lambda = 0$), as the exponential utility is defined for all real numbers.

For crude oil, our assumption that price changes greater than 100% in a single day would not occur was violated, as depicted in Figure 11. Therefore, we consider the exponential utility to determine whether agents based on it significantly outperform agents based on logarithmic utility, including its fallback value of zero, in terms of risk-adjusted returns and drawdown management during market disruptions.

We define the reward of the DQN agents based on exponential utility as follows:

$$R_{t+1}^{exp,(i)} = 1 - \exp\left\{-\alpha\left(A_t \frac{p_{t+1}^{(i)} - p_t^{(i)}}{p_t^{(i)}} - \lambda 1_{A_t \in \{-1,1\}}\right)\right\} \tag{43}$$

where α represents the absolute constant risk aversion parameter of the exponential utility, with larger α implying higher risk aversion. Then, the corresponding maximization problem of the risk-averse agent becomes:

$$V^*\left(S_t^{(i)}\right) = \max_{A_t} \mathbb{E}\left[R_{t+1}^{exp,(i)} + \gamma V^*\left(S_{t+1}^{(i)}\right) \mid S_t^{(i)}, A_t\right]. \tag{44}$$

In Figure 12, we illustrate the shape and characteristics of the exponential utility function (with various degrees of risk aversion) and the logarithmic utility function.

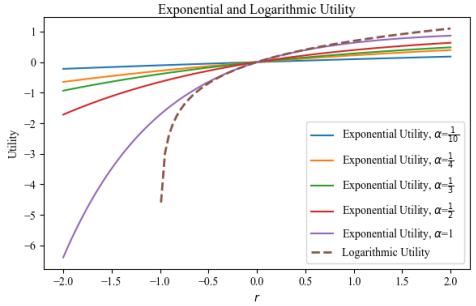


Figure 12: This graph depicts the logarithmic and exponential utility functions on which the rewards of the DQN agent can be based, with r representing the simple returns made on a trade. The logarithmic utility, characterized by constant relative risk aversion, is plotted alongside exponential utilities with various α values, characterized by constant absolute risk aversion. By comparing these functions over the domain -2 to 2, we can see that exponential utility functions are defined for r values smaller than -1, corresponding to losses greater than 100% (assuming zero bps transaction costs). For both positive and negative values of r, the exponential utility with $\alpha = 1$ is closest to the logarithmic utility, indicating that, in terms of risk aversion, the agent based on exponential utility with $\alpha = 1$ should most closely resemble the agent based on logarithmic utility.

As discussed in Section 3.6.2, the risk aversion in exponential utility is absolute rather than relative, meaning the agent's risk aversion remains constant regardless of returns. In contrast, for logarithmic utility, risk aversion is constant but applied proportionally to returns. This leads to different trading behaviors; for example, exponential utility penalizes smaller (and particularly negative) values more than logarithmic utility due to its constant absolute risk aversion. Therefore, we consider various agents with different values of the risk aversion parameter α for the exponential utility and compare them with the behavior of the agent based on logarithmic utility.

Lastly, to focus on the differences between the reward designs, particularly during periods of market disruption where crude oil experiences significant fluctuations, we only consider a single position sizing rule and present the results for volatility targeting. Below, we show the cumulative returns for the period from 2019-12-01 to 2021-02-01, which includes the COVID-19 pandemic and the Russian-Saudi oil price war, covering pre-disruption and post-disruption phases. We also provide the performance metrics of the different agents for this period, as we did earlier with logarithmic utility in Section 5.4.

Utility Function	Sharpe	Sortino	P&L	MDD	AADD	Annualized Return	Annualized Std Dev	Calmar Ratio	Positive Percentage
Exponential Utility									
$\alpha = \frac{1}{10}$	**2.93**	5.20	1.50	**-0.25**	**-0.13**	2.57	**0.47**	10.32	53.59
$\alpha = \frac{1}{4}$	2.85	**5.23**	**1.60**	-0.26	-0.13	**3.00**	0.54	**11.67**	51.96
$\alpha = \frac{1}{3}$	2.70	4.53	1.37	-0.37	-0.18	2.87	0.56	7.75	54.25
$\alpha = \frac{1}{2}$	2.63	4.50	1.49	-0.33	-0.16	2.69	0.55	8.26	51.63
$\alpha = 1$	1.85	2.88	1.33	-0.35	-0.17	1.45	0.58	4.10	51.63
Logarithmic Utility									
	2.82	4.96	1.32	-0.27	-0.15	2.98	0.54	10.88	**56.21**

Table 7: The performance metrics of various DQN agents from 2019-12-01 to 2021-02-01, covering the COVID-19 pandemic and the Russian-Saudi oil price war. The table includes pre-disruption and post-disruption periods for agents trading crude oil futures with exponential utility (various risk aversion levels) and logarithmic utility.

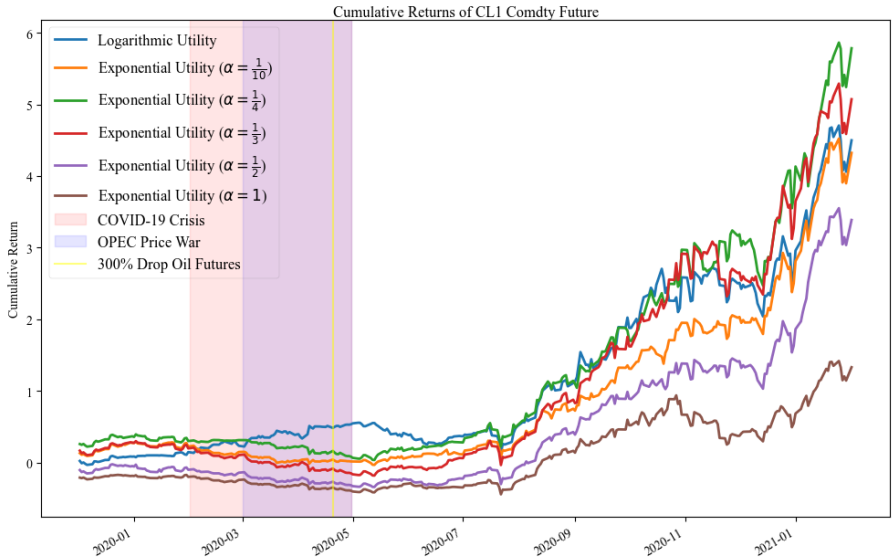


Figure 13: The cumulative returns of various DQN agents from 2019-12-01 to 2021-02-01, covering the COVID-19 pandemic and the Russian-Saudi oil price war. The chart includes pre-disruption and post-disruption phases for agents trading crude oil futures with exponential utility (various risk aversion levels) and logarithmic utility.

When looking at Table 7, we see that the exponential utility with $\alpha = \frac{1}{10}$ achieves the highest

Sharpe ratio, while the agent with $\alpha = \frac{1}{4}$ reaches the best Sortino ratio and highest profitability. The logarithmic utility ranks third, with remaining agents having higher α values ranking lower, and the agent with $\alpha = 1$, which from Figure 12 should resemble the logarithmic utility the most, ranking last. In terms of drawdown management, the logarithmic utility ranks second. Overall, with the exception of the exponential utility with $\alpha = 1$, the performance differences among the agents are comparable.

Furthermore, when looking at the cumulative returns, we see similar patterns and rankings, with similar movements between the returns of the different agents. However, during the inception of the COVID-19 pandemic and the Russian-Saudi oil price war (shaded areas), the logarithmic utility-based agent achieves higher returns and somewhat higher volatility than exponential utility functions. When the 300% drop occurs, the impact on the agent using logarithmic utility is comparable to the agents using exponential utility.

This provides evidence that the logarithmic utility, despite the extreme rare occurrence of 300% decrease in crude oil prices, remains robust and does not show significant shortcomings compared to the alternative exponential utility function, particularly regarding the fact that we do not penalize losses over a certain threshold due to the low probability of such occurrences for the logarithmic agent.

References

Jeffrey S. Abarbanell and Brian J. Bushee. Fundamental analysis, future earnings, and stock prices. *Journal of Accounting Research*, 35(1):1–24, 1997. ISSN 00218456, 1475679X. URL http://www.jstor.org/stable/2491464.

Hossein Abbasimehr and Reza Paki. Improving time series forecasting using lstm and attention models. *Journal of Ambient Intelligence and Humanized Computing*, 13(1):673–691, Jan 2022. ISSN 1868-5145. doi: 10.1007/s12652-020-02761-x. URL https://doi.org/10.1007/s12652-020-02761-x.

Salman Bahoo, Marco Cucculelli, Xhoana Goga, and Jasmine Mondolo. Artificial intelligence in finance: a comprehensive review through bibliometric and content analysis. *SN Business & Economics*, 4(2):23, Jan 2024. ISSN 2662-9399. doi: 10.1007/s43546-023-00618-x. URL https://doi.org/10.1007/s43546-023-00618-x.

Jamil Baz, Nicolas M Granger, Campbell R. Harvey, Nicolas Le Roux, and Sandy Rattray. Dissecting investment strategies in the cross section and time series. *SSRN Electronic Journal*, 2015. ISSN 1556-5068. doi: 10.2139/ssrn.2695101. URL http://dx.doi.org/10.2139/ssrn.2695101.

Stelios Bekiros. Heterogeneous trading strategies with adaptive fuzzy actor-critic reinforcement learning: A behavioral approach. *Journal of Economic Dynamics and Control*, 34(6):1153–1170, 2010. URL https://EconPapers.repec.org/RePEc:eee:dyncon:v:34:y:2010:i:6:p:1153-1170.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6 (5):679–684, 1957. ISSN 00959057, 19435274. URL http://www.jstor.org/stable/24900506.

Taras Bodnar, Dmytro Ivasiuk, Nestor Parolya, and Wolfgang Schmid. Mean-variance efficiency of optimal power and logarithmic utility portfolios. *Mathematics and Financial Economics*, 14 (4):675–698, May 2020. ISSN 1862-9660. doi: 10.1007/s11579-020-00270-1. URL http://dx.doi.org/10.1007/s11579-020-00270-1.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

João Carapuço, Rui Neves, and Nuno Horta. Reinforcement learning applied to forex trading. *Applied Soft Computing*, 73:783–794, 2018. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2018.09.017. URL https://www.sciencedirect.com/science/article/pii/S1568494618305349.

Mark Carhart. On persistence in mutual fund performance. *Journal of Finance*, 52(1):57–82, 1997. URL https://EconPapers.repec.org/RePEc:bla:jfinan:v:52:y:1997:i:1:p:57-82.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L2 regularization for learning kernels, 2012.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Anna Dreyer and Stefan Hubrich. Tail risk mitigation with managed volatility strategies. *SSRN Electronic Journal*, 2017. ISSN 1556-5068. doi: 10.2139/ssrn.3074529. URL http://dx.doi.org/10.2139/ssrn.3074529.

Eugene F. Fama. Random walks in stock market prices. *Financial Analysts Journal*, 21(5):55–59, 1965. ISSN 0015198X. URL http://www.jstor.org/stable/4469865.

Eugene F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 1970. ISSN 00221082, 15406261. URL http://www.jstor.org/stable/2325486.

Eugene F. Fama. Market efficiency, long-term returns, and behavioral finance1the comments of brad barber, david hirshleifer, s.p. kothari, owen lamont, mark mitchell, hersh shefrin, robert shiller, rex sinquefield, richard thaler, theo vermaelen, robert vishny, ivo welch, and a referee have been helpful. kenneth french and jay ritter get special thanks.1. *Journal of Financial Economics*, 49(3):283–306, 1998. ISSN 0304-405X. doi: https://doi.org/10.1016/S0304-405X(98)00026-9. URL https://www.sciencedirect.com/science/article/pii/S0304405X98000269.

Eugene F Fama and Kenneth R French. The Cross-Section of Expected Stock Returns. *Journal of Finance*, 47(2):427–465, June 1992. URL https://ideas.repec.org/a/bla/jfinan/v47y1992i2p427-65.html.

Eugene F. Fama and Kenneth R. French. Multifactor explanations of asset pricing anomalies. *The Journal of Finance*, 51(1):55–84, 1996. doi: https://doi.org/10.1111/j.1540-6261.1996.tb05202.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1996.tb05202.x.

Thomas G. Fischer. Reinforcement learning in financial markets - a survey. Technical report, 2018.

Kaijian He, Qian Yang, Lei Ji, Jingcheng Pan, and Yingchao Zou. Financial time series forecasting with the deep learning ensemble model. *Mathematics*, 11(4), 2023. ISSN 2227-7390. doi: 10.3390/math11041054. URL https://www.mdpi.com/2227-7390/11/4/1054.

John H. Holland. *Genetic Algorithms and Adaptation*, pages 317–333. Springer US, Boston, MA, 1984. ISBN 978-1-4684-8941-5. doi: 10.1007/978-1-4684-8941-5_21. URL https://doi.org/10.1007/978-1-4684-8941-5_21.

Htet Htet Htun, Michael Biehl, and Nicolai Petkov. Survey of feature selection and extraction techniques for stock market prediction. *Financial Innovation*, 9(1):26, Jan 2023. ISSN 2199-4730. doi: 10.1186/s40854-022-00441-7. URL https://doi.org/10.1186/s40854-022-00441-7.

Boming Huang, Yuxiang Huan, Lida Xu, Lirong Zheng, and Zhuo Zou. Automated trading systems statistical and machine learning methods and hardware implementation: a survey. *Enterprise Information Systems*, 13:132 – 144, 2018. URL https://api.semanticscholar.org/CorpusID:57663918.

Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291, 1979. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1914185.

Michael Kampouridis and Fernando E.B. Otero. Evolving trading strategies using directional changes. *Expert Systems with Applications*, 73:145–160, 2017. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2016.12.032. URL https://www.sciencedirect.com/science/article/pii/S095741741630714X.

M. G. Kendall and A. Bradford Hill. The analysis of economic time-series-part i: Prices. *Journal of the Royal Statistical Society. Series A (General)*, 116(1):11–34, 1953. ISSN 00359238. URL http://www.jstor.org/stable/2980947.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Deepak Kumar, Kartik Sahoo, and Manoj Thakur. A hybrid multicategory framework for generating automated trading systems. *Concurrency and Computation: Practice and Experience*, 35(22), April 2023. ISSN 1532-0634. doi: 10.1002/cpe.7746. URL http://dx.doi.org/10.1002/cpe.7746.

Karen Kunz and Jena Martin. Into the breech: The increasing gap between algorithmic trading and securities regulation. *Journal of Financial Services Research*, 47(1):135–152, Feb 2015. ISSN 1573-0735. doi: 10.1007/s10693-013-0184-0. URL https://doi.org/10.1007/s10693-013-0184-0.

Petko Kusev, Harry Purser, Renata Heilman, Alex J. Cooke, Paul Van Schaik, Victoria Baranova, Rose Martin, and Peter Ayton. Understanding risky behavior: The influence of cognitive, emotional and hormonal factors on decision-making under risk. *Frontiers in Psychology*, 8, 2017. ISSN 1664-1078. doi: 10.3389/fpsyg.2017.00102. URL https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2017.00102.

Ki-Yeol Kwon and Richard J. Kish. Technical trading strategies and return predictability: Nyse. *Applied Financial Economics*, 12(9):639–653, September 2002. ISSN 1466-4305. doi: 10.1080/09603100010016139. URL http://dx.doi.org/10.1080/09603100010016139.

Chetankumar J. Lad and Hiral R. Tailor. An empirical study on emotional bias affecting investment decisions of investors. *Global Journal of Research in Management*, 6(1), 2016. ISSN 2319-8915. URL https://www.i-scholar.in/index.php/gjrm/article/view/153250.

Baruch Lev and S. Ramu Thiagarajan. Fundamental information analysis. *Journal of Accounting Research*, 31(2):190, 1993. ISSN 0021-8456. doi: 10.2307/2491270. URL http://dx.doi.org/10.2307/2491270.

Hailin Li, C.H. Dagli, and David Enke. Short-term stock market timing prediction under reinforcement learning schemes. pages 233 – 240, 05 2007. ISBN 1-4244-0706-0. doi: 10.1109/ADPRL.2007.368193.

Bryan Lim, Stefan Zohren, and Stephen J. Roberts. Enhancing time-series momentum strategies using deep neural networks. In *The Journal of Financial Data Science*, 2019. URL https://api.semanticscholar.org/CorpusID:133608683.

Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655, 2021. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2021.03.088. URL https://www.sciencedirect.com/science/article/pii/S2212827121003796. 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020.

J. Bradford De Long, Andrei Shleifer, Lawrence H. Summers, and Robert J. Waldmann. Noise trader risk in financial markets. *Journal of Political Economy*, 98(4):703–738, 1990. ISSN 00223808, 1537534X. URL http://www.jstor.org/stable/2937765.

John M. Maheu and Thomas H. McCurdy. Components of Market Risk and Return. *Journal of Financial Econometrics*, 5(4):560–590, 08 2007. ISSN 1479-8409. doi: 10.1093/jjfinec/nbm012. URL https://doi.org/10.1093/jjfinec/nbm012.

Burton G Malkiel. Returns from investing in equity mutual funds 1971 to 1991. *Journal of Finance*, 50(2):549–72, 1995. URL https://EconPapers.repec.org/RePEc:bla:jfinan:v:50:y:1995:i:2:p:549-72.

Burton Gordon Malkiel. *A Random Walk Down Wall Street: The Time-Tested Strategy for Successful Investing*. W.W. Norton, 2003.

Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952. ISSN 00221082, 15406261. URL http://www.jstor.org/stable/2975974.

Charilaos Mertzanis. *Complexity Analysis and Systemic Risk in Finance: Some Methodological Issues*, pages 199–237. Springer International Publishing, Cham, 2014. ISBN 978-3-319-09683-4. doi: 10.1007/978-3-319-09683-4_11. URL https://doi.org/10.1007/978-3-319-09683-4_11.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

John Moody and Matthew Saffell. Reinforcement learning for trading. 11, 1998. URL https://proceedings.neurips.cc/paper_files/paper/1998/file/4e6cd95227cb0c280e99a195be5f6615-Paper.pdf.

Alan Moreira and Tyler Muir. When is the price of risk high? *SSRN Electronic Journal*, 2015. ISSN 1556-5068. doi: 10.2139/ssrn.2659431. URL http://dx.doi.org/10.2139/ssrn.2659431.

Tobias J. Moskowitz, Yao Hua Ooi, and Lasse Heje Pedersen. Time series momentum. *Journal of Financial Economics*, 104(2):228–250, 2012. ISSN 0304-405X. doi: https://doi.org/10.1016/j.jfineco.2011.11.003. URL https://www.sciencedirect.com/science/article/pii/S0304405X11002613. Special Issue on Investor Sentiment.

George Mylnikov. Volatility targeting: It's complicated! *Journal of Portfolio Management*, 47(8):57–74, Aug 2021 2021. URL https://tilburguniversity.idm.oclc.org/login?url=https://www.proquest.com/scholarly-journals/volatility-targeting-s-complicated/docview/2557199871/se-2.

Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, 53(4):3007–3057, Apr 2020. ISSN 1573-7462. doi: 10.1007/s10462-019-09754-z. URL https://doi.org/10.1007/s10462-019-09754-z.

Hassan Oukhouya and Khalid El Himdi. Comparing machine learning methods—svr, xgboost, lstm, and mlp— for forecasting the moroccan stock market. *Computer Sciences amp; Mathematics Forum*, 7(1), 2023. ISSN 2813-0324. doi: 10.3390/IOCMA2023-14409. URL https://www.mdpi.com/2813-0324/7/1/39.

Cheol-Ho Park and Scott H. Irwin. What do we know about the profitability of technical analysis? *Journal of Economic Surveys*, 21(4):786–826, 2007. doi: https://doi.org/10.1111/j.1467-6419.2007.00519.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-6419.2007.00519.x.

James M. Patell and Mark A. Wolfson. The intraday speed of adjustment of stock prices to earnings and dividend announcements. *Journal of Financial Economics*, 13(2):223–252, 1984. ISSN 0304-405X. doi: https://doi.org/10.1016/0304-405X(84)90024-2. URL https://www.sciencedirect.com/science/article/pii/0304405X84900242.

Ser-Huang Poon. Utility theory. In *Advanced Finance Theories*, chapter 1, pages 1–4. World Scientific Publishing Co. Pte. Ltd., 2018. URL https://EconPapers.repec.org/RePEc:wsi:wschap:9789814460385_0001.

John W. Pratt. Risk aversion in the small and in the large. *Econometrica*, 32(1/2):122–136, 1964. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913738.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

Peter Scholz. Size matters! how position sizing determines risk and return of technical timing strategies. CPQF Working Paper Series 31, Frankfurt a. M., 2012. URL https://hdl.handle.net/10419/55526.

Stephan Schulmeister. The interaction between technical currency trading and exchange rate fluctuations. *SSRN Electronic Journal*, 2005. ISSN 1556-5068. doi: 10.2139/ssrn.884330. URL http://dx.doi.org/10.2139/ssrn.884330.

Ashish Kumar Shakya, Gopinatha Pillai, and Sohom Chakrabarty. Reinforcement learning algorithms: A brief survey. *Expert Systems with Applications*, 231:120495, 2023. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2023.120495. URL https://www.sciencedirect.com/science/article/pii/S0957417423009971.

Robert J. Shiller. From efficient markets theory to behavioral finance. *Journal of Economic Perspectives*, 17(1):83–104, March 2003. doi: 10.1257/089533003321164967. URL https://www.aeaweb.org/articles?id=10.1257/089533003321164967.

Sweta Singh, Rahul Bhagat, S. H. Preeti, and G. P. Girish. Transforming the financial industry through machine and deep learning innovations. In Pandian Vasant, Mohammad Shamsul Arefin, Vladimir Panchenko, J. Joshua Thomas, Elias Munapo, Gerhard-Wilhelm Weber, and Roman Rodriguez-Aguilar, editors, *Intelligent Computing and Optimization*, pages 167–176, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-36246-0.

Jasper Snoek, H. Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems*, 2012. URL https://api.semanticscholar.org/CorpusID:632197.

Bruno H. Solnik. Note on the validity of the random walk for european stock prices. *The Journal of Finance*, 28(5):1151–1159, 1973. ISSN 00221082, 15406261. URL http://www.jstor.org/stable/2978754.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Thibaut Théate and Damien Ernst. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632, July 2021. ISSN 0957-4174. doi: 10.1016/j.eswa.2021.114632. URL http://dx.doi.org/10.1016/j.eswa.2021.114632.

Minh Tran, Duc Pham-Hi, and Marc Bui. Optimizing automated trading systems with deep reinforcement learning. *Algorithms*, 16(1), 2023. ISSN 1999-4893. doi: 10.3390/a16010023. URL https://www.mdpi.com/1999-4893/16/1/23.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:13756489.

Yongfeng Wang and Guofeng Yan. Survey on the application of deep learning in algorithmic trading. *Data Science in Finance and Economics*, 1(4):345–361, 2021. ISSN 2769-2140. doi: 10.3934/DSFE.2021019. URL https://www.aimspress.com/article/doi/10.3934/DSFE.2021019.

J. Welles Wilder. *New Concepts in Technical Trading Systems*. Trend Research, 1978.

Zihao Zhang, Stefan Zohren, and Stephen J. Roberts. Deep reinforcement learning for trading. In *The Journal of Financial Data Science*, 2019. URL https://api.semanticscholar.org/CorpusID:208248040.