MACHINE LEARNING TOOLS IN FINANCIAL MATHEMATICS: Direct Reinforcement Learning for High-Frequecy Foreign Exchange Trading

by

Student 22077004

October 2023

Dissertation Supervisor: Prof. Codina Cotar

Dissertation submitted in part fulfilment of the

Degree of Master of Science in Data Science

Department of Statistical Science

University College London

ABSTRACT

High-frequency trading has evolved significantly, becoming a dominant force in financial markets. This research explores the application of Machine Learning (ML), particularly Reinforcement Learning (RL), in the context of intraday foreign exchange trading. Focusing on the EUR/GBP exchange rate, the objective of the research is to develop a profitable trading algorithm that adapts to dynamic market conditions. This study introduces an automated financial trading model utilizing RL, offering a novel contribution by considering both true and detrended intraday variabilities of the EUR/GBP exchange rate. The research reveals that incorporating a shallow neural network for feature learning improves risk-adjusted trading performance, albeit with higher volatility. The inclusion of a detrended conditional volatility forecast stabilizes the trading performance. Additionally, the inclusion of a true conditional volatility forecast enhances trading performance, although the significance of this improvement remains uncertain. This research highlights the potential of RL in intraday trading and underscores the importance of feature engineering in enhancing performance of RL trading models. The research findings align with previous literature, opening avenues for further exploration in the realm of high-frequency foreign exchange trading.

Keywords: high-frequency trading, direct reinforcement learning, conditional volatility modelling, trading performance analysis

Length of the project (total word count): 14,800

CONTENTS

><-1000

LIST OF ABBREVIATIONS

ACF Autocorrelation Function 63, 102

ARCH-LM Autoregressive Conditional Heteroskedasticity Lagrange Multiplier 98, 99

AT Algorithmic Trading 18, 19

BPTT Backpropagation Through Time 73

DL Deep Learning 44

DNN Deep Neural Network 26, 44

DP Dynamic Programming 25

DRL Direct Reinforcement Learning 26, 27, 28, 42, 44, 45, 54, 69, 70, 71, 72, 90, 91

DSR Differential Sharpe ratio 37, 41, 42, 43, 44, 71, 73, 74, 78, 91

EMA SR Exponential Moving Average Sharpe ratio 38, 40, 41, 42

FX Foreign Exchange 46, 54, 55, 90, 91, 104

GA Genetic Algorithm 44

GARCH Generalized Autoregressive Conditional Heteroskedasticity 8, 13, 15, 16, 49, 50, 51, 52, 53, 66, 67, 68, 110, 111, 112, 113, 114

HFT High-Frequency Trading 17, 19, 33, 34, 35, 85, 90, 91

MDP Markov Decision Process 20, 21, 23, 24, 25, 26

ML Machine Learning 17, 19

MM Market Making 18

OE Order Execution 18

OHLC Open-High-Low-Close 54, 55, 56

PACF Partial Autocorrelation Function 63

PM Portfolio Management 18

QT Quantitative Trading 18

ReLU Rectified Linear Unit 70, 75, 77

RL Reinforcement Learning 17, 18, 19, 20, 21, 23, 25, 26, 27, 28

SR Sharpe ratio 35, 36, 37, 38, 41, 42, 44, 78, 93, 94

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

High-Frequency Trading (HFT) has consistently grown in popularity over the past several decades, with its emergence in the early 2000s being closely tied to advancements in related high-speed technologies, enabling faster execution of trades and providing nuanced strategies that exploit minute fluctuations in financial markets. In 2010, more than half of the entire US equity turnover in terms of volume was attributed to HFT [1]. HFT is executed exclusively by computers, which leverage smart algorithms provided by quantitative financial specialists.

Machine Learning (ML) methods have been extensively utilized for intraday trading. Reinforcement Learning (RL), a widely studied and explored ML method, has emerged as one of the preferred approaches due to its capability to directly learn optimal trading strategies by interacting with the market environment and adapting to its dynamic conditions.

Navigating through the intricacies of the HFT landscape, the present research introduces an automated financial trading model designed using the RL framework. The subject of the present research project is the real market EUR/GBP high-frequency exchange rate, while the objective is to develop a profitable trading algorithm for intraday foreign exchange trading. Moreover, the present research takes into consideration the variability in the EUR/GBP exchange rate and explores whether it can enhance the performance of the RL-based automated financial trading model.

The intraday variability of the EUR/GBP exchange rate tend to exhibit distinct and repetitive patterns. Therefore, both the true and detrended intraday variabilities of the EUR/GBP exchange rate are considered, and their potential to enhance trading performance is compared. This aspect represents the novel contribution of the present research to existing studies in the field.

The structure of the study is outlined as follows. Chapter 2 encompasses the literature review. The concept of RL is rigorously unpacked and RL method is thoroughly compared to ML supervised learning approaches. Furthermore, within Chapter 2, I discuss a popular RL trading system design and engage in a debate regarding various trading performance criteria (the RL trading system objectives). Chapter 2 concludes with a comprehensive technical overview of the high-frequency financial data analysis. Chapter 3 delves into the empirical findings in detail. Firstly, it involves modeling the price returns of the EUR/GBP exchange rate through time series analysis. Additionally, it entails the estimation and forecasting of the volatility of the EUR/GBP exchange rate price returns. Following that, I provide a thorough discussion of designed RL trading models. I compare and analyze the trading performance of these RL models using various input configurations. Chapters 4 and 5 wrap up the research by addressing potential pitfalls and identifying avenues for future research.

2. LITERATURE REVIEW

2.1 Quantitative Trading

Quantitative Trading (QT) is the process of trading securities solely based on computer algorithms' buy/sell decisions [10]. QT expands on technical analysis through assimilating fundamental data and news events, and blending those inputs into a complex quantitative system [10]. The main objective of QT is set to be long-term profit maximization given an agent's risk appetite [45]. Presently, quantitative hedge funds are at the forefront of securities investment.

QT tasks encompass Algorithmic Trading (AT), Portfolio Management (PM), Order Execution (OE), and Market Making (MM) [45]. AT and PM generalize to macro-level quantitative trading tasks [45].

PM is a fundamental task in QT, involving the allocation and periodic reallocation of financial assets by investors to maximize long-term profitability [45].

The main objectives of OE are twofold: it not only takes on the responsibility of fulfilling the entire order but also aims to achieve a more cost-effective execution that maximizes profit or minimizes costs [45].

In numerous markets, there are designated dealers known as market makers or specialists who bear specific responsibilities to ensure the smooth operation of the trading mechanism [4]. Market makers are considered as dealers who provide bid and ask prices at which they commit to buying and selling the asset, a practice known as MM [4].

AT refers to the continuous practice of traders buying and selling a specific financial asset with the intention of generating a profit [45]. Conventional AT methods identify trading signals by utilizing technical indicators or mathematical models [45]. HFT is a subset of AT, which typically involves professional traders leveraging rapid data retrieval and processing to implement short-term trading tactics [50]. High-frequency traders usually engage in multiple trades within a single day and tend not to hold positions overnight [50]. The majority of academic research indicates that AT and HFT enhance market quality and lower transaction costs [50]. They typically boost liquidity, price efficiency, and decrease short-term volatility [50]. However, in times of high volatility or market stress, they might intensify price fluctuations [50].

The current research primarily centers on AT, with a specific emphasis on HFT.

2.2 Reinforcement Learning

RL is a branch of ML that is concerned with how can an intelligent agent acquire the ability to make optimal sequences of decisions under uncertainty [9, 39]. The credit assignment problem for learning systems was established by Marvin Minsky in 1961. It was formulated as identification of the causal relationship between past decisions and future outcomes, which not only distinguishes RL from other ML paradigms but also constitutes one of its most significant challenges [9, 32].

In the RL framework, an intelligent agent repeatedly interacts with the world, gathering information about the quality of its decisions through trial-and-error [9, 46]. The optimality or quality of decisions is typically measured using utility or other well-defined objectives, such as profitability, which a rational agent aims to maximize [9, 39]. Because of the inherent stochasticity of the world, a rational agent seeks to maximize the objective on average or in expectation [9].

RL is applicable to a huge number of domains including video games, robotics, educational games, healthcare, and natural language processing [9, 39].

Markov Decision Processes

Markov Decision Process (MDP) is a mathematical framework, which represents the class of problems that RL aims to tackle [39]. MDP is an interactive and dynamic closed-loop process depicting sequential decision problem with uncertainty [39].

As discussed in [9, 39], at each time step $t = 0, 1, 2, 3, \dots$ in response to

- an observed *State* S_t ($S_t \in \mathcal{S}$, where $\mathcal{S}$ is a countable *State Space*) and
- a received numerical reward R_t ($R_t \in \mathcal{D}$, where $\mathcal{D}$ is a countable subset of real numbers $\mathbb{R}$)

an *Agent* (a designed algorithm) is taking a new action A_t ($A_t \in \mathcal{A}$, where $\mathcal{A}$ is a countable *Action Space*). The action A_t impacts the *Environment* producing

- an observation S_{t+1} and
- a reward $R_{t+1} = \mathcal{R}(s, a, s')$, where s' is a successor state, a is action taken from starting state s, and $\mathcal{R}$ is a reward function.

This cycle continues until the MDP comes to an end. The MDP is said to terminate at time step T if S_T is a state within the collection denoted as $\mathcal{T}$, where $\mathcal{T}$ is the set of *Terminal States* [39].

The state S_t could encompass any data type, for instance, daily closing price of a single stock [39].

Mapping noisy real world into a mathematical objective implies modelling stochastic processes [39]. Transition probability for time t captures uncertainty and stochasticity in the decision problem and is represented as follows:

$$\mathcal{P}_R(s, a, r, s') = \mathbb{P}[(R_{t+1} = r, S_{t+1} = s')|(S_t = s, A_t = a)]\ \forall t \geq 0, \forall s, s' \in \mathcal{S}, \forall a \in \mathcal{A}, \forall r \in \mathcal{D},$$

such that

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{D}} \mathbb{P}[(R_{t+1} = r, S_{t+1} = s')|(S_t = s, A_t = a)] = 1\ \forall s, \in \mathcal{S}, \forall a \in \mathcal{A}, \tag{2.1}$$

where s' is a successor state and r is an expected reward from action a taken from starting state s [39]. In the realm of RL the transition probabilities are unknown [39].

In the context of MDP, it is commonly assumed that transition probabilities defined in Equation 2.1 adhere to two fundamental properties [39]:

- the **Markov property** asserts that transitions are influenced solely by the previous state and action, as represented by the following quation: $\mathbb{P}[(R_{t+1}, S_{t+1})|(S_t, A_t)] = \mathbb{P}[(R_{t+1}, S_{t+1})|(S_t, A_t, S_{t-1}, A_{t-1}, \ldots, S_0, A_0)]$ for time step t $\forall t \geq 0$,
- **time-homogeneity** implies that $\mathbb{P}[(R_{t+1}, S_{t+1})|(S_t, A_t)]$ remains constant regardless of the specific time instance t $\forall t \geq 0$.

Total accumulated reward denoted as G_t at time period t is computed in the following manner:

$$G_t = \sum_{i=1}^{T} \gamma^{i-1} R_{t+i} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_{t+T} \ \forall t \geq 0, \tag{2.2}$$

where $\gamma \in [0, 1]$ is a discount factor [39].

The value of a discount factor γ determines the degree of "myopic" or "far-sighted" behavior exhibited by the agent [39]:

- a **myopic** agent places greater importance on immediate rewards compared to distant ones (when γ is closer to 0),
- a **far-sighted** agent treats current and distant rewards as equally significant (when γ is closer to 1).

Several factors contribute to the discounting of rewards R_t [39].

1. **Mathematical Convenience:** this prevents the summation defined in Equation 2.2 from diverging to infinity [39].
2. **Computational Tractability:** discounting enhances the manageability of algorithms, making them more feasible [39].

3. **Modeling Reasons:** when rewards are modeled as financial quantities, incorporating *time-value-of-money* aligns with fundamental concepts in Economics and Finance. This principle underscores the notion that there is more value in receiving a dollar now rather than in the future [39].

4. **Addressing Uncertainty:** setting the discount factor $0 < \gamma < 1$ is technically motivated by the fact that the real world models often fall short in fully capturing future uncertainty. Discounting with the factor γ helps mitigate potential inaccuracies in future rewards due to limitations in modeling future uncertainty [39].

5. **Human-like Behavior:** from an AI perspective, if we aim to create machines that emulate human behavior, psychologists have demonstrated that both humans and animals tend to favor immediate rewards over those in the future [39].

The agent's decision-making process depends on numerical feedback received from the environment. Formally, at each time step t, an action A_t is chosen based on the observation of the state S_t. The general way that a sequence of feedback is transformed into the agent's decisions is termed as the *Policy* function denoted as π. This function is formally defined as follows:

$$\begin{aligned}\pi(s,a) = \mathbb{P}[A_t = a|S_t = s] \ \forall t \geq 0, \forall s \in \mathcal{S}, \forall a \in \mathcal{A},\\ \text{such that} \sum_{a\in\mathcal{A}} \pi(s,a) = 1 \ \forall s \in \mathcal{S},\end{aligned} \tag{2.3}$$

as described in [39]. The policy π is stationary if $\mathbb{P}[A_t = a|S_t = s]$ remains unaffected by changes in the variable t [39].

Markov Decision Process Prediction Problem

Prediction refers to figuring out what rewards the agent can expect in the future when it follows a certain policy π [39].

The *Value function* V^π for a given policy π is defined as

$$V^\pi(s) = \mathbb{E}_{\pi,\mathcal{P}_R}[G_t|S_t = s] \ \forall t \geq 0, \forall s \in \mathcal{S}, \tag{2.4}$$

where the subscript $\mathcal{P}_R$ represents transition probability defined in Equation 2.1 and G_t is a total accumulated reward defined in Equation 2.2 [39]. The value function represents expected accumulated total accumulated reward if the agent consistently follows a given policy π while starting from a given state s [39].

The value function defined in Equation 2.4 can be expressed recursively as follows:

$$V^{\pi}(s) = \sum_{a\in\mathcal{A}} \pi(s,a)\cdot\left\{\mathbb{E}[R_{t+1}|(S_t = s, A_t = a)] + \gamma\cdot\sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{D}}\mathcal{P}_R(s,a,r,s')\cdot V^{\pi}(s')\right\}\ \forall s\in\mathcal{S}, \tag{2.5}$$

where $\mathbb{E}[R_{t+1}|(S_t = s, A_t = a)]$ is an immediate expected reward, s is a starting state at time step t, s' is a successor state at time step $t+1$, and $\pi(s,a)$ is a deterministic policy that an agent follows from a given starting state s [39].

Markov Decision Process Control Problem

The agent's ultimate goal can be described then as identifying the optimal policy denoted as π^* that maximizes the value function:

$$\pi^*(s) = \operatorname*{argmax}_{\pi\in\Pi} V^{\pi}(s)\ \forall s\in\mathcal{S}, \tag{2.6}$$

where Π is a set of stationary policies across the domains of $\mathcal{S}$ and $\mathcal{A}$ [39]. This is also known as MDP *Control* problem [39].

The resulting value function is called the *Optimal Value Function*: $V^*(s) = \max_{\pi\in\Pi} V^{\pi}(s)\ \forall s\in\mathcal{S}$ [39].

Equations 2.5 and 2.6 together allow to identify MDP State-Value Function Bellman Optimality Equation $\forall s\in\mathcal{S}$ [39]:

$$V^*(s) = \max_{a\in\mathcal{A}}\left(\sum_{a\in\mathcal{A}} \pi(s,a)\cdot\left\{\mathbb{E}[R_{t+1}|(S_t = s, A_t = a)] + \gamma\cdot\sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{D}}\mathcal{P}_R(s,a,r,s')\cdot V^{\pi}(s')\right\}\right). \tag{2.7}$$

RL algorithms seek to compute the optimal value function, denoted as $V^*(s)$ and defined in Equation 2.7, as well as the corresponding policy $\pi^*(s)$ defined in Equation 2.6 that achieves this optimal value function. This process effectively tackles the control problem within the framework of MDP, as detailed in the reference [39].

Trivial example of a Markov Decision Process

Let me introduce a trivial example of MDP. Imagine a financial market scenario where there is an agent who trades a single Stock j.

The action space $\mathcal{A}$ would include actions such as $\{$buy, sell, hold$\}$, where each action corresponds to a different trading decision for Stock j. The State space $\mathcal{S}$ would represent the relevant information for trading Stock j. It could include the current price of Stock j, some technical indicators, and the agent's current holding of stock j at each time step t. The reward value R_t would reflect the agent's financial gain or loss after taking action A_t at time step t. It could be defined as the change in the agent's portfolio value due to the action A_t defined as $R_t = P_t - P_{t-1}$, where P_t is the agent's portfolio value at time step t.

The agent's goal is to maximize its cumulative profit over a predefined trading horizon T by making appropriate trading decisions for Stock j.

Let us assume that the agent is far-sighted and values both immediate and future financial gains equally. Then, the expected total cumulative reward for the agent is $G_t = \sum_{t=1}^{T} R_t$.

The primary objective of the agent is to make the best trading decisions for Stock j to maximize its cumulative profit by optimizing its trading strategy.

This type of problem will be examined in the current research.

Solving Reinforcement Learning Problems

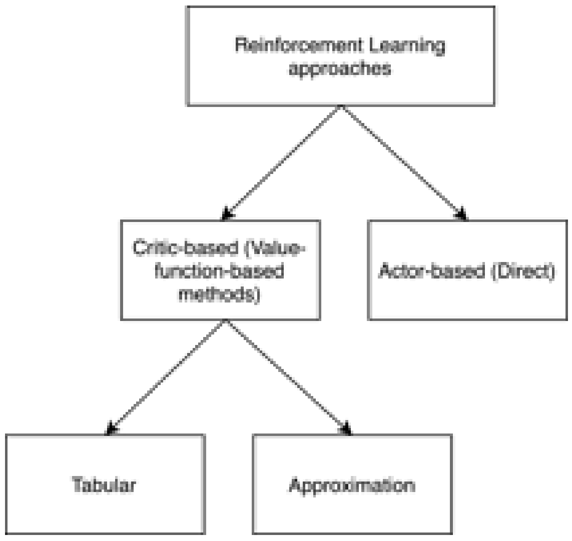

Reinforcement Learning approaches
Critic-based (Value-function-based methods)
Actor-based (Direct)
Tabular
Approximation

Figure 2.1: Reinforcement Learning approaches.

In the domain of RL, two fundamental learning approaches exist (refer to Figure 2.1): *critic-based*, which entails learning value functions (referred to as "value-function-based methods" henceforth), and *actor-based*, which focuses on learning actions [14]. Value-function-based methods are frequently employed and directly estimate value functions [14].

The two mainstream value-function-based methods encompass tabular and approximation methods (refer to Figure 2.1) [39]. These methods are particularly suitable for situations in which immediate performance feedback is not readily available at every time step [34].

Tabular methods resemble an exhaustive search through all the states to update the value function [39]. One of the most popular tabular methods is Dynamic Programming (DP) [34]. DP methods do not require interaction with the environment or the need to learn from the data stream of states and rewards provided by the environment [39]. Additionally, DP methods assume that the transition probabilities are known [39].

Approximation methods aim to estimate the value function for the MDP control problem when computational resources are limited [39, 45].

Large state and/or action spaces can pose significant challenges in RL problems for several reasons. Firstly, attempting to store a tabular representation of the MDP or the value function is likely to exceed the available storage capacity [39]. Secondly, the systematic examination of all states and their transition probabilities becomes impractical within the constraints of time [39]. As a result, computational feasibility when employing brute-force methods is primarily attainable in scenarios characterized by small state and action spaces [45].

In numerous real-world applications, dealing with expansive state and/or action spaces is commonplace, making it more practical to employ a RL approach based on approximating the value function, as suggested in the reference [39]. For example, one viable method for approximating the value function involves the use of Deep Neural Network (DNN), as highlighted in references [33, 39].

However, for trading challenges, relying on value-function-based methods is ill-suited [13, 14, 34]. In particular, the trading environment's complexity exceeds the feasibility of discrete space approximation [14]. Furthermore, value-function-based methods are not well-suited for the challenges of dynamic online trading [14].

On the other hand, actor-based RL algorithms do not require obtaining a value function to formulate a policy, as noted in the reference [34]. These algorithms are commonly referred to as Direct Reinforcement Learning (DRL) algorithms, initially introduced in the referenced work [34]. Unlike the value function methods discussed previously, DRL algorithms depend on estimating incremental performance to optimize strategies, as emphasized in the reference [34]. DRL is known for its relative simplicity, as it solely requires a differentiable objective function with latent parameters [14]. DRL provides two notable advantages [14]:

- flexibility in optimization objectives,
- the ability to capture continuous representations of market conditions.

Furthermore, in various financial decision-making contexts, results often unfold gradually over time, enabling the prompt evaluation of short-term performance [34]. This attribute of financial decision-making problems makes them suitable for the implementation of DRL [34].

Direct Reinforcement Learning versus Traditional Supervised Approaches for Trading Challenges

Within the domain of RL, the absence of explicit target outputs is a notable feature [45]. In this paradigm, the system takes actions, receives performance feedback, and subsequently adapts its internal parameters to enhance future rewards [45]. This method directly focuses on optimizing the ultimate measure of trading performance [45].

The effectiveness of DRL in training trading systems, particularly when factoring in transaction costs, has been demonstrated when compared to conventional supervised approaches [36]. DRL has been pitted against traditional supervised techniques such as trading based on forecasts and training trading systems on labeled data [36].

Trading based on forecasts primarily seeks to minimize forecast errors on a training dataset, but this objective may not align with the trading system's ultimate goal, potentially resulting in suboptimal performance [36]. Conversely, DRL methods sidestep the challenging task of predicting future prices and instead optimize the primary objective directly [45].

When training trading systems using labeled data, the actual performance achievable by the trading module often falls significantly short of what is indicated by labeled trades [45]. Moreover, the labeling process itself poses various challenges [45].

DRL and RL approaches, in general, eliminate the need for intermediate steps such as making forecasts or labeling desired trades, thereby establishing their superiority over conventional supervised approaches [45].

Finally, the DRL algorithm handles the high noise in financial data well and can quickly adjust to changing market conditions [34].

Given the benefits of DRL, the present study aims to employ a DRL algorithm to address a problem akin to the scenario outlined in Subsection 2.2.4.

2.3 Trading System and Trading Performance Criteria

Trading System

The DRL trading system framework is adapted from prior studies [10, 34, 35, 36, 45].

It is assumed that a consistent position size is maintained when trading a single risky security [34]. The trading system characterizes a trader with discrete actions.

Risky asset j price sequences released from the exchange center are denoted as $z_1^j, z_2^j, ..., z_t^j, \ldots, z_T^j$ [10, 34]. Risk-free asset price sequences are denoted as $z_1^f, z_2^f, ..., z_t^f, \ldots, z_T^f$. Note that the superscript f means that the value is related to a risk-free asset.

The trader is assumed to only take long, neutral or short positions, denoted as $F_t \in \{-1, 0, 1\}$, with a constant magnitude [34]. Within the context of RL, this is referred to as an action space. An alternative approach involves setting $F_t \in \{-1, 1\}$, which means the trader is always in the market, as neutral positions are not allowed [20]. A long position involves buying a certain quantity of a security, while a short position is established by selling a security. The two main types of short selling are *naked (uncovered)* and *covered*. The crucial difference stems from whether the asset sold by a short-seller was borrowed (secured). Naked short selling, which involves selling an asset which has not been secured, carries significant risks and is often subject to stricter regulations. Naked short selling is prohibited in the EU and in the UK [18, 48].

Short selling is differentiated further based on whether the asset sold was owned by an investor [4]. If an investor already owns the asset, selling it is a straightforward process [4]. However, if the investor does not own the asset, it must be borrowed from another party just before the sale is executed [4]. Short selling is a practice employed particularly in declining markets [4].

The real-time trading decision, which corresponds to an action in the context of RL, is determined based on the prevailing market conditions [10]. The position F_t is established at the end of time period t and is reassessed at the end of the subsequent time period at $t+1$ [34]. The trade is possible at the end of the each time period [34]. At the conclusion of the time interval $(t-1, t]$, the trading system's profit or loss generated by the position F_{t-1} maintained during that interval is determined [34]. The trading system's profit or loss also accounts for any transaction costs incurred at time t due to variations in the positions F_{t-1} and F_t [34].

From this point onward, and throughout the remainder of the current research, when referring to the learned system parameters at time t, a subscript t for parameter scalar or vector will be used. The concept is also known as "on-line learning" and for a detailed discussion about it refer to the Subsection 2.3.6. Additionally, it is crucial to highlight that the bold letters will be used to denote vector notation.

The decision function for a single-asset trading system is as follows:

$$\begin{aligned} F_t &= F(\boldsymbol{\theta}_\mathbf{t}; F_{t-1}, I_t), \\ I_t &= \{z_t, z_{t-1}, z_{t-2}, \dots, z_{t-m}, \mathbf{y_t}, \mathbf{y_{t-1}}, \mathbf{y_{t-2}}, \dots\}, \end{aligned} \tag{2.8}$$

where $\boldsymbol{\theta}_\mathbf{t}$ is a vector representing the learned system parameters at time t and I_t denotes the information set at time t, which includes current and m historical values of the price series z_p, $p \in \{t, t-1, \dots, t-m\}$ and an arbitrary number of additional external variables $\mathbf{y_q}$, $q \in \{t, t-1, \dots\}$ [34]. For instance, these external variables may represent certain technical indicators.

In a simplified scenario, the decision function $F(\cdot)$ can be represented as a linear combination of the following inputs: the position at the previous time period F_{t-1}, the current price return, and m past price returns denoted as $r_t, r_{t-1}, r_{t-2}, \dots, r_{t-m}$. The value of a decision function is mapped to the values $\{-1, 0, 1\}$, representing short, neutral, and long positions, respectively. The specific definition of the past price returns will be provided later in Subsection 2.3.3.

Then, in a simplified scenario, the decision function $F(\cdot)$ will assume the following form:

$$F_t = \operatorname{sgn}\left(\sum_{i=0}^{m} v_{i,t} r_{t-i} + u_{m+1,t} F_{t-1} + w_t\right), \tag{2.9}$$

where sgn stands for a *sign* function that outputs a sign of a real number, m is an integer value that signifies the length of the return lags, $v_{i,t}$ $\forall i$, $u_{m+1,t}$, and w_t represent scalar weights and a bias term at time period t.

The term $u_{m+1,t}F_{t-1}$ from Equation 2.9 aims to discourage the agent from frequently changing trading positions, thereby minimizing heavy transaction costs [14]. In the past, numerous trading systems have encountered a problem of excessive switching behavior, leading to significantly high transaction costs [13].

Note that Equation 2.9 represents a linear model; however, it includes a nonlinear mapping to discrete values through the *sign* function.

One straightforward way to generalize the trading system presented in Equation 2.9 is to utilize a continuously valued decision function $F(\cdot)$, for instance, by using a different mapping function, such as *tanh*, and only then impose discrete values $\{-1, 0, 1\}$, corresponding to short, neutral and long positions [34].

Allow me to provide further details regarding this generalization. Using a slightly different notation from [52], $\operatorname{sgn}(F_t) \in \{-1, 0, 1\}$ will refer to a trading position at time period t and $F_t \in [-1, 1]$ will refer to a trading signal at time period t, where

$$F_t = \tanh\left(\sum_{i=0}^{m} v_{i,t} r_{t-i} + u_{m+1,t}F_{t-1} + w_t\right) \tag{2.10}$$

with the same notations used as in Equation 2.9.

Let me provide a more detailed explanation of the visual representation of Equation 2.10 as depicted in Figure 2.2. Let us assume, for the sake of simplicity, that a trained trading system input comprises a single sample. At time period t, the system's inputs consist of price returns r_z, where $z \in \{t, t-1, \ldots, t-m\}$, and the previous trading signal F_{t-1}. This collection of inputs at time t is encapsulated within the input vector $\langle F_{t-1}, r_{t-m}, \ldots, r_t \rangle$ denoted as $\mathbf{x_t}$, which has a dimensionality of $m+2$.

Similar to the trading system defined by Equation 2.8, the model encompasses a vector $\boldsymbol{\theta}_\mathbf{t}$ and a scalar w_t as the learned system parameters at time t (refer to Figure 2.2). Subsequently, the dot product between $\boldsymbol{\theta}_\mathbf{t}$ and $\mathbf{x_t}$ adjusted for a bias w_t is fed into a *tanh* activation function, resulting in the generation of a trading signal F_t. Following this, at time period t, the trading position, which can assume values from the set $\{-1, 0, 1\}$, is determined by applying a *sign* function to the trading signal F_t.

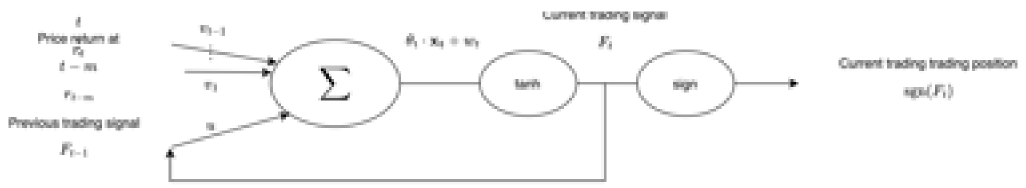


Figure 2.2: Recurrent structure of the trading system. Linear model with a continuously valued decision function and a nonlinear mapping to the discrete values. The diagram is inspired by the scheme presented in [52].

Profits in a trading system are contingent on interconnected decisions, making them reliant on sequences and thus path-dependent [36]. The recurrent nature of the introduced trading system becomes apparent when observing Figure 2.2.

Incorporating a neural network with a single hidden layer enables the construction of a more sophisticated and theoretically powerful trading rule [20].

Within this framework, $F_t \in \{-1, 0, 1\}$ represents the trading position at time period t. The output from the hidden layer (see Figure 2.3) with N neurons is

$$\mathbf{y_t} = \tanh(\Theta_t^{[1]} \cdot \mathbf{x_t} + \beta_t^{[1]}),$$

where $\Theta_t^{[1]}$ is a matrix of size $(N \times (m+2))$, where the weights in the hidden layer are represented row-wise for each node, $\beta_t^{[1]}$ is a scalar that denotes bias, and $\mathbf{x_t}$ is an input vector of size $m+2$ or a matrix of size $((m+2) \times 1)$. Dot product of $\Theta_t^{[1]}$ and $\mathbf{x_t}$ yields $\mathbf{y_t}$, which is a vector of size N or a matrix of size $(N \times 1)$.

The output layer (see Figure 2.3) yields a trading position

$$F_t = \mathrm{sgn}(\Theta_t^{[2]} \cdot \mathbf{y_t} + \beta_t^{[2]}),$$

where $\Theta_t^{[2]}$ is a matrix with dimensions $(1 \times N)$, where the weights in the output layer are arranged in a single row. $\beta_t^{[2]}$ is a scalar that signifies a bias term, and F_t is a scalar output that assumes discrete values from the set $\{-1, 0, 1\}$.

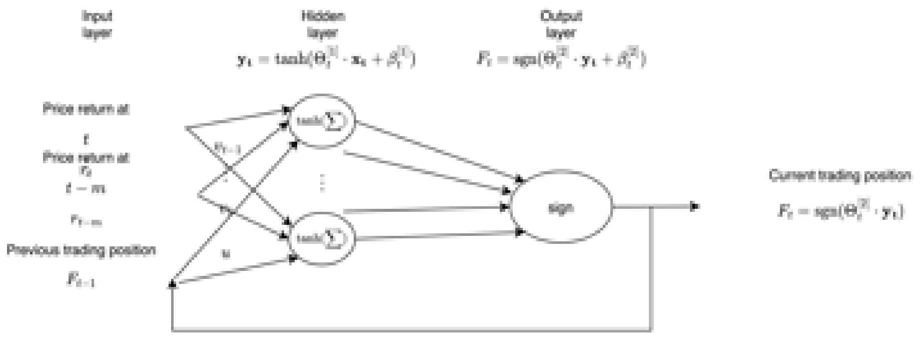

Input layer
Hidden layer
$y_t = \tanh(\Theta_t^{[1]} \cdot x_t + \beta_t^{[1]})$
Output layer
$F_t = \mathrm{sgn}(\Theta_t^{[2]} \cdot y_t + \beta_t^{[2]})$
Price return at t
r_t
Price return at t − m
r_{t-1}
r_{t-m}
Previous trading position
F_{t-1}
u
tanh(∑)
tanh(∑)
sign
Current trading position
$F_t = \mathrm{sgn}(\Theta_t^{[2]} \cdot y_t)$

Figure 2.3: Recurrent structure of the trading system. A neural network with a single hidden layer. The diagram is inspired by the materials presented in [20, 52].

It has been observed that a single-layer neural network outperforms a two-layer neural network in the context of high-frequency foreign exchange trading [20]. This phenomenon can potentially be attributed to the noise inherent in the financial data, which makes more complex models susceptible to overfitting [20].

Financial Market Frictions

The trading system must consider the presence of non-zero transaction costs, which are intended to discourage excessive trading [34]. These transaction costs encompass explicit commission fees, taxes, and additional charges incurred during trade executions [4].

Furthermore, there are institutional constraints governing trades [4]. These restrictions can take the form of either prohibitions on specific types of trades or prerequisites that must be met before trades are authorized [4]. For instance, in the UK, short selling is generally permitted but is subject to specific regulations and restrictions. For example, the Financial Conduct Authority, the financial regulatory body in the UK, imposed restrictions on short selling during periods of market volatility in early 2020, prompted by the spread of Covid-19 [47].

The inclusion of transaction costs within the trading system's operations will be elaborated upon in the upcoming Subsection 2.3.3. Nevertheless, modeling institutional constraints proves to be challenging, as it may vary depending on exceptional events occurring in the financial market and the trader's profile.

Realized Trading Profit

Let me introduce a simple two-asset portfolio consisting of a risk-free asset and a risky asset j. The realized trading profit π_t at time period $t \in \{1, 2, 3, \ldots, T\}$ is defined as follows [36]:

$$\pi_t \coloneqq \beta\{(1 - F_{t-1})r_t^f + F_{t-1}r_t^j - |F_t - F_{t-1}|\delta\}. \tag{2.11}$$

In this context, $\beta > 0$ represents a fixed trading position size, δ denotes the transaction cost rate per asset, and $r_t^j \coloneqq z_t^j - z_{t-1}^j$ and $r_t^f \coloneqq z_t^f - z_{t-1}^f$ stand for the differences in asset prices between time periods t and $t-1$. The trading position at time period t is denoted as $F_t \in \{-1, 0, 1\}$, representing short, neutral, and long trading positions, respectively [14, 36]. It is important to note that transaction costs are incurred only if the trading position changes from $t-1$ to t, i.e., when $F_t \neq F_{t-1}$ [14]. Furthermore, in practical applications, the asset price differences r_t^j and r_t^f are typically standardized [13].

Alternatively, one may choose to exclusively trade the risky asset j and disregard the risk-free rate r_t^f by setting it equal to zero in Equation 2.11 [13]. In this scenario, the realized trading profit π_t for the trade at time period t is defined as follows:

$$\pi_t = \beta\{F_{t-1}r_t^j - |F_t - F_{t-1}|\delta\} \tag{2.12}$$

with the same notations used as in Equation 2.11. In the context of HFT, the assumption that $r_t^f = 0$ can be supported by the insignificant variations in risk-free asset prices over short time periods and the desire to avoid transaction costs associated with purchasing the risk-free asset.

It is worth noting that both Equations 2.11 and 2.12 implicitly assume an additional condition, which is that the transaction costs incurred when switching between short and long positions are twice as high as those for switching between neutral and long/short trading positions [14, 36]. This assumption is made to discourage frequent and unnecessary switching between different trading positions, a behavior that the trading model aims to mitigate, as discussed in the Subsection 2.3.1.

Additionally, the trading models defined in Equations 2.11 and 2.12 assume negligible stock loan fees, meaning there are no costs associated with borrowing the risky asset when taking a short position. This assumption holds true in the context of HFT [52].

It is important to recognize that the trading models defined in Equations 2.11 and 2.12 introduce path-dependence. This arises from the inclusion of transaction costs, which are influenced by the difference between the trading position at time step t, denoted as F_t, and the trading position at time step $t-1$, denoted as F_{t-1}, as time progresses.

Realized Trading Return

It is crucial to highlight that to calculate the rate of return R_t from the trading profits π_t defined in Equation 2.12, one can assume, without any loss of generality, that the trading position size at time period t is $\beta_t = \frac{1}{z^j_{t-1}}$. Thus, the expression for Equation 2.12 can be presented as follows:

$$\begin{aligned} R_t &= \frac{1}{z^j_{t-1}}\{F_{t-1}r^j_t - |F_t - F_{t-1}|\delta\} \\ &= \frac{1}{z^j_{t-1}}\{F_{t-1}(z^j_t - z^j_{t-1}) - |F_t - F_{t-1}|\delta\} \\ &= \frac{F_{t-1}(z^j_t - z^j_{t-1}) - |F_t - F_{t-1}|\delta}{z^j_{t-1}}. \end{aligned} \tag{2.13}$$

In the framework defined in Equation 2.13, it is essential to note that the trading position size varies inversely with the risky asset j price at time $t-1$.

Alternatively, the price change of the risky asset j can be defined as $r^{*j}_t := \ln \frac{z^j_t}{z^j_{t-1}}$ [52]. Subsequently, the rate of return R_t can be directly formulated as follows:

$$R_t = \beta\{F_{t-1}r^{*j}_t - |F_t - F_{t-1}|\delta\}. \tag{2.14}$$

Trading Performance Criteria

For a risk-insensitive trader, the simplest and most intuitive performance metrics is the trading profit [35].

Instead of solely focusing on maximizing profits, modern fund managers generally strive to maximize risk-adjusted returns, as recommended by the Modern Portfolio Theory [29]. Sharpe ratio (SR) is one of the most popular risk-adjusted trading performance metric [34, 40, 41]. The SR, denoted as S, is defined as

$$S = \frac{\mu - r_f}{\sigma}, \tag{2.15}$$

where μ is an expected return of a single risky asset or a portfolio of risky assets, r_f is a risk-free interest rate, and σ is a standard deviation of a single risky asset or a portfolio of risky assets [4, 40, 41].

In practical applications, the SR at a time period T is typically calculated by substituting the sample mean rate of excess return for μ and the sample standard deviation of excess return for σ in Equation 2.15 [4, 34, 40, 36]:

$$S_T = \frac{\text{Average}(\tilde{R_T})}{\text{Standard deviation}(\tilde{R_T})}, \tag{2.16}$$

where $\tilde{R_T} = R_T - R_T^f$ is the excess return of a risky asset with R_T^f being the risk-free interest rate at time period T,

$$\text{Average}(\tilde{R_T}) = \frac{\sum_{t=1}^{T} \tilde{R_t}}{T}, \tag{2.17}$$

$$\text{Standard deviation}(\tilde{R_T}) = \sqrt{\frac{\sum_{t=1}^{T} (\tilde{R_t} - \text{Average}(\tilde{R_t}))^2}{T-1}} \tag{2.18}$$

by definition of an unbiased estimator of sample standard deviation. For simplicity, let us use the concept of realized trading return denoted by R_t defined in Equations 2.13 or 2.14 instead of the excess return $\tilde{R_t}$. In the context of HFT this is a reasonable simplification.

Standard deviation can be additionally expressed through the first $\left(\frac{\sum_{t=1}^{T} R_t}{T}\right)$ and second $\left(\frac{\sum_{t=1}^{T} R_t^2}{T}\right)$ raw moments of the return distribution R_t as follows:

$$\text{Standard deviation}(R_T) = \left(\frac{T}{T-1}\right)^{1/2} \left(\frac{\sum_{t=1}^{T} R_t^2}{T} - \left(\frac{\sum_{t=1}^{T} R_t}{T}\right)^2\right)^{1/2}. \tag{2.19}$$

This result is presented in the reference [36]. For a detailed step-by-step derivation process of this result, please refer to Appendix A.1.

Let us define the first moment of the R_t distribution for the T returns

$$A_T = \frac{1}{T}\sum_{t=1}^{T} R_t. \tag{2.20}$$

Let us define the second moment of the R_t distribution for the T returns

$$B_T = \frac{1}{T}\sum_{t=1}^{T} R_t^2. \tag{2.21}$$

Let the normalizing factor for T returns be

$$K_T = \left(\frac{T}{T-1}\right)^{1/2}. \tag{2.22}$$

Let the SR for a set of T returns R_t denoted as S_T be defined by utilizing estimates of the first A_T (refer to Equation 2.20) and second B_T (refer to Equation 2.21) raw moments of the distributions of the returns [36].

Let us substitute Equations 2.20, 2.21 and 2.22 into Equation 2.19:

$$\text{Standard deviation}(R_T) = K_T(B_T - A_T^2)^{1/2}. \tag{2.23}$$

Next, after substitution of Equations 2.20 and 2.23 into Equation 2.16 one obtains

$$S_T = \frac{A_T}{K_T(B_T - A_T^2)^{1/2}}. \tag{2.24}$$

The result is represented in the relevant literature [36].

To train the system to maximize the SR at each trade, it is required to calculate the derivative of the SR with respect to the trading system parameters at each time step t.

The following result is pertinent to trading a single risky asset [36]. The total derivative of the Sharpe ratio defined in Equation 2.24 with respect to the trading system parameters denoted as $\boldsymbol{\theta}_\mathbf{t}$ at time period t is

$$\frac{\mathrm{d}S_T}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} = \frac{1}{T}\sum_{t=1}^{T}\left\{\frac{B_T - A_T R_t}{K_T(B_T - A_T^2)^{3/2}}\right\}\left\{\frac{\mathrm{d}R_t}{\mathrm{d}F_t}\frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}}\right\}. \tag{2.25}$$

This result is presented in the reference [36]. For a comprehensive and detailed derivation process of this result, kindly consult Appendix A.2.

On-line Learning

In *on-line optimization problems*, the input is received and processed in an online manner, and the output is generated in an online manner as well [2]. An *on-line algorithm* must decide how to act on incoming items of information without any knowledge of future inputs [2]. Numerous financial problems exhibit an on-line nature [2], and this research project specifically addresses a one-way trading problem in that context.

Applying on-line performance functions in the financial context offers several benefits, including expediting the learning process by enabling parameter updates during each forward pass through the training data and adapting to dynamic market conditions in real-time during live trading [36].

Calculating the SR defined in Equation 2.16 involves computing the mean and standard deviation of the returns of the trading strategy at each time period. To compute the mean and standard deviation, the algorithm needs to access all the past returns up to that time step. As a result, at each new time step, the algorithm must recompute the mean and standard deviation of the trading returns, given that a new trading return is generated. This leads to a quadratic time complexity of $O(T^2)$ of the algorithm, where T represents the number of time steps or data points.

A Differential Sharpe ratio (DSR) is offered in reference [36] to aid in on-line optimization. DSR is discussed in the following subsection.

Differential Sharpe Ratio

To begin with, the recursive estimates of the first and second moments of the return distributions are established:

$$\begin{aligned} A_T &\overset{(1)}{=} \frac{1}{T}R_T + \frac{T-1}{T}A_{T-1}, \\ B_T &\overset{(2)}{=} \frac{1}{T}R_T^2 + \frac{T-1}{T}B_{T-1} \end{aligned} \tag{2.26}$$

with the same notations used as in Equations 2.20 and 2.21 [36].

I will show below how the equality 1 from the Equation 2.26 is derived:

$$\begin{aligned} A_T &= \frac{1}{T}\sum_{t=1}^{T} R_t \\ &= \frac{1}{T}(R_T + \sum_{t=1}^{T-1} R_t) \\ &= \frac{1}{T}(R_T + (T-1)A_{T-1}) \\ &= \frac{1}{T}R_T + \frac{1}{T}(T-1)A_{T-1} \\ &= \frac{1}{T}R_T + \frac{T-1}{T}A_{T-1}. \end{aligned}$$

Note that equality 2 from the Equation 2.26 is obtained in a similar manner.

Next, an Exponential Moving Average Sharpe ratio (EMA SR) within the time scale η^{-1} is introduced, where $\eta \in [0, 1]$ is exponential moving average decay rate. The definition is similar to the Equality 2.24 related to the classical SR.

$$S_t = \frac{A_t}{K_\eta(B_t - A_t^2)^{1/2}}, \tag{2.27}$$

where

$$\begin{aligned} A_t &\overset{(1)}{=} \eta R_t + (1-\eta)A_{t-1}, \\ B_t &\overset{(2)}{=} \eta R_t^2 + (1-\eta)B_{t-1}, \\ K_\eta &\overset{(3)}{=} \left(\frac{1-\eta/2}{1-\eta}\right)^{1/2}. \end{aligned} \tag{2.28}$$

In the equality 3 from Equation 2.28, K_η is a constant factor irrelevant for trading system optimization [36].

Note that in equalities 1 and 2 from Equation 2.28 the influence of prior inputs diminishes exponentially over time. For instance, the equality 1 from Equation 2.28 can be expressed as

$$\begin{aligned} A_t &\overset{(1)}{=} \eta R_t + (1-\eta)A_{t-1} \\ &\overset{(2)}{=} \eta R_t + (1-\eta)(\eta R_{t-1} + (1-\eta)A_{t-2}) \\ &\overset{(3)}{=} \eta R_t + \eta(1-\eta)R_{t-1} + (1-\eta)^2 A_{t-2} \\ &\overset{(4)}{=} \dots \\ &\overset{(5)}{=} \eta \sum_{k=0}^{t} (1-\eta)^k R_{t-k}. \end{aligned} \tag{2.29}$$

The final equality from Equation 2.29 is known as an inductive definition of the raw first moment of the return distribution R_t denoted as A_t. The equality 1 in Equation 2.29 is constructed in a similar manner to the equality 1 in Equation 2.26. In the equality 2 from Equation 2.29, a recursive definition of $A_{t-1} = \eta R_{t-1} + (1-\eta)A_{t-2}$ is substituted. Moving on to the equality 3 in Equation 2.29, the brackets are expanded. As we progress, a recursive structure can be observed in the way the weighting factor for previous inputs decreases exponentially. Lastly, the equality 5 presents a sparse representation of equality 3. Similarly, $B_T = \eta \sum_{k=0}^{t}(1-\eta)^k R_{t-k}^2$. As $\eta \in [0,1]$, $(1-\eta) \in [0,1]$ and the factor $(1-\eta)^k$ indeed shrinks as $k \to \infty$.

Further, the update equation for the exponential moving estimates can be reformulated. The equality 1 from Equation 2.29 can be rephrased using simple algebraic manipulations as follows

$$\begin{aligned} A_t &= \eta R_t + (1-\eta)A_{t-1} \\ &= \eta R_t + A_{t-1} - \eta A_{t-1} \\ &= A_{t-1} + \eta(R_t - A_{t-1}). \end{aligned}$$

By introducing the definition $\Delta A_t \coloneqq R_t - A_{t-1}$ similar to the reference [36], we can express the update equation in the following manner:

$$A_t = A_{t-1} + \eta \Delta A_t. \tag{2.30}$$

Similarly, the update equation for B_T can be written as follows:

$$\begin{aligned} B_t &= B_{t-1} + \eta(R_t^2 - A_{t-1}) \\ &= B_{t-1} + \eta \Delta B_t, \end{aligned} \tag{2.31}$$

using the definition $\Delta B_t \coloneqq R_t^2 - B_{t-1}$ [36].

Note that the EMA SR can be represented as a N^{th}-order Taylor expansion in time constant η of the Equation 2.27:

$$S_t \approx S_{t-1} + \eta \frac{\mathrm{d}S_t}{\mathrm{d}\eta}|_{\eta\to 0} + \frac{1}{2}\eta^2 \frac{\mathrm{d}^2 S_t}{\mathrm{d}\eta^2}|_{\eta\to 0} + \ldots + \frac{1}{N!}\eta^N \frac{\mathrm{d}^N S_t}{\mathrm{d}\eta^N}|_{\eta\to 0} + O(\eta^{N+1}), \tag{2.32}$$

where the big O notation is used to emphasize that as η approaches 0, the terms of order $N+1$ or higher become insignificant and have a negligible impact on the overall result [36].

Note that the first-order expansion term in the Equation 2.32 ($\eta \frac{\mathrm{d}S_t}{\mathrm{d}\eta}|_{\eta\to 0}$) is the only one that depends on the return R_t at time period time t (through ΔA_t and ΔB_t) [36]. Remember that

$$S_t = \frac{A_{t-1} + \eta \Delta A_t}{K_\eta (B_{t-1} + \eta \Delta B_t - (A_{t-1} + \eta \Delta A_t)^2)^{1/2}},$$

where $\Delta A_t = R_t - A_{t-1}$ and $\Delta B_t = R_t^2 - B_{t-1}$. This result is obtained by substitution of Equations 2.30 and 2.31 into Equation 2.27.

Following this, in the reference [36], the DSR is defined as the first-order expansion term in the Equation 2.32:

$$D_t := \frac{\mathrm{d}S_t}{\mathrm{d}\eta} = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}, \tag{2.33}$$

with the same notations used as in the Equations 2.30 and 2.31. Note that the current return R_t appears in this expression solely in the numerator, through ΔA_t and ΔB_t [36].

The paper [36] introduces a derivative of the DSR with respect to the trading return R_t, which further proves beneficial for optimizing the trading system. Along with the simplified form, I provide some intermediate steps leading to the final result:

$$\begin{aligned}\frac{\mathrm{d}D_t}{\mathrm{d}R_t} &= \frac{\mathrm{d}\left\{\dfrac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}\right\}}{\mathrm{d}R_t} \\ &= \frac{1}{(B_{t-1} - A_{t-1}^2)^{3/2}}\frac{\mathrm{d}(B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t)}{\mathrm{d}R_t} \\ &= \frac{1}{(B_{t-1} - A_{t-1}^2)^{3/2}}\frac{\mathrm{d}(B_{t-1}(R_t - A_{t-1}) - \frac{1}{2}A_{t-1}(R_t^2 - B_{t-1}))}{\mathrm{d}R_t} \\ &= \frac{B_{t-1} - A_{t-1}R_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}. \end{aligned} \tag{2.34}$$

The DSR offers advantages over the sliding the time window method for calculating the SR under the on-line learning framework for the following reasons outlined in the reference [36]:

- **recursive updating is facilitated** as there is no need to recompute the average and standard deviation of returns for the entire trading history when updating the EMA SR defined in the Equation 2.27 for the most recent time period (the first A_t and the second B_t raw moments of the return distribution R_t are updated according to the rules defined in the Equation 2.30 and 2.31);
- **allows for efficient online optimization** as the calculation of DSR denoted as D_t defined in the Equation 2.33 and its first order derivative with respect to R_t $\frac{\mathrm{d}D_t}{\mathrm{d}R_t}$ defined in the Equation 2.34 is computationally inexpensive, since it can be done using the previously computed moving averages A_{t-1} and B_{t-1} along with the

current return R_t;

- **recent returns are given more weight in DSR compared to older returns**, it is shown that in the expression for the first raw moment of the return distribution from Equation 2.29, the weights of prior returns decline exponentially over time;
- **the method offers interpretability** as the DSR separates the contribution of the current return R_t to the EMA SR.

The optimization of the DSR necessitates $O(T)$ calculations because of the efficient recursive updating, whereas the naive optimization of the SR requires $O(T^2)$ calculations, as previously mentioned.

Performance functions for Direct Reinforcement Learning

DRL involves tuning the parameters of a system to maximize the expected payoff or reward resulting from the system's actions [34]. Given a trading system with the structure defined in Equation 2.8 and trading returns defined as in Equation 2.13 or 2.13, the objective of the DRL algorithm is to maximize a relevant performance function subject to the trading system's parameters [36]. Two path-dependent performance functions are elaborated upon below.

1. In the reference [36], a path-dependent performance function is expressed as a function of the sequence of T trading returns

$$U_T = U(R_1, R_2, \dots, R_T). \tag{2.35}$$

In the reference [36], the gradient of U_T with respect to the system's parameters at time step t denoted as $\boldsymbol{\theta_t}$ after a sequence of T trades is

$$\frac{\mathrm{d}U_T(\boldsymbol{\theta_t})}{\mathrm{d}\boldsymbol{\theta_t}} = \sum_{t=1}^{T} \frac{\mathrm{d}U_T}{\mathrm{d}R_t} \left\{ \frac{\mathrm{d}R_t}{\mathrm{d}F_t}\frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right\} = \Delta\boldsymbol{\theta_t}. \tag{2.36}$$

For an extensive and in-depth explanation of this result, you can find a step-by-step derivation in Appendix A.3.

2. DSR at time step t denoted as D_t is a also a path-dependent performance function and can be interpreted as an adaptive utility function [36]. In the reference [34], a gradient of D_t with respect to the the system's parameters at time step t denoted as $\boldsymbol{\theta}_\mathbf{t}$ is

$$\frac{\mathrm{d}D_t(\boldsymbol{\theta}_\mathbf{t})}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} = \frac{\mathrm{d}D_t}{\mathrm{d}R_t}\left\{\frac{\mathrm{d}R_t}{\mathrm{d}F_t}\frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}}\right\} = \Delta\boldsymbol{\theta}_\mathbf{t}. \tag{2.37}$$

The explanation of the derivations is fairly similar to the one given in Appendix A.3. Note that the expression $\frac{\mathrm{d}D_t}{\mathrm{d}R_t}$ is defined in the Equation 2.34.

In reference to the realized trading return defined in Equation 2.14, the derivatives of the return function outlined in the reference [20] are as stated below:

$$\frac{\mathrm{d}R_t}{\mathrm{d}F_t} = -\beta \mathrm{sgn}(F_t - F_{t-1})\delta, \tag{2.38}$$

$$\frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} = r_t^j + \beta \mathrm{sgn}(F_t - F_{t-1})\delta. \tag{2.39}$$

Remember that sgn stands for a *sign* function. As noted in the relevant literature, the total derivatives $\frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}}$ from Equation 2.37 are contingent on the complete sequence of past trades, therefore, to calculate and optimize these total derivatives, a recurrent algorithm is required [36, 34]. In the reference [34], the following recursive update equation is presented for a parameter gradient that takes into account the temporal dependencies in a sequence of decisions

$$\frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} = \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} + \frac{\mathrm{d}F_t}{\mathrm{d}F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}}. \tag{2.40}$$

Note that the Equations 2.36, 2.37, and 2.40 assume that F_t is a differentiable function [34]. As it has been discussed before in the Subsection 2.3.1, the differentiable *tanh* function can be used for the trading position scaling to a $[-1, 1]$ interval, see the Equation 2.10.

The trading system's parameters can be iteratively updated using stochastic gradient ascent since the trading performance function is set to be maximized. The update rule is defined as follows:

$$\boldsymbol{\theta}_{\mathbf{t}} = \boldsymbol{\theta}_{\mathbf{t-1}} + \rho\Delta\boldsymbol{\theta}_{\mathbf{t}}, \tag{2.41}$$

where $\rho \in [0, 1]$ represents an adjustable learning rate, and $\Delta\boldsymbol{\theta}_{\mathbf{t}}$ is defined in the Equation 2.36 or Equation 2.37, depending on the chosen trading performance function [36].

The current research will concentrate on the DSR as the trading performance metric, primarily due to its outlined advantages and the substantial reduction in the computational workload required for its optimization.

Direct Reinforcement Learning Extensions

In the reference [14], it is highlighted that a significant limitation of DRL is its lack of a feature learning component to efficiently capture the complex financial market conditions. In the reference [14], the approach to include feature learning involves integrating a widely used Deep Learning (DL) into the framework of DRL. In general, DL builds a DNN that progressively transforms information across layers [14]. This deep representation approach fosters the creation of highly informative features tailored to a particular learning objective [14].

In the reference [13], DRL was expanded to consider additional inputs beyond historical returns and the current position F_t. Experiments were carried out to incorporate signals derived from technical indicators [13]. However, this approach did not result in an enhancement of performance [13]. This observation suggests that the DRL algorithm effectively leverages the information within past returns time series [13].

In the study referenced as [52], a Genetic Algorithm (GA) was incorporated into a DRL daily equity trading system. The purpose of GA was to select indicators from a pool that consisted of five frequently employed technical indicators, one financial fundamental indicator, and two volatility indicators. The aim of this integration was to enhance the overall trading performance. It has been discovered that the Relative Strength Index (a technical indicator), the Price to Earnings ratio (a fundamental indicator), along with the conditional volatility and implied volatility indicators, leads to the most substantial enhancement in the SR.

The objective of the current research is to integrate some of the presented extensions of DRL into the framework.

2.4 High-Frequency Financial Data Analysis

Stylized Facts related to High-frequency Financial Time Series Data

Allow me to provide further details regarding high-frequency financial data, which refers to observations recorded on a daily basis or at an even finer time scale [49]. Interestingly, high-frequency data was uncommon in econometrics due to limited data availability until the late 1980s [12]. High-frequency data possesses the following distinct attributes:

- irregularly spaced time intervals between transactions,
- discrete-valued prices,
- occurrences of multiple transactions within a single second,
- at the highest frequency, the mid-price (an average between bid and ask prices) is influenced by market microstructure effects,
- as the frequency of data increases, the distributions of returns exhibit progressively heavier tails,
- the presence of a daily periodic or diurnal pattern known as **intraday seasonality** or **diurnality** (for instance, return volatility clustering) [49, 12].

Return volatility is frequently characterized as the (conditional) standard deviation of returns. Intraday return volatility clustering pertains to the empirical observation in numerous financial time series where periods of elevated volatility are commonly succeeded by subsequent periods of heightened volatility, and conversely, periods of reduced volatility are typically followed by subsequent periods of lower volatility. This is often attributed to elevated trading volume or price fluctuations after the market absorbs the new information [5]. One of the intriguing yet complex aspects of volatility is its unobservability. In other words, true volatility is latent.

The **Efficient Market Hypothesis** was articulated by Eugene Fama in 1991 as the notion that "security prices fully incorporate all accessible information" [19]. Burton G. Malkiel is widely recognized as a prominent proponent of the efficient market hypothesis and has played a significant role in advocating for passive investing [28]. Malkiel adopts a definition of efficient financial markets that entails markets wherein investors cannot achieve returns greater than the average without assuming higher-than-average risks [28]. Hence, according to Malkiel's viewpoint, investors lack a reliable method to systematically capitalize on any existing anomalies or patterns in returns [28].

In practice, it is uncommon to observe seasonality in intraday returns [49], a fact reinforced by a prevailing consensus that securities markets were highly effective in promptly incorporating information concerning individual stocks and the broader stock market.

Let us concentrate on empirical observations related to the Foreign Exchange (FX) market where currencies are traded. The FX market remains operational 24 hours a day, exhibiting an irregular intraday volatility pattern [12]. The observed patterns can be rationalized by taking into account the composition of the global market, comprising three primary regions spanning distinct time zones: America, Europe, and East Asia [12, 5]. For instance, a lunch break in the European and East Asian markets aligns with the least intraday volatility periods in the FX market for an exchange rate between American and European currencies [12]. The main daily peak in intraday volatility for the exchange rate between American and European currencies within the FX market occurs when both the American and European markets are active [12].

Intraday seasonality introduces a deterministic trend, creating time-related non-stationary data. This can lead to spurious regressions with misleading coefficients and inconsistent estimators. Eliminating seasonality from the time series, known as **seasonal adjustment**, produces a **seasonally adjusted time series** [16]. An additional rationale for conducting seasonal adjustment is the fact that the existence of seasonality might be described as concealing fluctuations in more economically significant components [21]. Forecasting is usually carried out utilizing the seasonally adjusted time series, with the seasonal component being incorporated afterward [21]. The prevalent model to capture intraday seasonality is a Pure Diurnal Dummy Variable Model that is discussed in the following subsection.

Seasonal Adjustment: Pure Diurnal Dummy Variable Model

The pure diurnal dummy variable model consists of L dummy variables representing time intervals of consistent length (typically 30 minutes) throughout a trading day, accompanied by log returns r_t measured over briefer time periods (e.g., 5 minutes):

$$r_t = \sum_{i=1}^{L} \beta_i D_{i,t} + r_t^{deseasonalized}, \tag{2.42}$$

where $D_{i,t} = 1$ if the trade t occurred during the i^{th} 30-minute time interval, and $D_{i,t} = 0$ otherwise [16, 37, 5]. In this context, within Equation 2.42, the term $\sum_{i=1}^{L} \beta_i D_{i,t}$ will be denoted as the *seasonal* component of the log returns, while $r_t^{deseasonalized}$ will be identified as the *seasonally adjusted* log return series.

The dummy variables D_i are also referred to as seasonal factors and are expected to be identical $\forall i \geq 0$ in the absence of intraday seasonality [16, 37]. The following hypothesis is tested: $\mathcal{H}_0 : D_1 = D_2 = \ldots = D_L$ (intraday seasonality is absent) against the alternative $\mathcal{H}_1 : D_i \neq D_j$ for at least one pair $(i, j), i \neq j$ (intraday seasonality is present) [37].

Under the null hypothesis $\mathcal{H}_0$, the test statistic $\chi^2_{st} = \sum_{i=1}^{L} \left(\dfrac{\hat{\beta}_i - \bar{\beta}}{s_i} \right)^2$ is asymtotically distributed as a chi-squared distribution with $L - 1$ degrees of freedom denoted as χ^2_{L-1}, where $\bar{\beta} = \dfrac{\sum_{i=1}^{L} \beta_i}{L}$ and s_i is a heteroskedasticity-consistent (robust) standard error if intraday returns exhibit heteroskedasticity and non-robust standard error otherwise [37]. The null hypothesis $\mathcal{H}_0$ is rejected in support of the alternative hypothesis $\mathcal{H}_1$ when the test statistic χ^2_{st} surpasses the critical value $\chi^2_{L-1}(\alpha)$, which is an upper $100(1 - \alpha)$ percentile of χ^2_{L-1} distribution. This critical value is determined based on the predefined significance level α, typically set at 5%.

It is worth noting that observing clear and distinct day-of-the-week patterns in intraday exchange rate returns is relatively uncommon [5].

Modelling Asset Return

Return series often exhibit slight autocorrelation, which can be addressed by incorporating an ARMA model for conditional mean of returns denoted as $\mu_t := \mathbb{E}_{t-1}(r_t|I_{t-1})$, where I_{t-1} denotes a set of all available information at time step $t-1$. ARMA(p,q) is a class of stationary times series models expressed as follows:

$$\begin{aligned} r_t &= \mu_t + \epsilon_t, \\ \mu_t &= \omega + \sum_{i=1}^{p} \alpha_i r_{t-i} + \sum_{j=1}^{q} \beta_i \epsilon_{t-j}, \\ \epsilon_t &\sim \text{i.i.d.}(0, \sigma^2). \end{aligned} \tag{2.43}$$

A significant aspect of consideration is the distributional assumption related to ϵ_t, which is frequently assumed to conform to a normal distribution. Nevertheless, if the data displays notable non-Gaussian features, such as heavy tails, then an alternative distribution (for instance, Student-t distribution) that better matches the actual observations should be taken into account.

The well-known Box-Jenkins methodology for building asset return models will be used in the current research [8].

Modelling Asset Return Volatility

While many textbook models assume constant volatilities, it is widely acknowledged by both finance scholars and practitioners that financial market volatility varies significantly over time [3]. The prevailing consensus among researchers suggests that a more accurate portrayal of the data generation process is achieved through a conditional heteroskedastic model, rather than relying on an unconditional distribution [12]. The conditional asset return variance is going to be denoted as $\mathbb{V}_{t-1}(r_t|I_{t-1}) := \sigma_t^2$, where I_{t-1} denotes a set of all available information at time step $t-1$. Note that $r_t = \mu_t - \epsilon_t$ (refer to Subsection 2.4.3). Hence, $\mathbb{V}_{t-1}(r_t|I_{t-1}) = \mathbb{V}_{t-1}(\mu_t - \epsilon_t|I_{t-1}) = \mathbb{V}_{t-1}(\epsilon_t|I_{t-1})$ as μ_t is known at time step $t-1$, $\mu_t \in I_{t-1}$, so it can be treated as a constant.

Allow me to provide a detailed explanation of the ARCH process, which stands for autoregressive conditional heteroskedasticity process, along with its generalization, the GARCH process, which stands for generalized autoregressive conditional heteroskedasticity process.

ARCH model was introduced in 1982 by a Nobel laureate Robert Engle as an effort to incorporate past information into influencing the forecast variance [17]. This non-linear model demonstrates that the return variance is characterized by *conditional* heteroskedasticity. This model was primarily introduced to account for the following stylized facts observed in financial empirical data: volatility clustering and volatility mean-reversion. Let the conditional mean of returns r_t at time step t be μ_t and a random innovation term be ϵ_t, meaning that $r_t = \mu_t + \epsilon_t$. For instance, μ_t may take some form of ARMA model. The ARCH(p) model is defined by

$$\epsilon_t = \sigma_t \nu_t, \nu_t \sim \text{i.i.d.}(0, 1), \tag{2.44}$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \ldots + \alpha_p \epsilon_{t-p}^2, \tag{2.45}$$

where $\alpha_0 > 0$, and $\alpha_i \geq 0, \forall i \geq 0$ [49]. Note that the residuals ϵ_t in Equation 2.44 exhibit a multiplicative structure. Additionally, note that Equations 2.44 and 2.45 exhibit not only a model for conditional variance for r_t, but also a one-step-ahead conditional variance forecast.

Note that the parameter p, number of lags included in the ARCH(p) model, can be rather challenging, given the absence of a straightforward algorithm. In addition, a frequent requirement is the inclusion of a relatively extended lag in the conditional variance equation, making the ARCH model extremely non-parsimonious [7].

The idea behind the ARCH model generalization is to choose a more parsimonious model. GARCH(1,1) model was first introduced by Bollerslev in 1986 [7]. Let the disturbance term ϵ_t process be such that

$$\epsilon_t = \sigma_t \nu_t, \nu_t \sim \text{i.i.d.}(0, 1), \tag{2.46}$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2, \tag{2.47}$$

where $\alpha_0 > 0$, $\alpha_1 \geq 0$, $\beta_1 \geq 0$, and $\alpha_1 + \beta_1 < 1$ [49].

The primary findings indicate that there is no evidence of the GARCH(1,1) model being outperformed by extended GARCH-type models [22]. This conclusion is drawn from an evaluation utilizing daily realized variance, which is constructed from high-frequency exchange rate data [22].

Conditional Volatility in the Presence of Intraday Seasonality

In practical terms, seasonality in intraday volatility often poses a more significant challenge than seasonality in intraday returns. GARCH-type models are suitable for describing data with a constant unconditional variance, rendering them inappropriate for raw intraday data [37]. A frequent solution involves dividing the conditional volatility into

- a **deterministic** (seasonal), and
- a **stochastic** (GARCH)

components. An elegant framework for modeling conditional volatility in the presence of intraday seasonality, as presented in [37], is outlined below.

Let r_t be a log-return and μ_t be the conditional mean of log-return r_t at time step t. μ_t can be decomposed into an ARMA(p,q) part and a possible seasonal effect denoted as S_t^r. Then, the return model can be represented as follows:

$$\begin{aligned} r_t &= \mu_t + \epsilon_t, \\ \mu_t &= \omega + \sum_{i=1}^{p} \alpha_i r_{t-i} + \sum_{j=1}^{q} \beta_i \epsilon_{t-j} + S_t^r, \\ \epsilon_t &\sim (0, \sigma_t^2). \end{aligned} \tag{2.48}$$

Let S_t^ϵ be a seasonal component of residual ϵ_t. Let ϵ_t^* be a deseasonalized residual and σ_t^* be a deseasonalized return volatility. Then,

$$\epsilon_t \overset{(1)}{=} \sigma_t \nu_t \overset{(2)}{=} S_t^\epsilon \sigma_t^* \nu_t \overset{(3)}{=} S_t^\epsilon \epsilon_t^*, \tag{2.49}$$

where $\nu_t \sim$ i.i.d.$(0,1)$. Let me briefly explain the Equation 2.49. The equality 1 illustrates the multiplicative structure of residuals (as indicated by a GARCH model, as shown in the Equation 2.46). The equality 2 demonstrates that the conditional volatility σ_t is divided into a seasonal component S_t^ϵ and a GARCH component σ_t^*. The equality 3 shows that the GARCH component σ_t^*, along with the independent and identically distributed variable ν_t constitute a deseasonalized residual denoted as ϵ_t^*.

The ultimate objective of these subsequent manipulations is to determine an estimate for the seasonal component S_t^ϵ within the log return residual ϵ_t.

The GARCH model is based on a multiplicative framework, as it has been mentioned earlier. However, the aforementioned seasonal adjustment procedure (refer to the Subsection 2.4.2) follows an additive structure. As a result, it is necessary to convert the GARCH model into an additive form, for instance, by employing logarithms. Additionally, it is important to highlight that the input to a logarithmic function must always be non-negative. Consequently, prior to applying logarithms, both the right- (ϵ_t) and left-hand ($S_t^\epsilon \epsilon_t^*$) sides of the Equation 2.49 need to be squared.

$$\begin{aligned} \epsilon_t^2 &= S_t^{\epsilon 2} \epsilon_t^{*2}, \qquad (2.50) \\ \ln(\epsilon_t^2) &= \ln(S_t^{\epsilon 2}) + \ln(\epsilon_t^{*2}), \\ \ln(\epsilon_t^2) &= \ln(S_t^{\epsilon 2}) + u_t, \qquad (2.51) \end{aligned}$$

where $u_t = \ln(\epsilon_t^{*2})$.

Next, to produce an estimate $\widehat{\ln(S_t^{\epsilon 2})}$ of the logarithm of the squared seasonal component $\ln(S_t^{\epsilon 2})$, one can employ a pure diurnal dummy variable model (as discussed in the Subsection 2.4.2):

$$\begin{aligned} \ln(\epsilon_t^2) &\overset{(1)}{=} \widehat{\ln(S_t^{\epsilon 2})} + \widehat{u_t}, \\ \widehat{\ln(S_t^{\epsilon 2})} &\overset{(2)}{=} \sum_{i=1}^{16} \beta_i D_{i,t}. \end{aligned} \qquad (2.52)$$

Further, exponentiate both sides of the equality 1 from Equation 2.52 to arrive at the following expression:

$$\begin{aligned} \epsilon_t^2 &= \exp\{\widehat{\ln(S_t^{\epsilon 2})} + \widehat{u_t}\} \\ &= \exp\{\widehat{\ln(S_t^{\epsilon 2})}\} \exp\{\widehat{u_t}\}. \qquad (2.53) \end{aligned}$$

Note that $\exp\{\ln(S_t^{\epsilon 2})\}$ is a biased estimator of $S_t^{\epsilon 2}$. Therefore, it is usually assumed that $\widehat{S_t^{\epsilon 2}} := \widehat{\mathbb{E}[\epsilon_t^2|S_t^{\epsilon}]}$.

Next, the value $\mathbb{E}[\epsilon_t^2|S_t^{\epsilon}]$ can be expressed as follows:

$$\begin{aligned}\mathbb{E}[\epsilon_t^2|S_t^{\epsilon}] &\overset{(1)}{=} \mathbb{E}[\exp\{\widehat{\ln(S_t^{\epsilon 2})}\}\exp\{\widehat{u_t}\}|S_t^{\epsilon}] \\ &\overset{(2)}{=} \exp\{\widehat{\ln(S_t^{\epsilon 2})}\}\mathbb{E}[\exp\{\widehat{u_t}\}|S_t^{\epsilon}] \\ &\overset{(3)}{=} \exp\{\widehat{\ln(S_t^{\epsilon 2})}\}\mathbb{E}[\exp\{\widehat{u_t}\}]. \end{aligned} \tag{2.54}$$

Allow me to provide a more detailed explanation of the derivations for Equation 2.54. The equality 1 arises from substituting Equation 2.53 into the conditional expectation operator. The equality 2 is a result of factoring out $\exp\widehat{\ln(S_t^{\epsilon 2})}$ from the conditional expectation operator. This factor represents an expression solely dependent on $S_t^{\epsilon 2}$, which the expectation is conditioned upon. The equality 3 holds true when it is assumed that u_t is independent of S_t^{ϵ}.

Additionally, in order to formulate an estimate for $\mathbb{E}[\epsilon_t^2|S_t^{\epsilon}]$, denoted as $\widehat{\mathbb{E}[\epsilon_t^2|S_t^{\epsilon}]}$, in place of the expectation operator $\mathbb{E}[\exp\widehat{\{u_t\}}]$ within Equation 2.54, one can make use of a sample average through the following substitution:

$$\begin{aligned}\widehat{S_t^{\epsilon 2}} &\overset{(1)}{:=} \widehat{\mathbb{E}[\epsilon_t^2|S_t^{\epsilon}]} \\ &\overset{(2)}{=} \exp\{\widehat{\ln(S_t^{\epsilon 2})}\}\mathbb{E}[\exp\{\widehat{u_t}\}] \\ &\overset{(3)}{\approx} \exp\{\widehat{\ln(S_t^{\epsilon 2})}\}\frac{\sum_{t=1}^{T}\exp\{\widehat{u_t}\}}{T}, \end{aligned} \tag{2.55}$$

where T is a sample size. equality 1 corresponds to the assumption introduced earlier. equality 2 emerges from Equation 2.54. Approximation 3 arises through the replacement of the expectation operator with a sample average.

In conclusion, the deseasonalized residual estimate $\widehat{\epsilon_t^*}$ is calculated using the following procedure and subsequently employed in the estimation of a deseasonalized GARCH(1, 1) model, as an illustration:

$$
\begin{aligned}
\widehat{\epsilon_t^*} &= \frac{r_t - \widehat{\mu_t}}{\sqrt{\widehat{S_t^{\epsilon 2}}}} \qquad (2.56)\\
\epsilon_t^* &\sim \text{i.i.d.}(0, \sigma_t^*)\\
\sigma_t^{*2} &= \omega + \alpha\epsilon_{t-1}^* + \beta\sigma_{t-1}^{*2}.
\end{aligned}
$$

Note that the distribution of ϵ_t^* is not specified and should be assumed through an
1100 *ad hoc* procedure, considering the characteristics of empirical distribution for $\widehat{\epsilon_t^*}$. A it has been mentioned in the Subsection 2.4.1, high-frequency financial time series data tend to exhibit severe kurtosis, hence the common assumption of conditional normality should be loosened by considering alternative fat-tailed distributions such as the Student-t distribution for ϵ_t [6].

In brief, the main goal of conditional volatility modeling is to obtain a one-step-ahead forecast for the conditional variance. As previously mentioned, the GARCH model provides such a forecast. Therefore, the process for obtaining a one-step-ahead forecast for the conditional variance is as follows:

$$
\begin{aligned}
\mathbb{V}_{t-1}(\epsilon_t) &= \sigma_t^2\\
&\approx \widehat{S_t^{\epsilon 2}}\sigma_t^{*2}\\
&= \exp\left\{\widehat{\ln(S_t^{\epsilon 2})}\right\}\frac{\sum_{t=1}^{T}\exp\left\{\widehat{u_t}\right\}}{T}\widehat{\sigma_t^{*2}}. \qquad (2.57)
\end{aligned}
$$

1105 The present research will utilize the outlined framework for seasonal adjustment of intraday return volatility and forecasting intraday return volatility.

3. EMPIRICAL RESULTS

In the literature, the majority of discussions regarding DRL trading have been conducted with a focus on studying high-frequency FX trading [20].

FX spot market holds the distinction of being the world's largest financial market, assuming a pivotal role in high-frequency finance [12]. The market is characterized as highly liquid [12].

The designed trading system will be supplied with historical FX data, and its performance on the distinct dataset — unrelated to its training — will then be logged and analyzed.

Furthermore, as elaborated in the Section 2.3.9, a conditional volatility indicator will be employed to enhance the informativeness of the trading system. A comparison will be made between the trading performance of the original system and its extended version featuring the volatility indicator.

3.1 Historical Foreign Exchange Rate Data Analysis

EUR/GBP High-Frequency Price Data

EUR/GBP accounted for around 2% of the overall trading volume in April 2022 [30]. Historical FX EUR/GBP Open-High-Low-Close (OHLC) price data was acquired through Bloomberg. The data is captured at five-minute intervals and covers the time frame from January 3rd 2023 to August 11th 2023, a total of 408 trading days. The choice of a five-minute frequency is deliberate, as it provides sufficient time for the disruptive effects of market microstructure frictions not to overwhelmingly influence the data [3]. The exchange rate is quoted as the number of pounds sterling (GBP) per euro (EUR).

FX market is accessible around 24 hours during business days. However, the level of liquidity undergoes substantial changes throughout the 24-hour trading period. During active trading hours, the level of liquidity increases, which allows trades to be executed with minimal price fluctuations. Additionally, important events that can significantly impact the market often take place during periods of high trading activity. During active trading hours the new information gets absorbed into market prices rapidly. Furthermore, during non-active trading hours, unusual price patterns might emerge, which could potentially compromise the accuracy of analysis outcomes. This occurs because the influence of individual trades can exert a more pronounced impact on prices, resulting in less predictable movements. Thus, a common practice is to restrict the FX trading to the active hours corresponding to optimal liquidity, starting from 9 a.m. London time and concluding at 5 p.m. London time [13]. As a result, there are a total of 95 five-minute returns for each trading day. A comparable process is carried out in the ongoing research, resulting in a reduction of price observations from 138,956 to 45,453.

The set of $T = 45,453$ observations is divided into a training subset encompassing 70% observations from the initial dataset ($T_{train} = 31,817$) and a test subset encompassing 30% observations from the initial dataset ($T_{test} = 13,636$). The statistical summaries presented below pertain to the training subset.

EUR/GBP 5-Minute Mid Price and Log Return Defined in Bid/Ask Trading Framework

Let bid and ask prices of a EUR/GBP exchange rate (labelled as risky asset j) at time step t be denoted as $z_t^{j,bid}$ and $z_t^{j,ask}$, respectively. Next, mid price used for the calculation of trading profit and return at time step t (as defined in the Subsections 2.3.3 and 2.3.4) is specified as follows [20], [24]:

$$z_t^j = \frac{z_t^{j,bid} + z_t^{j,ask}}{2}.$$

In the present study, historical EUR/GBP 5-minute OHLC price data is employed (refer to Subsection 3.1.1). Bid and ask prices are typically approximated using the lowest and highest transaction prices observed during the period, respectively. The lowest and highest prices of the risky asset j at time step t are denoted as $z_t^{j,low}$ and $z_t^{j,high}$, respectively. Therefore, the mid price series of the risky asset j utilized for computing mid price returns in the present study is determined as a simple average of the lowest and highest prices.

This can be expressed as:

$$z_t^j = \frac{z_t^{j,low} + z_t^{j,high}}{2}. \tag{3.58}$$

Note that the mid price of the risky asset j, denoted as z_t^j from Equation 3.58, will be further referred to as the EUR/GBP 5-minute mid price. Note that the superscript j will be omitted henceforth for convenience.

Subsequently, the logarithm of price change of the risky asset j at time period t, denoted as r_t^{*j} (refer to Subsection 2.3.4), is computed as follows:

$$r_t^{*j} = \ln \frac{z_t^j}{z_{t-1}^j}. \tag{3.59}$$

Note that the mid price return of the risky asset j, denoted as r_t^{*j} from Equation 3.59, will be further referred to as the EUR/GBP 5-minute log return. For simplicity, the superscript j will be omitted in subsequent discussions.

Transaction Costin Bid/Ask Trading Framework

In bid/ask trading, the transaction cost for trading the risky asset j at time period t is computed as half of the spread (difference between the ask price denoted as $z_t^{j,ask}$ and the bid price denoted as $z_t^{j,bid}$), which is defined as $z_t^{j,ask} - z_t^{j,bid}$ [20]:

$$\delta_t^j = \frac{z_t^{j,ask} - z_t^{j,bid}}{2}.$$

As currency trading usually does not involve commissions, no additional transaction costs are typically imposed [20].

Note that in the current study, historical EUR/GBP 5-minute OHLC price data is employed (refer to Subsection 3.1.1). Similar to Subsection 3.1.2, bid and ask prices of the risky asset j are approximated using the lowest and highest transaction prices observed during the period, respectively. Hence, the EUR/GBP 5-minute transaction cost at time step t denoted as δ_t^j is computed as follows:

$$\delta_t^j = \frac{z_t^{j,high} - z_t^{j,low}}{2}, \tag{3.60}$$

where the lowest and highest prices of a risky asset j at time step t are denoted as $z_t^{j,low}$ and $z_t^{j,high}$, respectively. Moving forward, the superscript j will be omitted for simplicity.

Leakage

Leakage occurs when the model gains access to information about future prices, directly or indirectly, during decision-making. Generally, leakage undermines experimental outcomes, as we anticipate improved performance when possessing future information.

All EUR/GBP 5-minute log returns are normalized using the mean and variance of the EUR/GBP 5-minute log returns observed during the training period [20]. Furthermore, in order to prevent leakage, the test dataset is normalized using the mean and variance of the EUR/GBP 5-minute log returns observed during the training period.

Visualization and Descriptive Statistics for EUR/GBP 5-minute Log Returns

In Figure 3.1, the upper panel reveals a prevailing long-term upward trend accompanied by fluctuations in the EUR/GBP 5-minute mid price. The middle panel of the Figure 3.1 visualizes fluctuations in the EUR/GBP 5-minute log return, whereas the bottom panel depicts the dynamics of the standardized EUR/GBP 5-minute log return.

Notably, in all the panels of Figure 3.1, a substantial spike in both the mid price and log returns of EUR/GBP becomes evident, roughly aligning with the later part of September 2022. This occurrence has been linked to the detrimental fiscal policy announcements made by the former British Prime Minister, Liz Truss [44].

After visually inspecting the Figure 3.1, it becomes clear that the return series contains a positive outlier, coinciding with a return over the first 5-minute interval in the morning of September 26, 2022. During the early hours of that Monday morning, the British pound experienced an all-time low against the dollar and suffered a significant drop against the euro [43].

In the bottom panel of the Figure 3.1, there is indication of serial correlation in the amplitude of the returns, volatility clustering is evident. The observed long-run dependence in volatility aligns with relevant findings from previous empirical research [3].

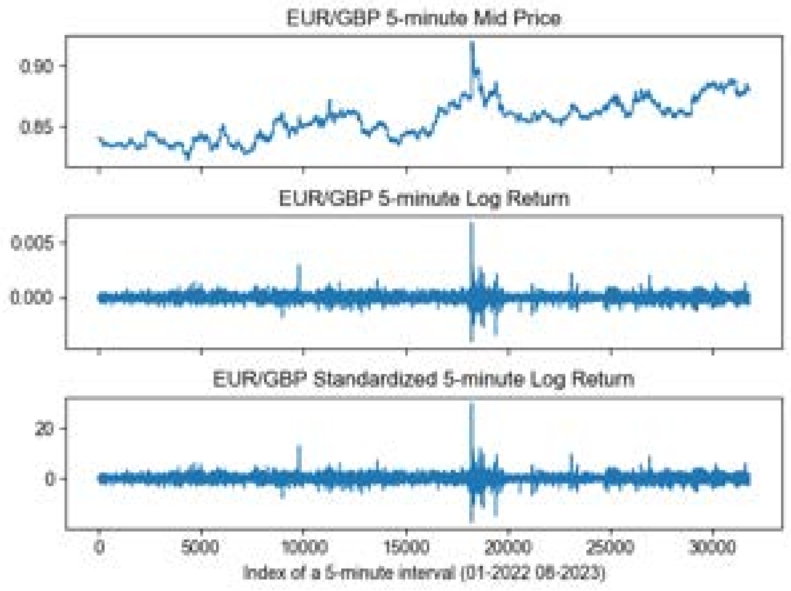


Figure 3.1: The upper panel displays 5-minute mid prices of EUR/GBP during active trading hours. The mid price is calculated as the average between the high and low prices. The middle panel illustrates 5-minute log returns of EUR/GBP, which is the logarithmic difference of the 5-minute mid price series. The bottom panel shows standardized 5-minute log returns of EUR/GBP. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

The further analysis is going to utilize standardized EUR/GBP 5-minute log returns. EUR/GBP 5-minute log returns are standardized by using the mean and standard deviation values for non-standardized log returns provided in Table 3.1. It is worth noting that the final entry in the Table represents the Jarque-Bera test statistic. This test evaluates whether the sample data adheres to a normal distribution by considering the sample skewness and sample kurtosis. Note that the Jarque-Bera test statistic remains consistent for both non-standardized and standardized EUR/GBP 5-minute log returns because standardization does not affect the sample skewness and kurtosis. The associated p-value for the Jarque-Bera test statistic is exceedingly low, indicating that both the non-standardized and standardized EUR/GBP 5-minute log returns do not follow a normal distribution at any meaningful level of significance.

Statistic	**Non-standardized**	**Standardized**
Number of observations	31,817	31,817
Mean	0.000002	0.000000
Standard deviation	0.000229	1.00000
Minimum	-0.004002	-17.507448
25% Quantile	-0.000009	-0.382852
50% Quantile (Median)	-0.000000	-0.007159
75% Quantile	0.000009	0.382308
Maximum	0.006823	29.830719
Skewness	0.9500864	0.9500864
Kurtosis	63.01904	63.01904
Jarque–Bera test statistic	4,780,366	4,780,366

Table 3.1: Descriptive statistics for non-standardized and standardized EUR/GBP 5-minute log returns. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

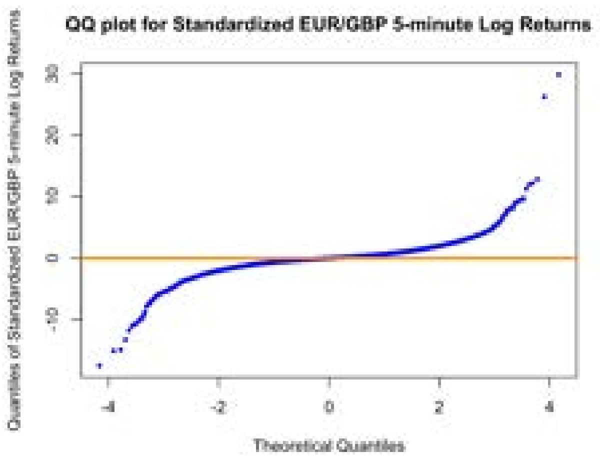
Prior Analysis of Standardized EUR/GBP 5-minute Log Returns

Before continuing with any additional analysis, it is crucial to investigate the standardized EUR/GBP 5-minute log return series and determine if a unit root is present. This step is necessary because when dealing with nonstationary time series, the usual assumptions for asymptotic analysis become invalid. The Dickey-Fuller test and augmented Dickey-Fuller test are among the most well-known unit root tests. The equation for the Dickey-Fuller test applied to the given time series $\{y_t\}$ is expressed as follows:

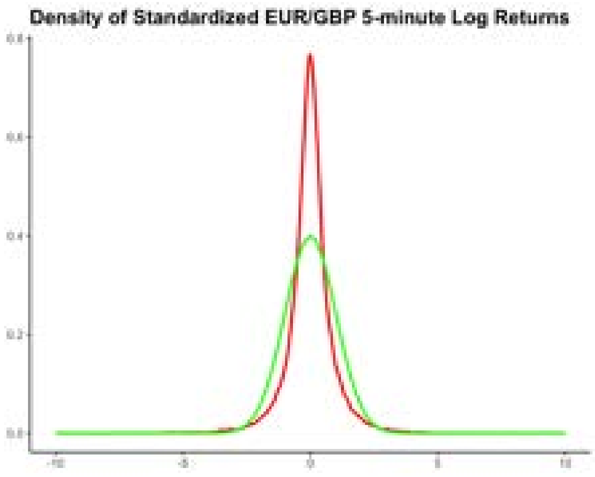
$$\Delta y_t = (\rho - 1)y_{t-1} + \epsilon_t, \tag{3.61}$$

where $\{\epsilon_t\}$ denotes a sequence of uncorrelated Gaussian random variables with a mean of zero and a constant and time-independent variance of σ^2 [15]. The null and alternative hypotheses of a Dickey-Fuller test are $\mathcal{H}_0 : \rho = 1$ (unit root is present, indicating that the time series $\{y_t\}$ is nonstationary), $\mathcal{H}_1 : \rho \neq 1$ (no unit root is present, suggesting that the time series $\{y_t\}$ is not characterized by random-walk stationarity). However, it is important to note that the Dickey-Fuller test is susceptible to autocorrelated residuals within test Equation 3.61. While the augmented Dickey-Fuller test is a modified version of the Dickey-Fuller test that addresses autocorrelated residuals, it can be sensitive to residual heteroskedasticity within the test equation. It is important to highlight that the current research deals with intraday returns, which are often expected to be heteroskedastic. This means that their volatility can change over time. For instance, the volatility of intraday returns often exhibits a parabolic shape. This phenomenon can be attributed mainly to a reduced trading activity that typically occurs in the middle of the day. Thus, it is important to accommodate this diurnal feature in the statistical analysis. The Phillips-Perron nonparametric unit root test accommodates autocorrelated and heteroscedastic residuals in the Dickey-Fuller test equation [38]. The Phillips-Perron test results in the rejection of the null hypothesis of a unit root presence in the standardized EUR/GBP 5-minute log returns at any reasonable significance level (refer to Appendix A.4.1). As a result, it can be concluded that the log return series is weakly stationary.

Furthermore, a data distribution analysis is conducted. One of the most popular visual tool for examining distributions is a QQ (Quantile-Quantile) plot. This graphical method compares empirical quantiles against quantiles of an assumed theoretical distribution. The constructed QQ plot presented in Figure 3.2 (a) reveals that there are significant systematic deviations of quantiles of standardized EUR/GBP 5-minute log returns from the quantiles of a normal distribution. It is evident that the distribution of sample standardized EUR/GBP 5-minute log returns is characterized by heavy tails, meaning it exhibits leptokurtic behavior. This observation is further supported by the significant kurtosis value presented in Table 3.1 (it's worth noting that the kurtosis of a normal distribution is equal to 3). The empirical findings can be summarized as indicating that the standardized EUR/GBP 5-minute log return distribution tends to exhibit characteristics of being non-Gaussian, sharply peaked and having heavy tails. Please refer to Figure 3.2 (b) for visual representation. These findings align with the well-established empirical characteristics of asset returns [11]. Consequently, the standardized EUR/GBP 5-minute log returns do not appear to be well-described by a normal distribution.



(a)



(b)

Figure 3.2: (a) The QQ-plot depicts the standardized EUR/GBP 5-minute log returns, with empirical quantiles plotted against the quantiles of a normal distribution. The orange line represents a 45-degree reference line. (b) Empirical and theoretical normal probability density for standardized EUR/GBP 5-minute log returns. The red line represents a kernel estimator of the empirical density, while the green line depicts the probability density for a simulated normally distributed variable. The parameters of this simulated variable, including mean and standard deviation, match the corresponding sample values of standardized EUR/GBP 5-minute log returns. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Let us examine the standardized EUR/GBP 5-minute log returns to determine if intraday seasonality is present. Upon visually analyzing Figure 3.3, it can be concluded that there is no pronounced trend in the standardized EUR/GBP mean 5-minute log return dynamics over 30 minute intervals during active trading hours. In simpler terms, there seems to be no observable intraday seasonality in the standardized EUR/GBP mean

5-minute log return dynamics over 30 minute intervals. The formal test for intraday seasonality will enhance the information gained from visual observations.

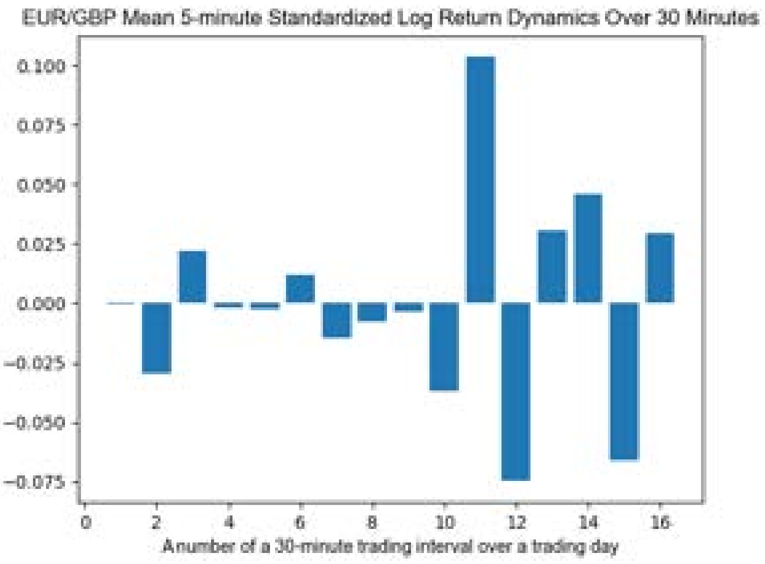


Figure 3.3: Standardized EUR/GBP mean 5-minute log return dynamics over 30 minute intervals during active trading hours. There are sixteen 30 minute intervals over active trading hours. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

In the study, we looked at the pure diurnal dummy variable model for standardized EUR/GBP 5-minute log return, as mentioned in Subsection 2.4.2. We used ordinary least squares to fit the model and examined the residuals for signs of heteroskedasticity. The findings are detailed in Appendix A.4.2, indicating that the residual heteroskedasticity was detected.

The chi-squared test used to assess the joint significance of the dummy variables in a Pure Diurnal Dummy Variable Model (as discussed in the Subsection 2.4.2) indicates that the assertion of no intraday seasonality in the standardized EUR/GBP 5-minute log returns is rejected at a 5% significance level. For more details, please refer to Appendix A.4.3.

In conclusion, the intraday seasonality present in standardized EUR/GBP 5-minute log returns should be duly considered in the subsequent analysis. More specifically, it is necessary to eliminate this intraday seasonality component denoted as S_t^r from the standardized EUR/GBP 5-minute log return series denoted as r_t before proceeding with model construction for the deseasonalized log return series denoted as $r_t^{deseasonalized}$ as follows (refer to Equation 2.48):

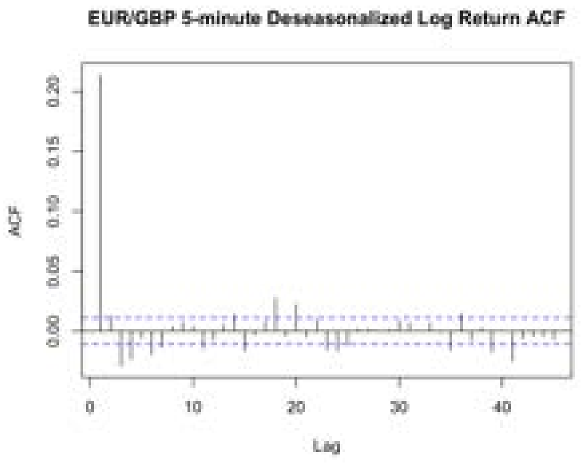
$$\begin{aligned} r_t &= \mu_t + \epsilon_t \\ r_t^{deseasonalized} &= \mu_t - S_t^r \\ \epsilon_t &\sim \text{i.i.d.}(0, \sigma^2). \end{aligned} \tag{3.62}$$

Standardized EUR/GBP 5-minute Log Return Model Building

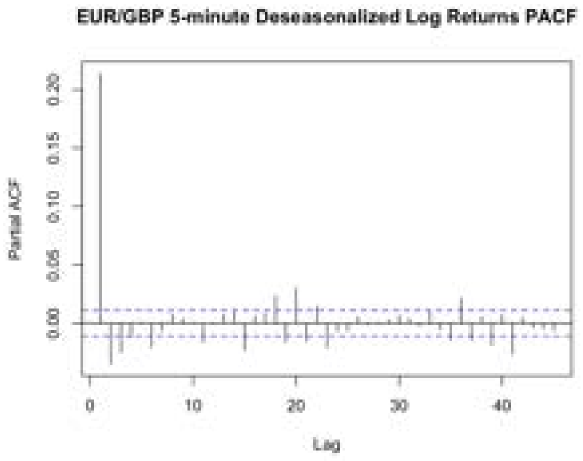
A stationary time series model ARMA(p, q) model will be used to capture the behavior of standardized EUR/GBP 5-minute log return series r_t with removed intraday seasonality component $r_t^{deseasonalized}$ (refer to Equation 3.62).

The primary tools for model identification will involve using the sample Autocorrelation Function (ACF) in combination with the sample Partial Autocorrelation Function (PACF).

The sample ACF and sample PACF are graphed in the Figure 3.4. Both the sample ACF and PACF roughly show a damped sinusoidal pattern, and approximately beyond the lag 3, they fall within the confidence interval. For lags beyond 3, certain autocorrelation function values extend beyond the confidence interval. Nonetheless, these will be regarded as data anomalies.



(a)



(b)

Figure 3.4: (a) Autocorrelation function (ACF) for deseasonalized EUR/GBP 5-minute log returns. (b) Partial autocorrelation function (PACF) for deseasonalized EUR/GBP 5-minute log returns. The 95% confidence interval boundaries are indicated by blue dotted lines. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

To prevent overfitting, the goodness of fit for the following 15 ARMA models is going to be assesed based on the information criteria AIC and BIC: ARMA(0,1), ARMA(0,2), ARMA(0,3), ARMA(1,0), ARMA(2,0), ARMA(3,0), ARMA(1,1), ARMA(1,2), ARMA(1,3), ARMA(2,1), ARMA(2,2), ARMA(2,3), ARMA(3,1), ARMA(3,2), ARMA(3,3). Since the EUR/GBP 5-minute log returns have been standardized, the models are configured to exclude the mean term.

Both information criteria considered agree that the models ARMA(2,1) and ARMA(3,1) should be chosen (refer to Figure 3.5). Since the principle of preferring simple models to complicated ones is followed, ARMA(2,1) with no mean term is chosen.

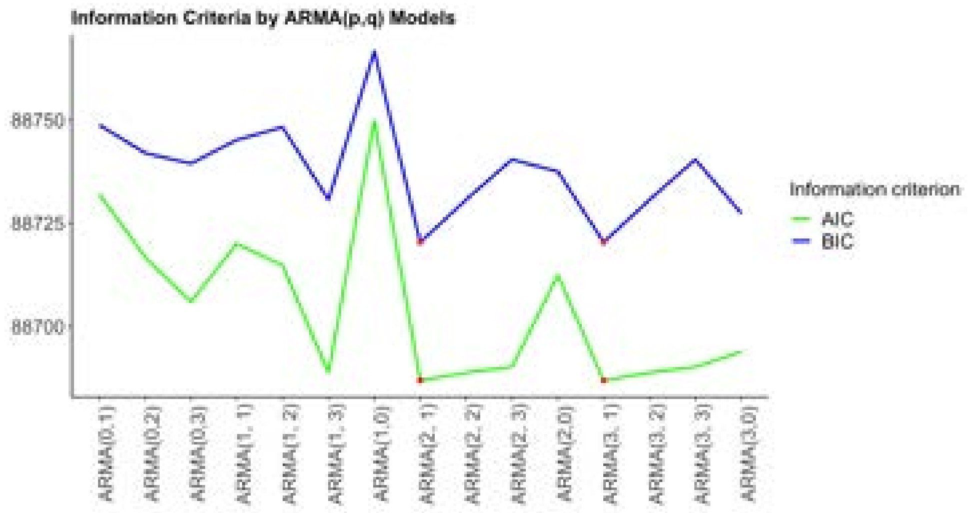


Figure 3.5: Information criteria (AIC in green and BIC in blue) by ARMA(p, q) model. The red dots on the graph represent the points of lowest AIC and BIC values. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Estimation of the ARMA(2,1) model results in the following parameters (coefficient
1295 standard errors are in parentheses below the coefficient estimates):

$$\begin{aligned} r_t^{deseasonalized} &= \underset{(0.0731)}{0.8469} r_{t-1}^{deseasonalized} - \underset{(0.0141)}{0.1695} r_{t-2}^{deseasonalized} - \underset{(0.0737)}{0.6266} \epsilon_{t-1} + \epsilon_t, \\ \epsilon_t &\sim \text{i.i.d.}(0, \sigma^2). \end{aligned} \tag{3.63}$$

It is worth noting that the invertibility condition for the moving average component is met because the absolute value of the coefficient in the moving average component is less than 1, specifically, $|-0.6266| < 1$. Regarding the fitted residuals for Equation 3.63, you can find diagnostic information in Appendix A.4.4.

Standardized EUR/GBP 5-minute Conditional Volatility Model Building

Sample intraday conditional volatility will be estimated using the ARMA(2,1)-GARCH(1,1) model framework [12] (see Subsections 2.4.4 and 2.4.5):

$$
\begin{aligned}
r_t &= \mu_t + \epsilon_t, \\
\widehat{\mu_t} &= (0.8469 r_{t-1}^{deseasonalized} - 0.1695 r_{t-2}^{deseasonalized} - 0.6266 \epsilon_{t-1}) + \widehat{S_t^r} = \widehat{r_t^{deseasonalized}} + \widehat{S_t^r}, \\
\widehat{S_t^r} &= \sum_{i=1}^{16} \beta_i D_{i,t}, \\
\epsilon_t &= S_t^{\epsilon} \sigma_t^* \nu_t = S_t^{\epsilon} \epsilon_t^*, \nu_t \sim \text{i.i.d.}(0,1), \\
\ln(\widehat{\epsilon}_t^2) &= \widehat{\ln(S_t^{\epsilon 2})} + \widehat{u_t}, \\
\widehat{\ln(S_t^{\epsilon 2})} &= \sum_{i=1}^{16} \beta_i D_{i,t}, \\
\widehat{S_t^{\epsilon 2}} &\approx \exp\{\widehat{\ln(S_t^{\epsilon 2})}\} \frac{\sum_{t=1}^{T} \exp\{\widehat{u_t}\}}{T}, \\
\widehat{\epsilon_t^*} &= \frac{\widehat{\epsilon_t}}{\sqrt{\widehat{S_t^{\epsilon 2}}}} = \frac{r_t - \widehat{\mu_t}}{\sqrt{\widehat{S_t^{\epsilon 2}}}}, \\
\epsilon_t^* &= \gamma + \varepsilon_t^* \\
\varepsilon_t^* &\sim \text{i.i.d.}(0, \sigma_t^{*2}) \\
\sigma_t^{*2} &= \omega + \alpha_1 \varepsilon_{t-1}^* + \beta_1 \sigma_{t-1}^{*2},
\end{aligned}
\tag{3.64}
$$

where

- r_t represents a standardized EUR/GBP 5-minute log return, while $r_t^{\text{deseasonalized}}$ signifies a deseasonalized EUR/GBP 5-minute log return, and the fitted deseasonalized EUR/GBP 5-minute log return is denoted as $\widehat{r_t^{\text{deseasonalized}}}$;

- μ_t denotes the conditional expectation of the standardized EUR/GBP 5-minute log return (see Subsection 3.1.6 and the Equation 3.63 in particular);

- ϵ_t is a deseasonalized EUR/GBP 5-minute log return series residual, which is further decomposed into a deterministic seasonal volatility component S_t^{ϵ}, a deseasonalized GARCH component σ_t^*, and an independent and identically distributed random variable ν_t with zero mean and unit variance (refer to Appendix A.4.5);

- S_t^r captures the intraday seasonality in the standardized EUR/GBP 5-minute log return, modeled through a pure diurnal dummy variable (for the estimated model, please refer to Appendix A.4.3);
- ε_t^* denotes the deseasonalized residual ϵ_t^* after removing its mean represented as γ;
- all hatted variables indicate estimates of the true underlying variables;
- T represents the sample size of a training subset, with $T = 31,817$.

The `fGarch` package from `R` is used to estimate the daily conditional volatility σ_t, its deterministic seasonal component S_t^ϵ and its stochastic GARCH component σ_t^*. A comprehensive explanation of the true and deseasonalized volatility estimation and forecasting, along with the forecast optimality test, is provided in Appendices A.4.5 and A.4.6. For the relevant `R` code please consult Appendix A.5.

Find the visualizations depicting the estimation and forecasting of true and deseasonalized conditional volatility of EUR/GBP 5-minute deseasonalized log returns in Figures 3.6 and 3.7. Please note that in Figure 3.6, the long-run average volatility is also represented by a horizontal red line. We can observe periods of high volatility (exceeding the long-run average volatility) and periods of low volatility (falling below the long-run average volatility). This phenomenon is commonly referred to as volatility clustering and is a key characteristic observed in the volatility of returns for the majority of financial instruments.

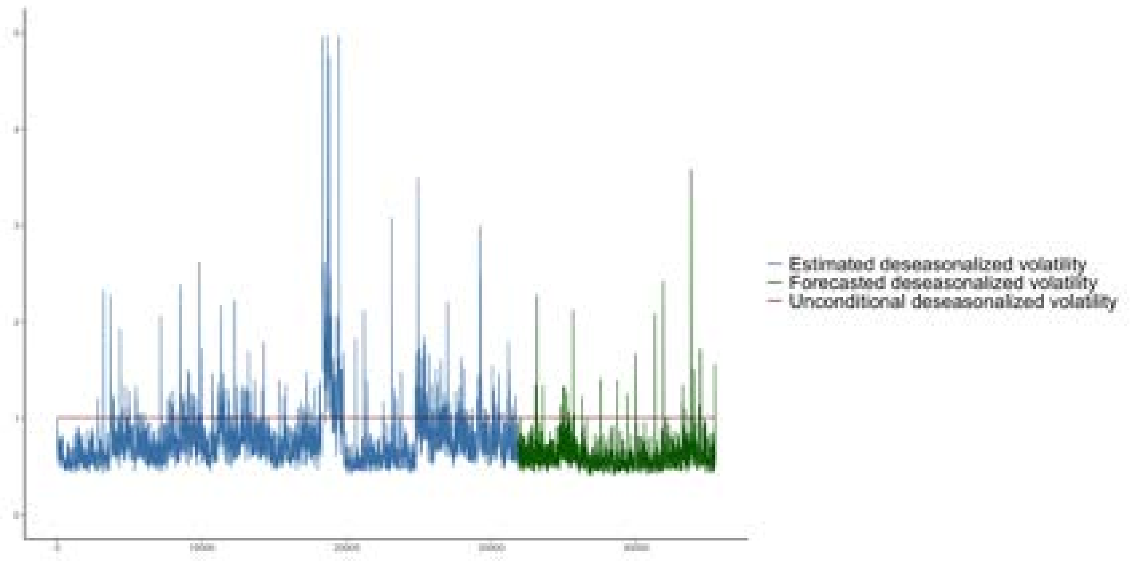


Figure 3.6: The plot displays the estimated (in blue) and forecasted (in green) deseasonalized conditional volatility for EUR/GBP 5-minute deseasonalized log returns. This conditional volatility estimation has been achieved using an ARMA(0,0)-GARCH(1,1) model. The horizontal red line represents the unconditional (long-run) volatility calculated based on the coefficient estimates of the GARCH(1,1) model. The plot is limited to visualize the conditional volatility within the range of 0 to 5 for the sake of clarity. The dataset covers the time period from January 2022 to January 2023 and has been sourced from Bloomberg.

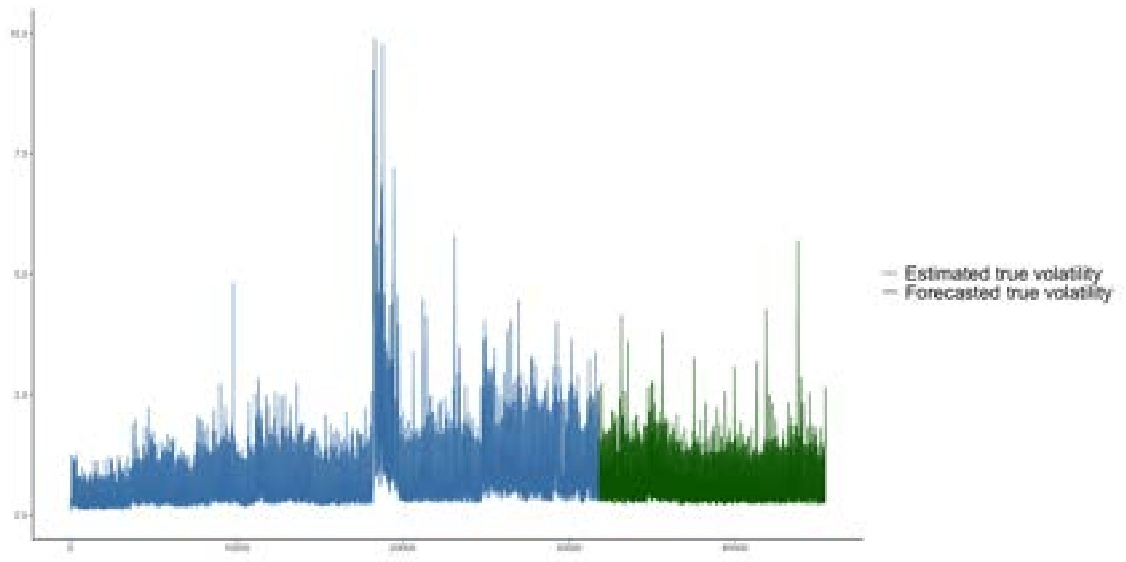


Figure 3.7: The plot displays the estimated (in blue) and forecasted (in green) true conditional volatility for EUR/GBP 5-minute deseasonalized log returns. This conditional volatility estimation has been achieved using an ARMA(0,0)-GARCH(1,1) model. The plot is limited to visualize the conditional volatility within the range of 0 to 10 for the sake of clarity. The dataset covers the time period from January 2022 to January 2023 and has been sourced from Bloomberg.

Please note that the estimated and forecasted conditional volatilities of EUR/GBP 5-minute deseasonalized log returns will be utilized as inputs in a DRL trading system. This is discussed in the next subsection.

3.2 Architecture of the Models

Below, you will find a comprehensive explanation of the DRL trading system for your convenience. The two similar models are considered. The main difference between Model 1 (refer to Figure 3.8) and Model 2 (refer to Figure 3.9) is whether the input is transformed using a shallow neural network. Both DRL trading systems are designed to trade a single risky asset labeled as j, using a two-position action (long/short). The design is backed by relevant research (refer to Subsection 2.3). In the current research, the single risky asset j is EUR/GBP exchange rate.

Model 1

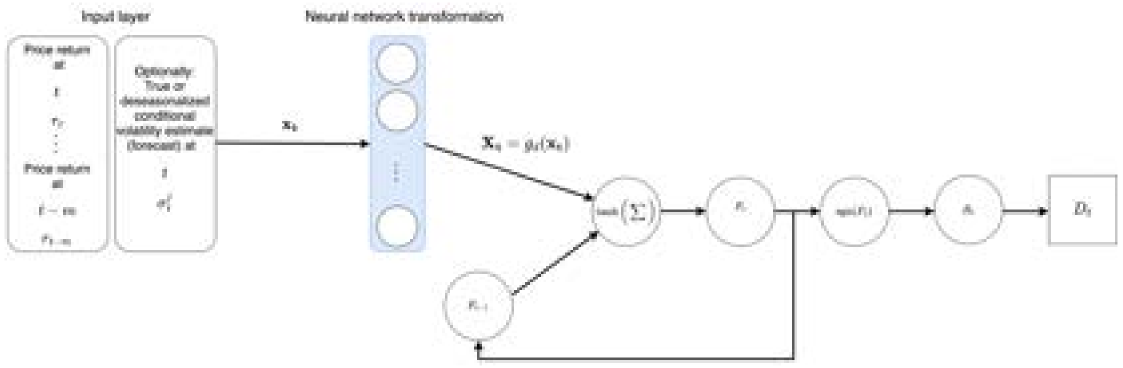

Input layer
Price return at
t
r_t
Price return at
$t-m$
r_{t-m}
Optionally: True or deseasonalized conditional volatility estimate (forecast) at
t
σ_t^2
Neural network transformation
$\mathbf{x}_t$
$\mathbf{X}_t = g_\ell(\mathbf{x}_t)$
$\tanh(\sum)$
F_t
$\mathrm{sgn}(F_t)$
D_t
F_{t-1}

Figure 3.8: Model 1 architecture. A shallow neural network for feature engineering (in blue) [14].

Note that the input layer of Model 1 (refer to the left part of Figure 3.8) accommodates three distinct input configurations:

1) standardized lagged returns,

2) standardized lagged returns combined with a true conditional volatility estimate,

3) standardized lagged returns combined with a deseasonalized conditional volatility estimate.

Model 1 is going to be estimated separately for each input configuration.

Similar to the reference [14], a feature learning component is integrated into the DRL trading system. The feature learning part is represented by a shallow neural network. The neural network representation of an input vector denoted as $g_d(\cdot)$ (see blue panel in the Figure 3.8) is structured as follows. The fully connected shallow neural network has a single layer with 10 output nodes. The activation function used is the leaky Rectified Linear Unit (ReLU) function with a constant of 0.01. For each node $i \in [1, 2, \ldots, 10]$ in the output layer

$$\begin{aligned} y_i &= \mathbf{w_i} \cdot \mathbf{x_t} + b_i, \\ a_i &= \max(0.01 y_i, y_i), \end{aligned} \tag{3.65}$$

where $\mathbf{x_t}$ refers to the vector of values from the input layer, $(\mathbf{w_i}, b_i)$ $\forall i$ are the latent variables that the shallow neural network aims to learn.

A neural network transformation produces a high-level feature representation of the present market condition (see blue panel in Figure 3.8) [14]. The trading signal at time period t denoted as F_t is a function of a neural network representation of the input features at time period t denoted as $\mathbf{X_t}$ (comprised by the a_i values for $\forall i$ from the second equality in Equation 3.65):

$$\begin{aligned} F_t &= \tanh(\mathbf{v} \cdot \mathbf{X_t} + u F_{t-1} + w), \\ \mathbf{X_t} &= g_d(\mathbf{x_t}), \end{aligned}$$

where $\mathbf{x_t} = \langle r_t, r_{t-1}, \ldots, r_{t-m} \rangle$ is an input vector that may optionally include the conditional volatility forecast at time step t, m is a length of the return lag, $g_d(\cdot)$ is a neural network transformation defined in Equation 3.65, and $(\mathbf{v}, u, w)$ are the coefficients that correspond to the feature regression.

Next, the realized rate of return at time period t, represented as R_t (refer to Subsection 2.3.4), is computed as follows:

$$R_t = \beta \{ \operatorname{sgn}(F_{t-1}) r_t - |\operatorname{sgn}(F_t) - \operatorname{sgn}(F_{t-1})| \delta_t \},$$

where $\beta = 1$, as in the references [14, 52], $\operatorname{sgn}(F_t)$ is a trading position at time step t, r_t

refers to a EUR/GBP 5-minute standardized log return (see Subsections 3.1.2 and 3.1.4), and δ_t is a transaction cost at time step t (see Subsection 3.1.3).

The DRL algorithm's objective function, which it seeks to maximize, is the DSR, see Subsection 2.3.7. DSR is defined as follows:

$$D_t = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}},$$

where $A_t = A_{t-1} + \eta \Delta A_t$, $\Delta A_t = R_t - A_{t-1}$, $B_t = B_{t-1} + \eta \Delta B_t$, $\Delta B_t = R_t^2 - B_{t-1}$.

Bringing all components together, the optimization problem for Model 1 is expressed as follows:

$$\begin{aligned} \max_{\{\boldsymbol{\theta}, g_d(\cdot)\}} D_t &= \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}, \ \forall t \in [1, 2, \dots, T] \\ \text{s.t. } R_t &= \operatorname{sgn}(F_{t-1})r_t - |\operatorname{sgn}(F_t) - \operatorname{sgn}(F_{t-1})|\delta_t \\ F_t &= \tanh(\mathbf{v} \cdot \mathbf{X_t} + uF_{t-1} + w) \\ \mathbf{X_t} &= g_d(\mathbf{x_t}). \end{aligned} \tag{3.66}$$

The trading system parameters that are subject to learning are indicated as $\boldsymbol{\theta} = (\mathbf{v}, u, w)$. In addition, the neural network representation $g_d(\cdot)$ is to be learned.

Model 2

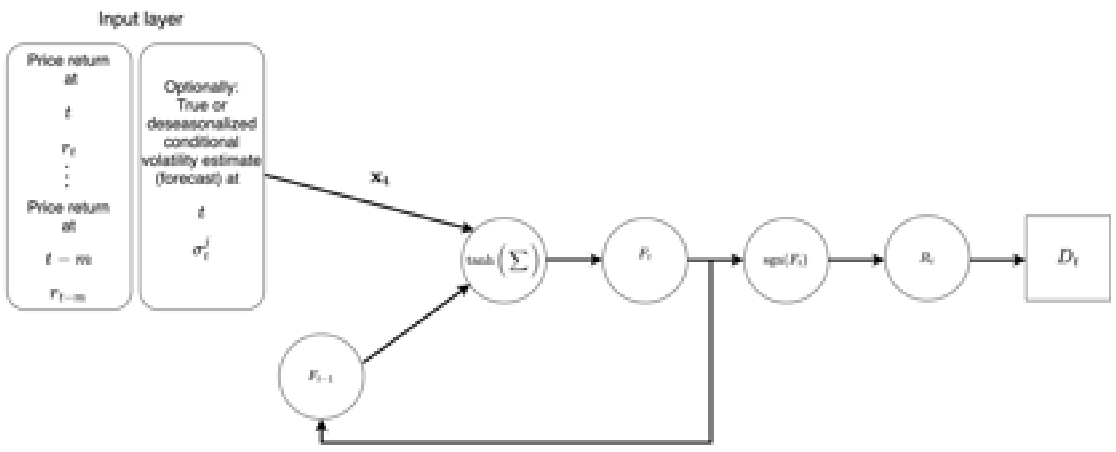

Input layer
Price return at t r_t
Price return at $t-m$ r_{t-m}
Optionally: True or deseasonalized conditional volatility estimate (forecast) at t σ_t^j
$\mathbf{x}_t$
$\tanh\left(\sum\right)$
F_t
$\operatorname{sgn}(F_t)$
R_t
D_t
F_{t-1}

Figure 3.9: Model 2 architecture.

Note that, similar to Model 1, the input layer of Model 2 (refer to the left part of Figure 3.9) accommodates three distinct input configurations:

1) standardized lagged returns,
2) standardized lagged returns combined with a true conditional volatility estimate,
3) standardized lagged returns combined with a deseasonalized conditional volatility estimate.

Similarly, Model 2 is going to be estimated separately for each input configuration.

As noted previously, Model 2 (refer to the Figure 3.9) is highly similar to Model 1, with the key distinction being the absence of a neural network transformation for the input vector. In Model 2, the input vector is directly supplied to the DRL trading system. Similar to Model 1, the optimization problem for Model 2 is expressed as follows:

$$\begin{aligned}\max_{\boldsymbol{\theta}} D_t &= \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}, \ \forall t \in [1, 2, \ldots, T] \\ \text{s.t. } R_t &= \operatorname{sgn}(F_{t-1})r_t - |\operatorname{sgn}(F_t) - \operatorname{sgn}(F_{t-1})|\delta_t \\ F_t &= \tanh(\mathbf{v} \cdot \mathbf{x_t} + uF_{t-1} + w),\end{aligned} \tag{3.67}$$

for the notations, refer to Subsection 3.2.1. Similar to the Model 1, the trading system parameters that are subject to learning are indicated as $\boldsymbol{\theta} = (\mathbf{v}, u, w)$.

3.3 Models 1 and 2 Optimization

The trading system parameters denoted as $\boldsymbol{\theta}$ are adjusted through an on-line learning (see the Subsection 2.3.6) using a stochastic gradient ascent. The gradient of D_t with respect to the parameter set $\boldsymbol{\theta_t}$ learned by the Models 1 and 2 at time period t (refer to Subsections 2.3.7 and 2.3.8) is as follows:

$$\begin{aligned}
\frac{\mathrm{d}D_t(\boldsymbol{\theta_t})}{\mathrm{d}\boldsymbol{\theta_t}} &= \frac{\mathrm{d}D_t}{\mathrm{d}R_t}\left\{\frac{\mathrm{d}R_t}{\mathrm{d}F_t}\frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_{t-1}}}\right\}, \text{ where} \\
\frac{\mathrm{d}D_t}{\mathrm{d}R_t} &= \frac{B_{t-1} - A_{t-1}R_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}, \\
\frac{\mathrm{d}R_t}{\mathrm{d}F_t} &= -\mathrm{sgn}(F_t - F_{t-1})\delta_t, \\
\frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} &= r_t + \mathrm{sgn}(F_t - F_{t-1})\delta_t, \\
\frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} &= \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}F_t}{\mathrm{d}F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_{t-1}}}.
\end{aligned} \tag{3.68}$$

At time step t, the signal parameter set $\boldsymbol{\theta_t}$ is updated in an on-line manner by employing the following rule

$$\boldsymbol{\theta_t} = \boldsymbol{\theta_{t-1}} + \rho\Delta\boldsymbol{\theta_t}, \tag{3.69}$$

where ρ represents a learning rate, and $\Delta\boldsymbol{\theta_t} = \dfrac{\mathrm{d}D_t(\boldsymbol{\theta_t})}{\mathrm{d}\boldsymbol{\theta_t}}$ is the gradient of the DSR with respect to the parameter set $\boldsymbol{\theta_t}$ (refer to Subsection 2.3.8).

Note that the computations $\dfrac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} = \dfrac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \dfrac{\mathrm{d}F_t}{\mathrm{d}F_{t-1}}\dfrac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_{t-1}}}$ evolve recursively [14, 36]. This gives rise for the need of a recurrent optimization algorithm, for instance, Backpropagation Through Time (BPTT) [14, 36, 51].

In essence, BPTT involves unfolding a recurrent neural network over a specified fixed number of time steps, effectively transforming it into a straightforward feedforward neural network. A segment length for unfolding was selected as two, which is consistent with the approach used in the reference [14].

The training algorithm iterates through the observations in the training dataset, advancing by two steps at a time. During each step, it calculates the gradient of the recurrent neural network output, denoted as F_t, with respect to the recurrent neural network parameters as $\dfrac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta}_t}$. These gradients are then employed to update the parameters using the formula specified in the last equality of Equation 3.68.

Subsequently, following the second time step, the trading system's parameters are adjusted using the rule defined in Equation 3.69.

Additionally, it is important to mention that the output of the recurrent network (current trading position F_t) is continuous and ranges from -1 to 1. During training, the continuous value of the trading position is converted into a binary one when calculating the realized trading return, which is then further utilized in the DSR calculation. However, during the testing phase, the output of the neural network (current trading position F_t) is binary and takes values of -1 or 1. The *sign* function for the output of the neural network is only used in the testing phase because the *sign* function is not differentiable at every point. Therefore, it may introduce complexities for the gradient flow and efficient model training.

3.4 Models 1 and 2 Parameters

Network parameter	**Comment**
Number of inputs	Number of inputs is set to 50 similar to the reference [14]. Various experiments have been performed wherein the number of inputs was set to 10, 30, and 50; however, the model seemed to perform best with 50 price inputs. Fifty 5-minute inputs mean that the model requires price information from approximately the past 4 hours.
Number of layers	The number of layers was initially set to 4, following the approach described in the reference [14]. However, this choice introduced significant training complexity and associated issues, such as the vanishing gradient problem, especially under certain trading system parameter initializations (refer to Subsection 3.4.1). Therefore, a decision was made to transition to a simpler neural network architecture with only a single hidden layer. The sigmoid activation function from the reference [14] was not retained; instead, it was replaced with a leaky ReLU function with a constant of 0.01. Furthermore, given the focus of the current research on high-frequency financial data, opting for a relatively shallow neural network was considered reasonable for the sake of training efficiency, as suggested in the reference [14].
Number of neurons	The number of neurons in the hidden layer is 10, as is the number of neurons in the output layer.

Table 3.2: Network parameters.

Training parameter	**Comment**
Learning rate ρ	Learning rate ρ is set to 0.01.
Number of epochs for training	Due to computational restrictions, the training was conducted for only a single epoch.

Table 3.3: Training parameters.

Training parameter	**Comment**
A_0, B_0, F_0	These constants are initialized using a standard normal distribution.
Time constant η	The time constant η is configured to be 0.01, which is consistent with the relevant prior research [36].

Table 3.4: Trading system parameters.

Model Parameter Initialization

To begin, it is crucial to differentiate between the initialization of neural network parameters, denoted as $\boldsymbol{\theta}_\mathbf{t}$ and detailed in Table 3.2, and trading system parameters, denoted as A_0, B_0, F_0, and η, as outlined in Table 3.4.

In the reference [52], for daily equity trading, the signal parameters are initialized using random numbers following a Gaussian distribution with a mean of 0 and a standard deviation of 0.05. In the current research, I attempted to perform a similar initialization approach for the neural network parameters. However, this led to a gradient vanishing problem, similar to what was observed in the reference [14].

The gradient vanishing issue arises as gradients becoming extremely close to zero. This phenomenon typically occurs when the trading system employs sigmoid and tangent activation functions, and the inputs to these functions become exceptionally large or small, causing the functions to become nearly flat in those regions. Consequently, this results in gradients close to zero, effectively halting the gradient flow and impeding efficient model training. Moreover, the presence of a recurrent structure in the trading system may exacerbate the gradient vanishing issue.

After conducting several experiments with different neural network parameter initializations, I found that using Kaiming He initialization for the shallow neural network portion allowed to avoid the vanishing gradient issue. Specifically, a combination of this with a leaky ReLU activation and Xavier initialization for the recurrent part, utilizing a tangent activation.

The description of the Kaiming He parameter initialization used in the present study is as follows:

$$\begin{gathered} W^{[l]} \sim \mathcal{N}\left(0, \frac{2}{n^{[l]}}\right), \\ b^{[l]} = 0. \end{gathered} \tag{3.70}$$

In Equation 3.70, $W^{[l]}$ represents the weight matrix for layer l and $n^{[l]}$ represents the number of nodes in layer l and $b^{[l]}$ stands for a bias term associated with layer l. The Kaiming He parameter initialization has been specifically designed for non-linear leaky ReLU activation function [23].

The Xavier parameter initialization applied in the current study is described as follows:

$$\begin{gathered} W^{[l]} \sim \mathcal{N}\left(0, \frac{2}{n^{[l-1]} + n^{[l]}}\right), \\ b^{[l]} = 1. \end{gathered} \tag{3.71}$$

In Equation 3.71, $W^{[l]}$ represents the weight matrix for layer l, $n^{[l-1]}$ denotes the number of nodes in layer $l-1$, $n^{[l]}$ represents the number of nodes in layer l, and $b^{[l]}$ stands for a bias term associated with layer l. Xavier parameter initialization is often used for neural networks with *tanh* activation functions [25]. Note that Xavier parameter initialization is valid only for linear activation functions [23].

Models 1 and 2 Implementation in Python

The models are implemented in Python programming language using the PyTorch framework. Since the objective function was custom-designed, automatic optimization capabilities provided by PyTorch were not suitable. Consequently, a meticulous manual optimization process was undertaken. Please find the code in the Appendix A.6. Note that the code is provided for the implementation of Model 1 with lagged returns as an input. Implementations of the models with different input configurations are fairly similar.

3.5 Models 1 and 2 Training Dynamics

As you can observe, the training dynamics for Model 1 (refer to Figure 3.10) and Model 2 (refer to Figure 3.11) are very similar. This similarity persists even when different input configurations are used, so the training dynamics of models with varying input configurations are not presented here. Notably, the realized trading returns are predominantly positive throughout the training data. The Sharpe ratio exhibits a consistent upward trend and remains positive for nearly all values in the training sample. Moreover, an approximately equal proportion of buy and sell signals are generated.

Take note of a notable decline in both the DSR and SR, as seen in the top panels of Figures 3.10 and 3.11, occurring around observation number 18,000. This coincides with a significant appreciation of the euro against the British pound.

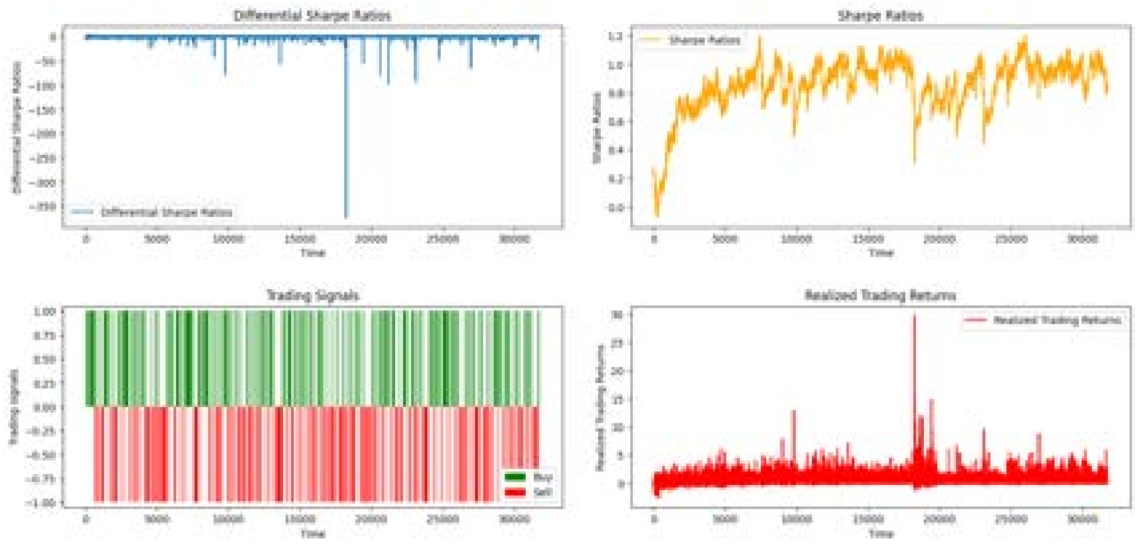


Figure 3.10: Training dynamic of Model 1 with a shallow neural network for feature learning which exclusively employs lagged returns as input. In the top left panel, we see the differential Sharpe ratio, the main training objective. The bottom left panel displays trading signals (green for "buy and red for "sell). In the top right panel, there is an upward trend in the Sharpe ratio. The bottom right panel shows realized trading returns. This analysis is based on a 5-minute EUR/GBP foreign exchange rate dataset, spanning January 2022 to January 2023, with 31,817 observations sourced from Bloomberg.

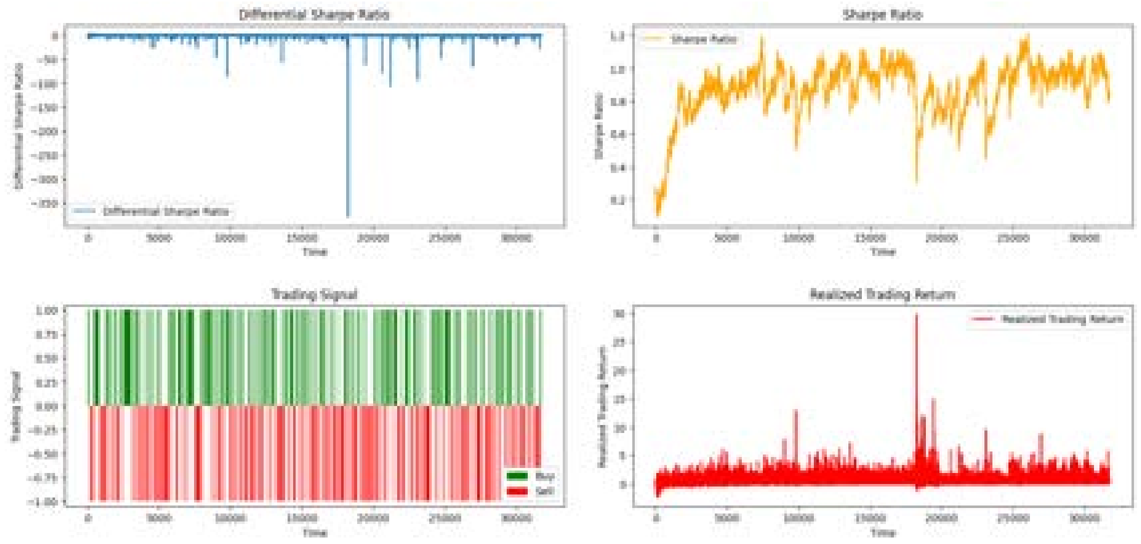


Figure 3.11: Training dynamic of Model 2 which exclusively employs lagged returns as input. The top left panel shows the dynamic of the differential Sharpe ratio, our primary training objective. In the bottom left panel, you will find the trading signals (green for "buy" and red for "sell"). The top right panel displays an upward trend in the Sharpe ratio, also a training goal. Lastly, the bottom right panel presents realized trading returns. This analysis is based on a 5-minute EUR/GBP foreign exchange dataset, with 31,817 observations spanning January 2022 to January 2023, sourced from Bloomberg.

3.6 Trading Performance in Evaluation

Similar to the reference [52], a random trading strategy and a conservative buy-and-hold strategy are employed to serve as a benchmark for evaluating the out-of-sample trading performance of the introduced trading system.

Trading Performance Comparison of Model 1 with Various Input Configurations

Based on the findings depicted in the top panel of the Figure 3.12, it can be concluded that the trading system designed using Model 1 exhibits superior performance compared to random trading and the buy-and-hold strategy. This is evident when considering a cumulative Sharpe ratio. The cumulative Sharpe ratio is calculated as the cumulative sum of Sharpe ratios.

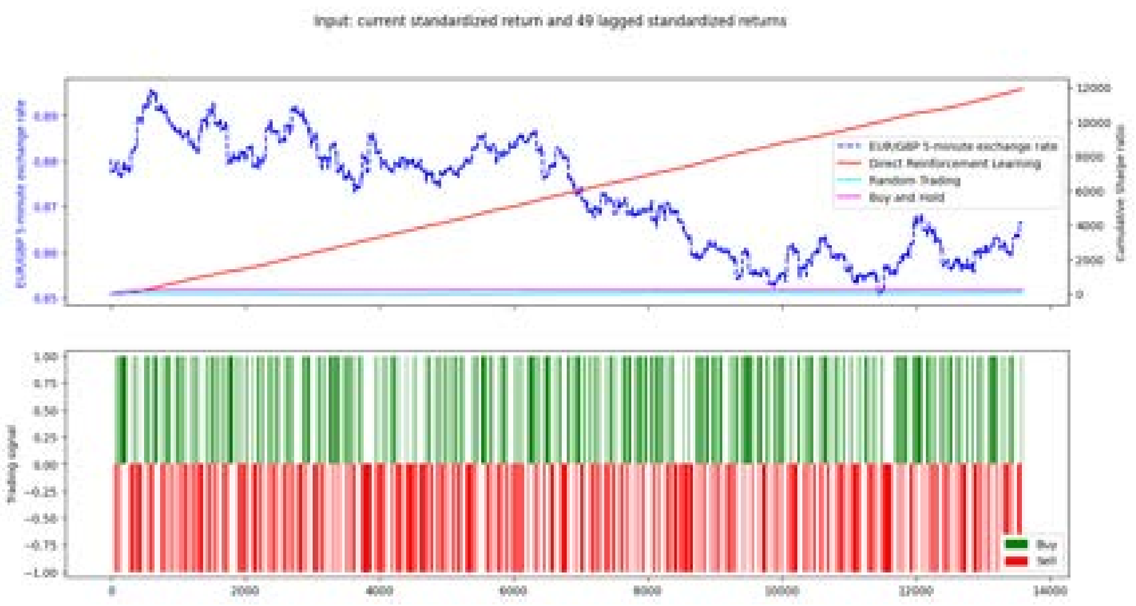
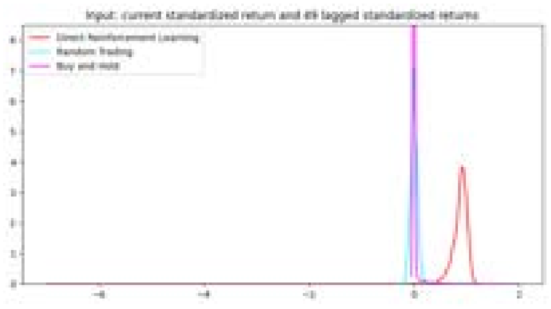


Figure 3.12: The out-of-sample dynamics of Model 1 which exclusively utilizes lagged returns as input. The top panel illustrates the dynamics of the 5-minute EUR/GBP exchange rate and the cumulative Sharpe ratio for Model 1, random trading, and a buy-and-hold strategy. The bottom panel depicts the generated trading signals. This analysis is conducted using a 5-minute EUR/GBP foreign exchange dataset, comprising 13,636 observations spanning from January 2023 to August 2023, sourced from Bloomberg.

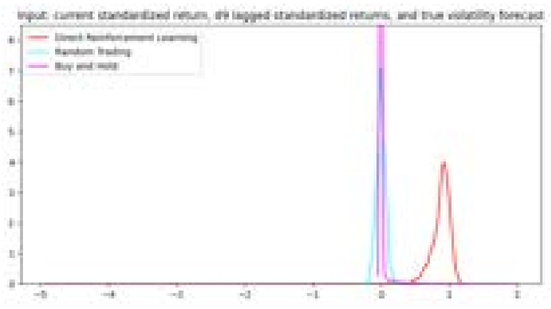
Figure 3.13 displays kernel density estimations of Sharpe ratios generated by Model 1 with various input configurations, as well as the kernel density estimation of Sharpe ratios produced by random trading and a buy-and-hold strategy. It is important to note that the kernel density estimation is a non-parametric method used to estimate the probability density function of a random variable.

Note that the means of the Sharpe ratios generated by random trading and the buy-and-hold strategy tend to fluctuate around zero, whereas the means of the Sharpe ratios produced by Model 1 with various input configurations are consistently greater than zero, fluctuating around one. Additionally, the variation in Sharpe ratios produced by Model 1 with various input configurations is higher compared to the variation in Sharpe ratios produced by other strategies. Furthermore, note that the distribution of the Sharpe ratios generated by Model 1 with lagged returns and Model 1 with true conditional volatility forecast is skewed to the left.

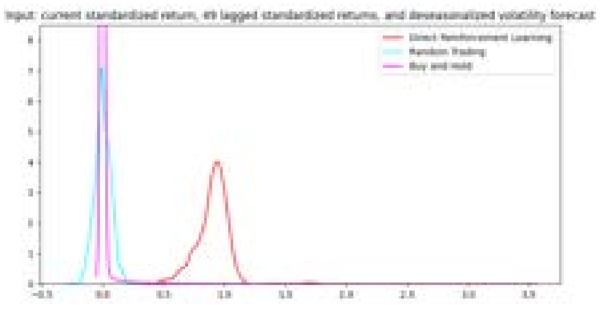
Therefore, one can conclude that Model 1 with various input configurations generates the best trading performance.



(a)



(b)



(c)

Figure 3.13: (a) Kernel density estimation of out-of-sample Sharpe ratios generated by Model 1 using lagged returns as input. (b) Kernel density estimation of out-of-sample Sharpe ratios generated by Model 1 using lagged returns and one-step-ahead true conditional volatility forecast as input. (c) Kernel density estimation of out-of-sample Sharpe ratios generated by Model 1 using lagged returns and one-step-ahead deseasonalized conditional volatility forecast as input. Note that the Sharpe ratio density produced by Model 2 is compared against the Sharpe ratio densities generated by random trading and a buy-and-hold strategy. This analysis is conducted using a 5-minute EUR/GBP foreign exchange dataset, comprising 13,636 observations spanning from January 2023 to August 2023, sourced from Bloomberg.

Trading Performance Comparison of Model 2 with Various Input Configurations

The trading system defined by Model 2 consistently outperforms random trading and conservative buy-and-hold strategy in terms of a cumulative Sharpe ratio, as seen in the top panel of the Figure 3.14.

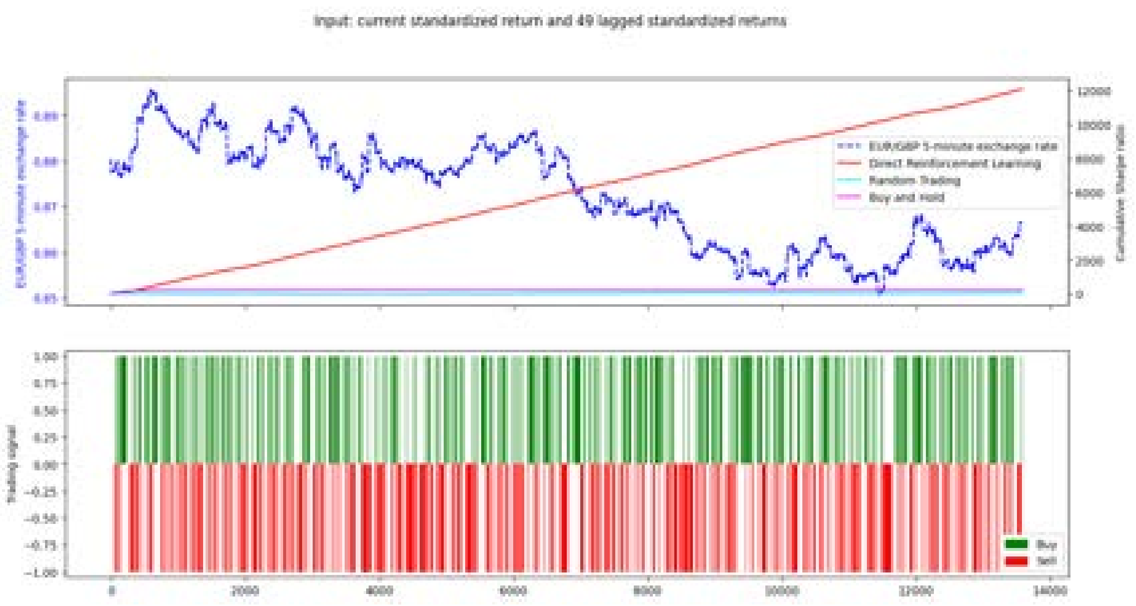
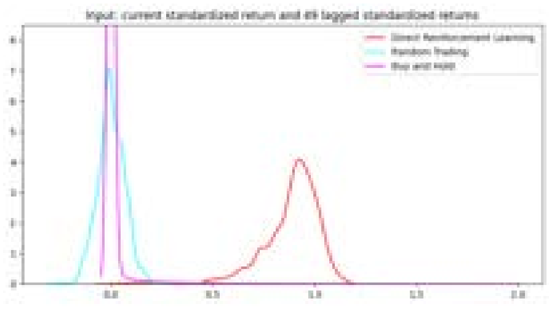


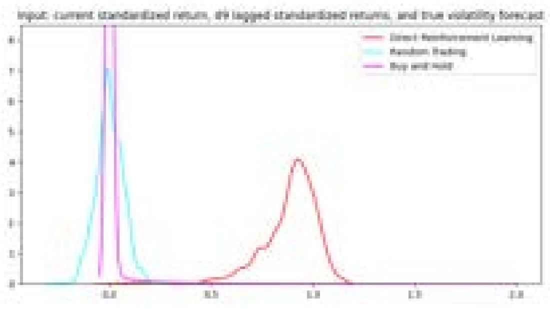
Figure 3.14: The out-of-sample dynamics of Model 2 which exclusively utilizes lagged returns as input. The top panel illustrates the dynamics of the 5-minute EUR/GBP exchange rate and the cumulative Sharpe ratio for Model 2, random trading, and a buy-and-hold strategy. The bottom panel depicts the generated trading signals. This analysis is conducted using a 5-minute EUR/GBP foreign exchange dataset, comprising 13,636 observations spanning from January 2023 to August 2023, sourced from Bloomberg.

The Figure 3.15 displays kernel density estimations of Sharpe ratios generated by Model 2 with various input configurations, as well as the kernel density estimation of Sharpe ratios produced by random trading and a buy-and-hold strategy.

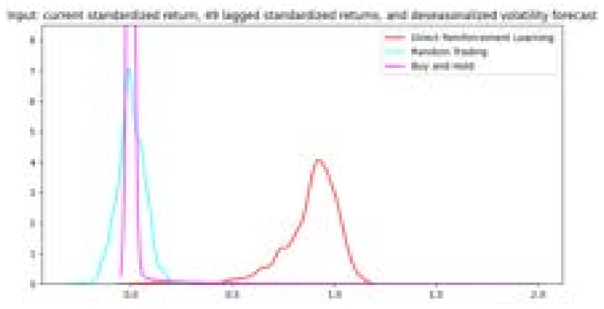
Similar to Model 1, Model 2 with various input configurations also appears to offer the best risk-adjusted performance, as its peak is prominently shifted toward the positive side and consistently fluctuates around one.



(a)



(b)



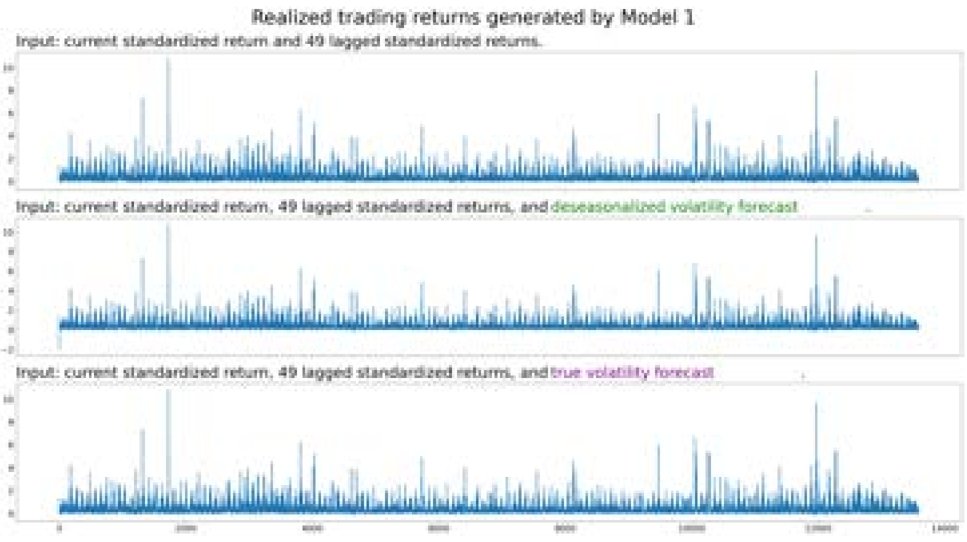
(c)

Figure 3.15: (a) Kernel density estimation of out-of-sample Sharpe ratios generated by Model 2 using lagged returns as input. (b) Kernel density estimation of out-of-sample Sharpe ratios generated by Model 2 using lagged returns and one-step-ahead true conditional volatility forecast as input. (c) Kernel density estimation of out-of-sample Sharpe ratios generated by Model 2 using lagged returns and one-step-ahead deseasonalized conditional volatility forecast as input. Note that the Sharpe ratio density produced by Model 2 is compared against the Sharpe ratio densities generated by random trading and a buy-and-hold strategy. This analysis is conducted using a 5-minute EUR/GBP foreign exchange dataset, comprising 13,636 observations spanning from January 2023 to August 2023, sourced from Bloomberg.

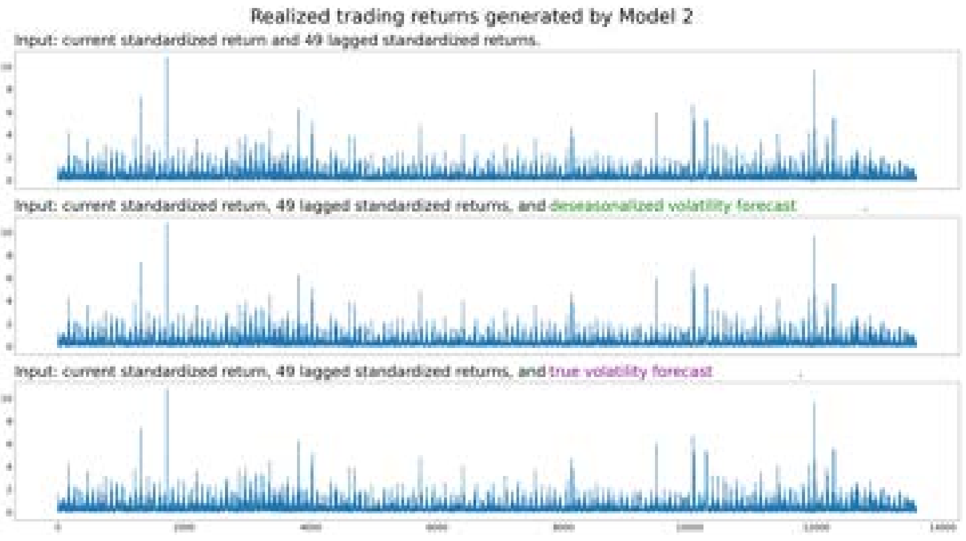
In summary, the trading performances of Models 1 and 2, across different input configurations, consistently demonstrate superior results compared to random trading or a conservative buy-and-hold strategy.

Trading Performance Comparison of Model 1 and Model 2 with Various Input Configurations

Realized trading returns, which factor in the current and one-time-step-behind trading positions as well as time-varying transaction costs, demonstrate a similar pattern across Models 1 and 2 with different input configurations, as illustrated in Figure 3.16.



(a)



(b)

Figure 3.16: Realized trading returns obtained using (a) Model 1 and (b) Model 2 with various input configurations. This analysis is conducted using a 5-minute EUR/GBP foreign exchange dataset, comprising 13,636 observations spanning from January 2023 to August 2023, sourced from Bloomberg.

Let us now turn to the analysis of descriptive statistics for realized trading returns, which are outlined in Table 3.5. The mean being greater than the median for all instances, combined with the positive skewness values, suggests that the considered trading models produce frequent modest gains with occasional significant ones. The high kurtosis values indicate that there might be significant outliers or extreme values in the realized returns, meaning that while there might be occasional high returns, there is also a potential for sudden and severe losses.

Model and input	**Mean**	**Median**	**SD**	**Skewness**	**Kurtosis**
Model 1 with lagged returns	0.44	0.28	0.54	4.27	40.15
Model 1 with lagged returns and deseasonalized volatility forecast	0.44	0.28	0.54	4.29	40.39
Model 1 with lagged returns and true volatility forecast	0.44	0.28	0.54	4.28	40.30
Model 2 with lagged returns	0.45	0.28	0.54	4.30	40.54
Model 2 with lagged returns and deseasonalized volatility forecast	0.45	0.28	0.54	4.30	40.57
Model 2 with lagged returns and true volatility forecast	0.45	0.28	0.54	4.30	40.57

Table 3.5: Descriptive statistics for realized trading returns generated by Models 1 and 2 with various input configurations. Note that "SD" stands for a standard deviation.

Now, let us shift our focus to the analysis of Sharpe ratios. The Sharpe ratio densities are visualized in Figure 3.17, while descriptive statistics for Sharpe ratios are presented in Table 3.6.

The peak of Sharpe ratio densities is around a Sharpe ratio of one (refer to the Figure 3.17). This suggests that, on average, the considered trading models have generated a return equal to one times their standard deviation or volatility. It is important to note that, due to the HFT setting, the risk-free rate in Sharpe ratio calculations is disregarded. A Sharpe ratio of one indicates a balanced risk-return profile, implying that the trading strategy neither significantly outperforms nor underperforms relative to its level of risk. Finally, note that densities for different models and input configurations are quite overlapping, suggesting that there is not a significant difference in the risk-return profiles of these models in general.

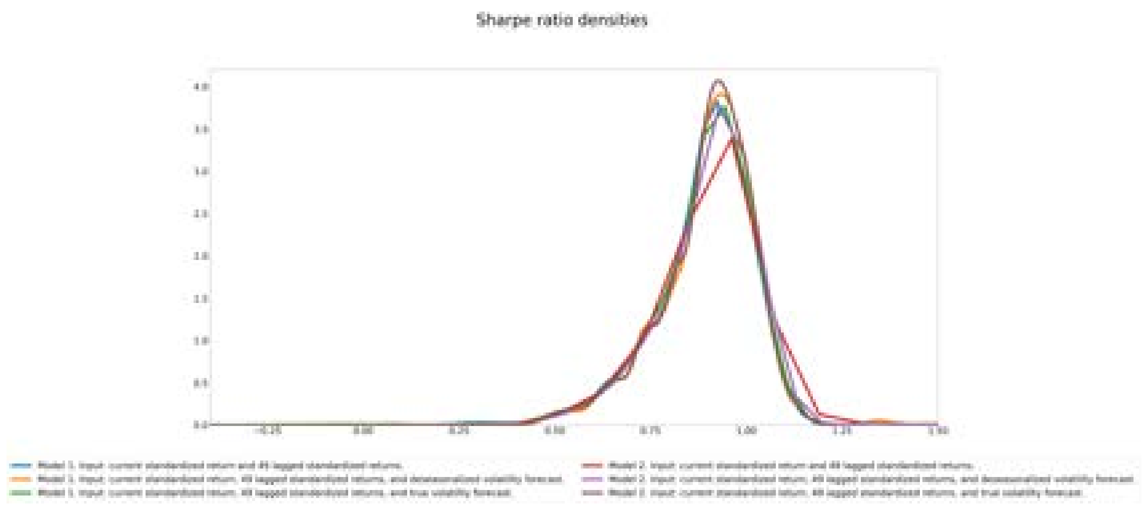


Figure 3.17: Kernel density estimators applied to Sharpe ratios obtained using Models 1 and 2 with various input configurations. This analysis is conducted using a 5-minute EUR/GBP foreign exchange dataset, comprising 13,636 observations spanning from January 2023 to August 2023, sourced from Bloomberg.

The mean and median values of the Sharpe ratio across all models and input configurations are quite similar, as shown in Table 3.6, both averaging around 0.90. This suggests that the distributions of the Sharpe ratios are rather symmetric. Models 1 and 2 using lagged returns and a conditional deseasonalized volatility forecast seem to have relatively smaller standard deviations (0.22 and 0.13, respectively) compared to when just lagged returns are used (0.25 and 0.16, respectively). This might suggest that incorporating a conditional deseasonalized volatility forecast might be providing a stabilizing effect on the trading performance of Models 1 and 2.

Moreover, when considering the conditional true volatility forecast for Model 1, there is a significant reduction in standard deviation compared to when only lagged returns are taken into account. Specifically, there is a nearly twofold reduction (from 0.25 to 0.1) in standard deviation. Note that a similar trend does not hold for Model 2.

For Model 1 with lagged returns, the standard deviation is significantly greater than that for Model 2 with lagged returns (0.25 and 0.16, respectively). This disparity can be attributed to the greater complexity of Model 1, making it more susceptible to overfitting and thus having higher variance. These findings align with relevant previous literature [20].

An inclusion of any (deseasonalized or true) conditional volatility forecast has a significant stabilizing effect on the trading performance of Model 1. Conversely, for Model 2, the inclusion of a deseasonalized conditional volatility forecast provides a stabilizing effect, while the inclusion of the true conditional volatility forecast has the opposite effect.

It is important to note that the skewness patterns for Models 1 and 2 are entirely distinct. Model 1 with lagged returns and Model 1 with lagged returns and deseasonalized volatility exhibit significant positive skewness (22.09 and 30.31, respectively). Such high positive skewness indicates that the Sharpe ratio distributions for these models have a long right tail, implying frequent small losses and a few extremely successful periods. The neural network layer tailored for input transformation could enable Model 1 to recognize intricate data patterns, leading to occasional strong performance. However, it could also mean that Model 1 is overfitting to certain patterns, often leading to minor losses, but sometimes achieving highly accurate predictions. In contrast, Model 2 demonstrates relatively small negative skewness in most input configurations, implying a longer left tail and suggesting that the model experiences frequent small gains and a few extremely poor periods.

Both Models 1 and 2, in several configurations, exhibit extremely high kurtosis, suggesting heavy tails and, therefore, a higher likelihood of extreme values in the Sharpe ratios.

Model and input	**Mean**	**Median**	**SD**	**Skewness**	**Kurtosis**
Model 1 with lagged returns	0.90	0.91	0.25	22.09	739.57
Model 1 with lagged returns and deseasonalized volatility forecast	0.90	0.92	0.22	30.31	1,359.23
Model 1 with lagged returns and true volatility forecast	0.90	0.92	0.12	-0.80	0.78
Model 2 with lagged returns	0.88	0.91	0.16	-2.93	16.57
Model 2 with lagged returns and deseasonalized volatility forecast	0.89	0.92	0.13	-1.22	2.65
Model 2 with lagged returns and true volatility forecast	0.88	0.91	0.19	-10.68	320.90

Table 3.6: Descriptive statistics for Sharpe ratios generated by Models 1 and 2 with various input configurations. Note that "SD" stands for a standard deviation.

Model 1 appears to offer higher profit opportunities but comes with greater risks, as shown by its positive skewness and pronounced kurtosis in various configurations. It often shows strong positive outcomes, but this suggests a higher chance of extreme results, making it more volatile. On the other hand, Model 2 tends to provide steady, smaller profits and generally stays away from big losses, as indicated by its usual negative skewness. Nonetheless, in certain settings, it often registers minor profits, but also risks occasional substantial losses, evidenced by high kurtosis in those setups.

Another method to compare the trading performances of Models 1 and 2, with various input configurations, is by examining the confidence intervals generated using bootstrapped samples of Sharpe ratios. Bootstrapping is a robust statistical method for comparing trading performance because it does not rely on any restrictive distributional assumptions about the data. Refer to Figure 3.18 for the visual representation of the bootstrapped confidence intervals.

Note that narrower confidence intervals indicate greater precision in the estimation. The narrowest confidence intervals are produced by Model 2 with lagged returns and deseasonalized conditional volatility forecast and Model 1 with lagged returns and deseasonalized conditional volatility forecast.

The confidence interval with the highest mean is generated by Model 1 with lagged returns and true conditional volatility forecast. However, there is a significant overlap between this confidence interval and the one produced by Model 1 with lagged returns only. Therefore, the difference in trading performance between Model 1 with lagged returns and true conditional volatility forecast and Model 1 with lagged returns is not statistically significant.

At the same time, it can be concluded that both Model 1 with lagged returns and true conditional volatility forecast, as well as Model 1 with lagged returns, outperform both Model 2 with all input configurations and Model 1 with lagged returns and deseasonalized conditional volatility estimate. This conclusion is based on the observation that the confidence intervals for Sharpe ratios generated by Model 1 with lagged returns and true conditional volatility forecast, and Model 1 with lagged returns are centered around a higher Sharpe ratio mean and do not significantly overlap with the confidence intervals of the models with different input configurations.

Hence, the inclusion of an additional neural network layer for input transformation indeed enhances trading performance.

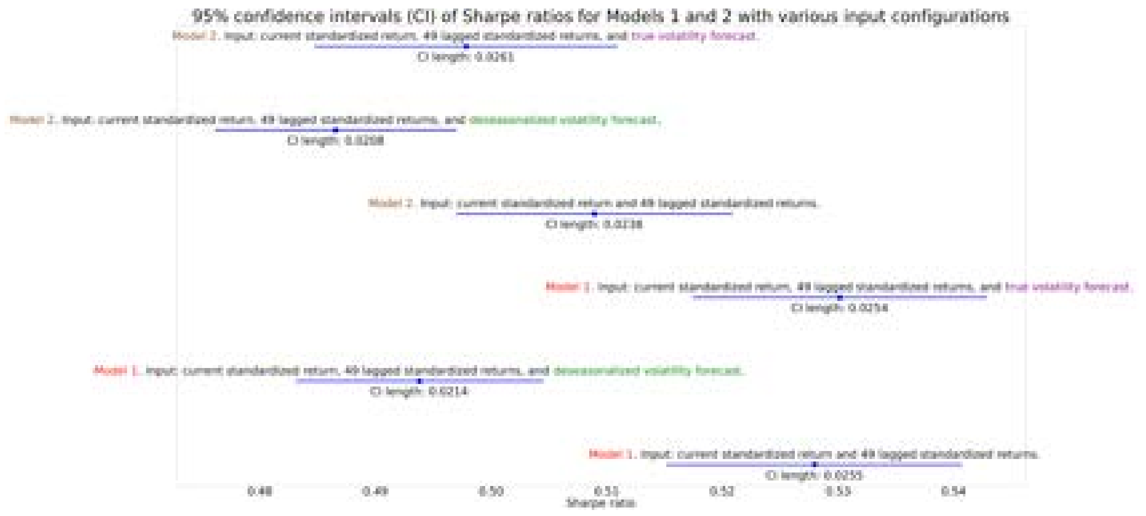


Figure 3.18: 95% confidence intervals of Sharpe ratios for Models 1 and 2 with various input configurations with 10,000 bootstrapped samples for each model and input configuration. The length of confidence intervals is shown below their corresponding graphical representations. This analysis is conducted using a 5-minute EUR/GBP foreign exchange dataset, comprising 13,636 observations spanning from January 2023 to August 2023, sourced from Bloomberg.

Therefore, although the use of true conditional volatility forecast may seem to improve performance of Model 1 on average, it cannot be definitively claimed that it offers a significant advantage over the Model 1 solely relying on lagged returns, at least when considering statistical analysis.

A sensitivity analysis of the responses produced by Model 1 to slight variations in the architecture and inputs (for instance, reconsidering the number of lagged returns) may be performed to determine which input configuration (lagged returns or lagged returns and a true conditional volatility forecast) provides more robust results.

4. LIMITATIONS AND FUTURE AVENUES FOR RESEARCH

In the present research, bid and ask prices were estimated using low and high prices over a 5-minute interval, respectively. As described in the reference [34], such an approximation might not faithfully represent the price at which the trading system could actually transact. Additionally, in a manner analogous to the scenarios described in the reference [34], the trading system may be detecting market microstructure effects that are not pragmatically tradeable in a real-time framework. To truly evaluate the trading system presented in this research, it is essential to conduct an actual trading with real-time prices.

The trading system presented here does not incorporate reinvestment of trading profits, as is done in the reference [36]. Future research could potentially yield benefits by integrating the reinvestment of trading profits into the DRL trading system.

Similar to the reference [20], the DRL method appears to encounter an issue prevalent in gradient ascent training of neural networks: it possesses numerous fixed parameters that can only be adjusted through a process of trial and error. This, in turn, requires an extensive number of experiments, which may prove to be overly time-consuming for HFT given tight time constraints.

The presented DRL trading system pertains to the EUR/GBP high-frequency FX market. Similar studies [20] have demonstrated that the performance of DRL trading models can vary significantly among different currency markets; therefore, the presented model may not generalize well to other FX markets.

It is widely acknowledged within the trading community that automated trading systems tend to operate successfully for a period, only to abruptly lose their profitability [13, 1]. In my opinion, this phenomenon may be attributable to the prevalent belief in financial market efficiency (note that efficient market hypothesis was discussed in Subsection 2.4.1). Temporary mispricings, or deviations from the recognized fundamental asset value, often arise in the financial market but do not inherently conflict with the notion of efficient markets. In fact, once such mispricings are identified, they are swiftly exploited and consequently neutralized. Thus, in essence, the financial market reverts to efficiency. Within the realm of HFT, this correction transpires exceedingly quickly. Another reason automated trading systems can rapidly lose their profitability is the swift exposure of these algorithms to competitors [1].

Therefore, a diligent management and ongoing monitoring of the designed trading system are necessary. In the context of the reference [13], an automated trading system incorporates a performance management layer that identifies unusual performance at an early stage to protect accumulated profits. If such a situation arises, the system is automatically paused, and a warning is triggered [13]. The present research does not provide any risk management instruments, and the design of such instruments could be a subject for future research.

5. CONCLUSION

This comprehensive exploration delves into the world of HFT, with a specific focus on the EUR/GBP FX market. It uncovers valuable insights regarding the trading performance of a DRL model, comparing its performance with and without a shallow neural network structure for input transformation, across various input configurations, including lagged price returns, lagged price returns with a true conditional volatility forecast, and lagged price returns with a deseasonalized conditional volatility forecast. The objective function of the trading system is a DSR, an online performance metric that contributes to simplifying the complexity of the trading algorithm.

When benchmarked against random trading and a conservative buy-and-hold strategy, performances of the designed DRL trading systems stand out as exceptional.

Incorporation of a shallow neural network for feature learning boosts the risk-adjusted trading performance of the DRL trading model, however, at a cost of higher volatility. These findings align with the relevant previous literature [14].

Incorporation of a deseasonalized conditional volatility forecast in the input layer of both trading models is found to significantly stabilize the trading performance. Simultaneously, incorporating a true conditional volatility forecast, on average, improves the trading performance of the DRL model with a shallow neural network for feature learning. Nevertheless, whether there is a statistically significant difference in the performances of the model with a shallow neural network for feature learning that considers only lagged price returns, and the model with a shallow neural network for feature learning that considers lagged price returns and a true conditional volatility forecast, remains ambiguous. These findings align well with previous research [13]. Potentially, the designed DRL may be effectively capturing all the necessary information from the lagged returns series and may not necessitate forecasting the true conditional volatility.

A. APPENDIX

A.1 Excess return standard deviation expressed through the first and the second raw moments of the excess return distribution

Allow me to elaborate a bit on the final result in demonstrated by Equation A.1 and demonstrate the step-by-step process of deriving it. Substitution of the definition of the return average in the definition of return standard deviation results in the equality 1 of A.1. Expanding the square brackets yields the following equality 2. The next equalities 3, 4 and 5 are the results of simple algebraic manipulations. In the penultimate equality 6 the numerator of an algebraic fraction inside the square root is multiplied and divided by T to facilitate further derivations. The last equality 7 is just a convenient representation the previous one.

$$
\begin{aligned}
\text{Standard deviation}(R_t) &\overset{(1)}{=} \sqrt{\frac{\sum_{t=1}^{T}(R_t - \frac{\sum_{t=1}^{T} R_t}{T})^2}{T-1}} \\
&\overset{(2)}{=} \sqrt{\frac{\sum_{t=1}^{T} R_t^2 - \sum_{t=1}^{T} 2R_t \frac{\sum_{t=1}^{T} R_t}{T} + \sum_{t=1}^{T} \left(\frac{\sum_{t=1}^{T} R_t}{T}\right)^2}{T-1}} \\
&\overset{(3)}{=} \sqrt{\frac{\sum_{t=1}^{T} R_t^2 - \frac{2}{T}(\sum_{t=1}^{T} R_t)^2 + T\left(\frac{\sum_{t=1}^{T} R_t}{T}\right)^2}{T-1}} \\
&\overset{(4)}{=} \sqrt{\frac{\sum_{t=1}^{T} R_t^2 - \frac{2}{T}(\sum_{t=1}^{T} R_t)^2 + \frac{1}{T}(\sum_{t=1}^{T} R_t^{\tilde{j}})^2}{T-1}} \\
&\overset{(5)}{=} \sqrt{\frac{\sum_{t=1}^{T} R_t^2 - \frac{1}{T}(\sum_{t=1}^{T} R_t)^2}{T-1}} \\
&\overset{(6)}{=} \sqrt{\frac{T\left(\frac{\sum_{t=1}^{T} R_t^2}{T} - (\frac{\sum_{t=1}^{T} R_t}{T})^2\right)}{T-1}} \\
&\overset{(7)}{=} \left(\frac{T}{T-1}\right)^{1/2}\left(\frac{\sum_{t=1}^{T} R_t^2}{T} - \left(\frac{\sum_{t=1}^{T} R_t}{T}\right)^2\right)^{1/2}.
\end{aligned}
\tag{A.1}
$$

A.2 Sharpe ratio S_T derivative with respect to the trading system parameters $\boldsymbol{\theta}_{\mathbf{t}}$

1690 The mathematical expression for the Sharpe ratio

$$
S_T\left\{A_T\Big(R_t[F_t(\boldsymbol{\theta}_{\mathbf{t}}), F_{t-1}(\boldsymbol{\theta}_{\mathbf{t}})]\Big), B_T\Big(R_t[F_t(\boldsymbol{\theta}_{\mathbf{t}}), F_{t-1}(\boldsymbol{\theta}_{\mathbf{t}})]\Big)\right\} \tag{A.2}
$$

can be described in words as follows.

1. The derivative of the SR S_T defined by Equation 2.24. SR S_T is a function of parameters A_T and B_T, K_T is a constant irrelevant for optimization [36].

2. A_T is a function of trading return R_t (from Equation 2.20) and so is B_T (from Equation 2.21).

3. Trading return R_t is a function of a trading positions $F_t,\ F_{t-1}$ at time periods $t,\ t-1$, respectively, from Equation 2.14.

4. Trading position F_t is identified in the decision function $F(\cdot)$ from Equation 2.8 and is a function of the trading system parameters, which are denoted as $\boldsymbol{\theta}_\mathbf{t}$.

Additionally, a comprehensive, step-by-step explanation is provided for taking the derivative of SR S_T with respect to trading system parameters $\boldsymbol{\theta}_\mathbf{t}$.

Allow me to elaborate a bit on the final result of Equation A.3. The equality 1 is the result of applying the quotient rule to find the derivative of a complex SR function defined in Equation 2.24. The next equality 2 is the result of expanding the brackets in the denominator. In the following equality 3, a common factor K_T in the numerator and the denominator is reduced. The final equality 4 is simply a convenient representation of the previous one.

$$\begin{aligned}
\frac{\mathrm{d}S_T}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} &\overset{(1)}{=} \sum_{i=1}^{T}\left\{\frac{K_T(B_T-A_T^2)^{1/2}\dfrac{\mathrm{d}A_T}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} - A_T\dfrac{\mathrm{d}K_T(B_T-A_T^2)^{1/2}}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}}}{(K_T(B_T-A_T^2)^{1/2})^2}\right\} \\
&\overset{(2)}{=} \sum_{i=1}^{T}\left\{\frac{K_T(B_T-A_T^2)^{1/2}\dfrac{\mathrm{d}A_T}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} - A_T K_T\dfrac{\mathrm{d}(B_T-A_T^2)^{1/2}}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}}}{K_T^2(B_T-A_T^2)}\right\} \\
&\overset{(3)}{=} \sum_{i=1}^{T}\left\{\frac{(A_T-A_T^2)^{1/2}\dfrac{\mathrm{d}A_T}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} - A_T\dfrac{\mathrm{d}(B_T-A_T^2)^{1/2}}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}}}{K_T(B_T-A_T^2)}\right\} \\
&\overset{(4)}{=} \sum_{i=1}^{T}\frac{1}{K_T(B_T-A_T^2)}\left\{(B_T-A_T^2)^{1/2}\frac{\mathrm{d}A_T}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}} - A_T\frac{\mathrm{d}(B_T-A_T^2)^{1/2}}{\mathrm{d}\boldsymbol{\theta}_\mathbf{t}}\right\}
\end{aligned} \tag{A.3}$$

Furthermore, let me introduce the intermediate results necessary to obtain the final expression for Equation A.3. To take the total derivative of A_T and B_T with respect to $\boldsymbol{\theta}_\mathbf{t}$, one must apply the chain rule for multivariable functions, as A_T and B_T are functions of R_t, which in turn is a function of two variables F_t and F_{t-1}, as shown in Equation A.2.

The results are represented in the equalities 1 and 5 in Equation A.4. The equalities 3 and 4 are intermediate results necessary for the second equality in Equation A.4. The equality 2 represents the total derivative of the denominator of S_t, defined in Equation 2.24, with respect to $\boldsymbol{\theta_t}$. The equalities 6 and 7 are partial derivatives of A_t from Equation 2.20 and B_T from Equation 2.21 with respect to R_t, respectively.

$$\begin{aligned}
\frac{\mathrm{d}A_T}{\mathrm{d}\boldsymbol{\theta_t}} &\overset{(1)}{=} \frac{\mathrm{d}A_T}{\mathrm{d}R_t}\left(\frac{\mathrm{d}R_t}{\mathrm{d}F_t}\frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}}\right) \\
\frac{\mathrm{d}(B_T - A_T^2)^{1/2}}{\mathrm{d}\boldsymbol{\theta_t}} &\overset{(2)}{=} \frac{\mathrm{d}(B_T - A_T^2)^{1/2}}{\mathrm{d}B_T}\frac{\mathrm{d}B_T}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}(B_T - A_T^2)^{1/2}}{\mathrm{d}A_T}\frac{\mathrm{d}A_T}{\mathrm{d}\boldsymbol{\theta_t}} \\
\frac{\mathrm{d}(B_T - A_T^2)^{1/2}}{\mathrm{d}B_T} &\overset{(3)}{=} \frac{1}{2(B_T - A_T^2)^{1/2}} \\
\frac{\mathrm{d}(B_T - A_T^2)^{1/2}}{\mathrm{d}A_T} &\overset{(4)}{=} \frac{2A_T}{2(B_T - A_T^2)^{1/2}} = \frac{A_T}{(B_T - A_T^2)^{1/2}} \\
\frac{\mathrm{d}B_T}{\mathrm{d}\boldsymbol{\theta_t}} &\overset{(5)}{=} \frac{\mathrm{d}B_T}{\mathrm{d}R_t}\left(\frac{\mathrm{d}R_t}{\mathrm{d}F_t}\frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}}\right) \\
\frac{\mathrm{d}A_T}{\mathrm{d}R_t} &\overset{(6)}{=} \frac{1}{T} \\
\frac{\mathrm{d}B_T}{\mathrm{d}R_t} &\overset{(7)}{=} \frac{2R_t}{T}
\end{aligned} \tag{A.4}$$

Finally, let me substitute all the intermediate results from the Equation A.4 into the Equation A.3. After a series of algebraic manipulations, one arrives at the final result shown in Equation A.5. This important intermediate result is represented in the relevant literature [36].

$$
\begin{aligned}
\frac{\mathrm{d}S_T}{\mathrm{d}\boldsymbol{\theta_t}} =& \sum_{i=1}^{T} \frac{1}{K_T(B_T - A_T^2)} \cdot \Bigg\{ (B_T - A_T^2)^{1/2} \frac{1}{T} \left(\frac{\mathrm{d}R_t}{\mathrm{d}F_t} \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} \frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right) \\
& - A_T \Bigg[\frac{1}{2(B_T - A_T^2)^{1/2}} \frac{2R_t}{T} \left(\frac{\mathrm{d}R_t}{\mathrm{d}F_t} \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} \frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right) \\
& + \frac{A_T}{(B_T - A_T^2)^{1/2}} \frac{1}{T} \left(\frac{\mathrm{d}R_t}{\mathrm{d}F_t} \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} \frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right) \Bigg] \Bigg\} \\
=& \sum_{i=1}^{T} \frac{1}{K_T(B_T - A_T^2)} \Bigg\{ \left(\frac{\mathrm{d}R_t}{\mathrm{d}F_t} \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} \frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right) \\
& \cdot \left[(B_T - A_T^2)^{1/2} \frac{1}{T} - \frac{A_T R_t}{(B_T - A_T^2)^{1/2}} \frac{1}{T} + \frac{A_T^2}{(B_T - A_T^2)^{1/2}} \frac{1}{T} \right] \Bigg\} \\
=& \frac{1}{T} \sum_{i=1}^{T} \left\{ \frac{1}{K_T(B_T - A_T^2)} \right\} \left\{ (B_T - A_T^2)^{1/2} - \frac{A_T R_t - A_T^2}{(B_T - A_T^2)^{1/2}} \right\} \left\{ \frac{\mathrm{d}R_t}{\mathrm{d}F_t} \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} \frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right\} \\
=& \frac{1}{T} \sum_{i=1}^{T} \left\{ \frac{1}{K_T(B_T - A_T^2)} \right\} \left\{ \frac{(B_T - A_T^2) - (A_T R_t - A_T^2)}{(B_T - A_T^2)^{1/2}} \right\} \left\{ \frac{\mathrm{d}R_t}{\mathrm{d}F_t} \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} \frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right\} \\
=& \frac{1}{T} \sum_{i=1}^{T} \left\{ \frac{1}{K_T(B_T - A_T^2)} \right\} \left\{ \frac{B_T - A_T R_t}{(B_T - A_T^2)^{1/2}} \right\} \left\{ \frac{\mathrm{d}R_t}{\mathrm{d}F_t} \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} \frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right\} \\
=& \frac{1}{T} \sum_{i=1}^{T} \left\{ \frac{B_T - A_T R_t}{K_T(B_T - A_T^2)^{3/2}} \right\} \left\{ \frac{\mathrm{d}R_t}{\mathrm{d}F_t} \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} \frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right\}
\end{aligned}
\tag{A.5}
$$

A.3 Reinforcement Learning Performance Function Derivative

As mentioned previously, the performance function in Equation 2.35 is influenced by the entire time sequence of trades, considering that the trading return R_t is subject to path-dependent effects. The trading return R_t defined in Equations 2.13 or 2.14 and is a function of the current F_t and the previous F_{t-1} trading positions.

Therefore, in Equation 1 from A.6, a chain rule is utilized for U_T. Subsequently, in Equation 2, a multivariable chain rule is applied for R_t.

$$
\begin{aligned}
\frac{\mathrm{d}U_T(\boldsymbol{\theta_t})}{\mathrm{d}\boldsymbol{\theta_t}} &\overset{(1)}{=} \sum_{t=1}^{T} \frac{\mathrm{d}U_T}{\mathrm{d}R_t} \frac{\mathrm{d}R_t(\boldsymbol{\theta_t})}{\mathrm{d}\boldsymbol{\theta_t}} \\
&\overset{(2)}{=} \sum_{t=1}^{T} \frac{\mathrm{d}U_T}{\mathrm{d}R_t} \left\{ \frac{\mathrm{d}R_t}{\mathrm{d}F_t} \frac{\mathrm{d}F_t}{\mathrm{d}\boldsymbol{\theta_t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}} \frac{\mathrm{d}F_{t-1}}{\mathrm{d}\boldsymbol{\theta_t}} \right\}.
\end{aligned}
\tag{A.6}
$$

A.4 EUR/GBP 5-minute Standardized Log Return analysis

Phillips-Perron Test

Test Statistic	-141.245
P-value	0.000
Lags	51

Table A.1: Phillips-Perron Test for the EUR/GBP 5-minute standardized log return series. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Trend: Constant and Linear Time Trend

Critical Values: -3.96 (1%), -3.41 (5%), -3.13 (10%)

Null Hypothesis: The process contains a unit root.

Alternative Hypothesis: The process is weakly stationary.

Pure Diurnal Dummy Variable Model Residual (Deseasonalized EUR/GBP 5-minute Log Return) Inspection

As noted earlier, intraday returns are anticipated to show heteroskedasticity. An examination will be conducted on Pure Diurnal Dummy Variable Model (refer to Equation 2.42) seasonally adjusted log return series $r_t^{deseasonalized}$ to identify the heteroskedasticity presence.

Upon visual inspection of Figure A.1, it becomes evident that the squared seasonally adjusted log return series $r_t^{deseasonalized}$ of standardized 5-minute log returns for EUR/GBP are primarily clustered around 0 at the beginning and end of the analyzed period. However, there is a significant increase in the dispersion of squared residuals during the middle part of the considered period. This pattern corresponds to a period of heightened volatility in the GBP foreign exchange rate. On the whole, the squared residuals display substantial dispersion throughout the entire observed timeframe.

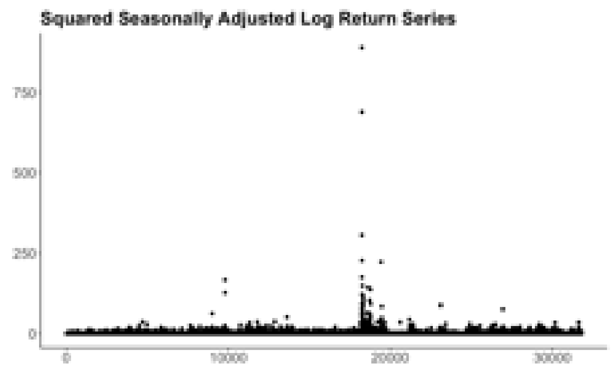


Figure A.1: Squared seasonally adjusted log return series $r_t^{deseasonalized}$ from the Pure Diurnal Dummy Variable Model 2.42 that was estimated using ordinary least squares for standardized EUR/GBP 5-minute log returns during active trading hours. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

The autocorrelation function of squared seasonally adjusted log return series $r_t^{deseasonalized}$ derived from the Pure Diurnal Dummy Variable Model 2.42 depicted in Figure A.2 exhibits statistically significant differences from zero. These findings cast doubt on the assumption of homoskedasticity in the residuals. Nevertheless, it is possible that these observations might arise due to insignificant data artifacts. In order to arrive at a well-informed conclusion, a formal test for heteroskedasticity will be carried out.

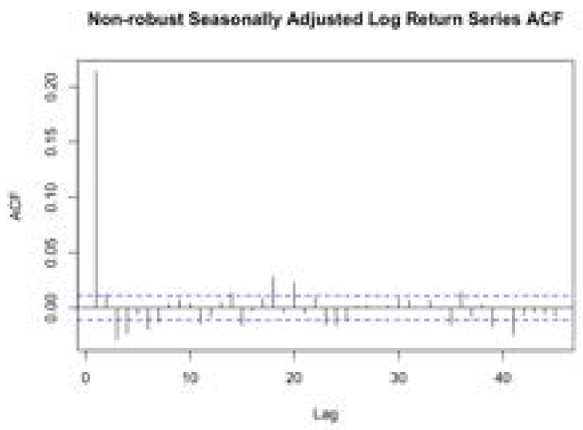


Figure A.2: Autocorrelation function (ACF) of squared residuals from the Pure Diurnal Dummy Variable Model 2.42 that was estimated using ordinary least squares for standardized EUR/GBP 5-minute log returns during active trading hours. The 95% confidence interval boundaries are indicated by blue dotted lines. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

To formally assess heteroskedasticity, the Autoregressive Conditional

Heteroskedasticity Lagrange Multiplier (ARCH-LM) test will be employed [17]. The null hypothesis of the ARCH-LM test asserts the absence of ARCH effects or autocorrelation. This test involves estimating a regression of squared residuals on lagged squared residuals and assessing the joint significance of the regression coefficients. The ARCH-LM test is automatically conducted using R.

The resulting ARCH-LM test p-value presented in Table A.2 is smaller than any reasonable significance level, therefore, the null hypothesis of no ARCH effects (no heteroskedasticity) is rejected. Therefore, the further analysis cannot be based on the assumption of homoskedasticity of standardized EUR/GBP 5-minute log returns.

Chi-squared	9,546.6
P-value	$< 2.2 \cdot 10^{-16}$

Table A.2: ARCH LM-test. Null hypothesis: no ARCH effects.

This indicates that in subsequent analyses, it is necessary to utilize heteroskedasticity-consistent robust standard errors [5, 37]. Specifically, research has determined that the HC_3 robust standard error estimator is the favored option among the range of available heteroskedasticity-consistent covariance matrix estimators [27].

Pure Diurnal Dummy Variable Model

A chi-squared test with a 5% significance level is conducted to check whether there is an intraday seasonal pattern in standardized EUR/GBP 5-minute log returns, which involves looking at how coefficients of dummy variables behave together in the Pure Diurnal Dummy Variable Model (refer to Subsection 2.4.2). Locate the estimated model using pure diurnal dummy variables with HC_3 standard error estimators in Table A.3.

For the results of the chi-squared test for joint significance of the dummy variable coefficients refer to Table A.4. The critical value for the chi-squared test at this significance level, considering 15 degrees of freedom, is 24.995790. The calculated chi-squared statistic is 41.9863. In simple terms, the chi-squared statistic (41.9863) is greater than the critical value (24.995790). This means that there are enough statistical evidence to say there is an intraday seasonality in standardized EUR/GBP 5-minute log returns at 95% confidence level.

Table A.3: The results from ordinary least squares regression estimation of Pure Diurnal Dummy Variable Model 2.42 with HC_3 standard error estimators for standardized EUR/GBP 5-minute log returns during active trading hours. The symbol *** indicates statistical significance at 1% level, the symbol ** indicates statistical significance at 5% level, the symbol * indicates statistical significance at 10% level. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

	Estimate β_i	**HC_3 standard error s_i**	**t-value**	**P-value**
D_1	-0.00005967222	0.02418399	-0.002467426	0.99803129601
D_2	-0.02950045936*	0.01559262	-1.891949582	0.05850677464
D_3	0.02194719201	0.01480139	1.482779385	0.13814294452
D_4	-0.00196540918	0.01251925	-0.156991011	0.87525292882
D_5	-0.00258691048	0.01047822	-0.246884613	0.80499911851
D_6	0.01202893175	0.01572692	0.764862530	0.44435908371
D_7	-0.01462186305	0.01257705	-1.162583096	0.24500741653
D_8	-0.00784294971	0.01241750	-0.631604585	0.52764982972
D_9	-0.00353177775	0.01461803	-0.241604236	0.80908839701
D_{10}	-0.03712752311	0.02261756	-1.641535256	0.10069624321
D_{11}	0.10365178883***	0.02590238	4.001631743	0.00006304959
D_{12}	-0.07486141220**	0.02967703	-2.522536817	0.01165601290
D_{13}	0.03059705907	0.03276410	0.933859423	0.35038350287
D_{14}	0.04569186100	0.03001993	1.522051080	0.12800622518
D_{15}	-0.06617396227**	0.03085483	-2.144687426	0.03198540081
D_{16}	0.029321469120	0.03320103	0.883149470	0.37716219433

1775

Table A.4: Chi-squared test for the joint significance of coefficients of the Pure Diurnal Dummy Variable Model 2.42 with HC$_3$ standard error estimators for standardized EUR/GBP 5-minute log returns during active trading hours. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Observations	31,817
Mean coefficient estimate $\bar{\beta} = \frac{\sum_{i=1}^{16} \beta_i}{16}$	0.000310398
Chi-squared test statistic $\chi^2_{st} = \sum_{i=1}^{16} \left(\frac{\hat{\beta}_i - \bar{\beta}}{s_i} \right)^2$	41.9863
Critical value $\chi^2_{15}(0.05)$	24.995790

Conditional Mean Model Residual Inspection

Considering the suitability of the ARMA(2,1) Model for the deseasonalized EUR/GBP 5-minute log return series (as shown in the estimated model 3.63), the expected outcome is that the fitted residuals, denoted as e_t, will exhibit characteristics similar to white noise [42]. As a reminder, white noise signifies a sequence of uncorrelated random variables with a consistent mean of zero and unchanging variance [42].

Upon visually inspecting a time plot of the fitted residuals (refer to Figure A.3), it is evident that these residuals exhibit an average value of zero and maintain a relatively consistent variance.

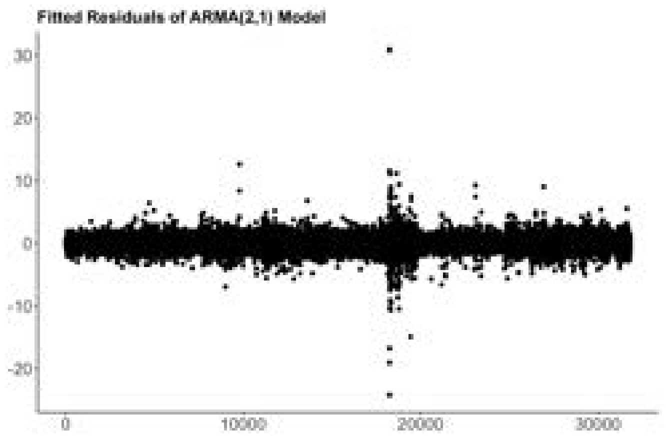


Figure A.3: A time-plot of the fitted residuals for deseasonalized EUR/GBP 5-minute log return series. Calculation of the fitted residuals is based on ARMA(2,1) model. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Formal test for fitted residual zero mean is conducted [42]. Null hypothesis $\mathcal{H}_0 : \mu = 0$ against an alternative hypothesis $\mathcal{H}_0 : \mu \neq 0$. A t-test statistic under the null hypothesis is

$$\frac{\bar{e}\sqrt{T}}{s_e} \sim t_{T-k-1},$$

where $\bar{e} = \dfrac{\sum_{t=1}^{T} e_t}{T} = 2.501617 \cdot 10^{-6}$; $T = 31,816$; $k = p + q = 2 + 1 = 3$; $s_e = \dfrac{1}{T-k-1}\sum_{t=1}^{T}(e_t - \bar{e})^2 = 0.9506613$ [42].

A t-test statistic of 0.00046938 is being contrasted with critical values derived from the normal distribution. It is important to recognize that the Student's t-distribution approaches the normal distribution as the degrees of freedom increase, typically becoming more apparent when the degrees of freedom reach approximately 100 or higher. In this instance, the test statistic is smaller than any critical value of the normal distribution within a reasonable significance level range. Consequently, the null hypothesis cannot be rejected at a reasonable significance level. Hence, the formal test leads to the determination that the mean of the fitted residuals is equal to zero.

The Figure A.4 indicates that certain individual values of the ACF display statistically significant results. However, the primary focus of the research lies in determining the joint significance of these ACF values.

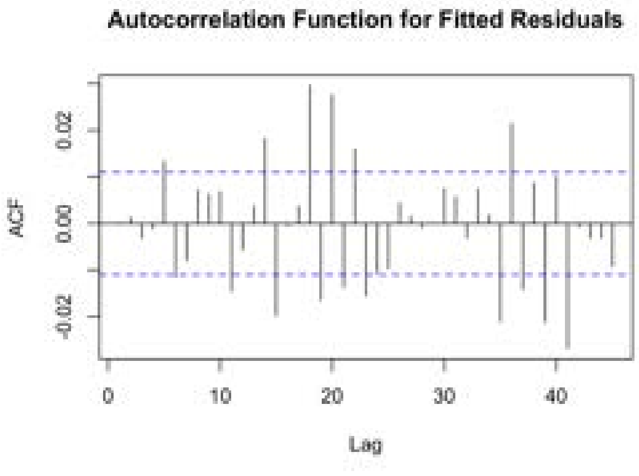


Figure A.4: Autocorrelation function (ACF) for the fitted residuals for deseasonalized EUR/GBP 5-minute log return series. Calculation of the fitted residuals is based on ARMA(2,1) model. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

In a formal context, one can investigate the existence of autocorrelation in the fitted residuals through a Ljung-Box test [42]. The Ljung-Box test is based on the assumption that the residuals of a stationary autoregressive moving-average model are independently and identically distributed according to a normal distribution with zero mean and constant variance [26]. Given that the fitted residuals exhibit a high kurtosis of 63.12099, the results of the Ljung-Box test may be invalidated. Thus, a regression-based robust LM test for serial correlation is conducted [26].

The Ljung-Box test and the regression-based robust LM test are performed automatically in `R`, and the results are presented in Table A.5. The number of lags, denoted as m, has been set to 10, 20, and 40 based on the visual insights provided by Figure A.4. The robust chi-squared test statistics for 10, 20, and 40 lags are all below their respective critical values at the 5% significance level. Consequently, the null hypothesis of no autocorrelation cannot be rejected. Upon examination of the Table A.5, it becomes evident that the Ljung-Box-Pierce test statistics are notably higher compared to the robust chi-squared test statistics.

Number of lags (m)	10	20	40
Ljung-Box-Pierce Test Statistic	16.793	109.46	198.76
Robust Chi-squared Test Statistic	1.362774	20.06366	36.28229
Degrees of Freedom $df = m - 3$	7	17	37
Critical Value $\chi^2(0.05, df)$	14.067	27.587	52.192

Table A.5: Ljung-Box-Pierce Test Statistic and Robust Chi-squared Test Statistic for the fitted residuals for deseasonalized EUR/GBP 5-minute log return series. Null hypothesis: the fitted residuals are independently distributed and exhibit no autocorrelation. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

In conclusion, the selected ARMA(2,1) model appears to be an optimal choice for modeling the deseasonalized 5-minute log returns of EUR/GBP.

Pure Diurnal Dummy Variable Model for EUR/GBP 5-minute Deseasonalized Return Volatility

At first, intraday volatility is going to be inspected for the presence of intraday seasonality (refer to Subsection 2.4.5).

In Figure A.5, a distinct U-shaped pattern is evident in intraday volatility. The instance of minimal volatility roughly coincides with the midpoint of the active trading hours, commonly associated with the lunch break effect [12]. Furthermore, towards the conclusion of the active trading hours, the volatility is markedly higher compared to the commencement of these hours earlier in the day.

Various research has examined exchange rate volatility across three prominent exchange rate markets: American, European and Asian [5, 12]. Notably, in Figure A.5, a marked rise in volatility is evident around 2:30 p.m. London time (11th 30-minute interval or 5.5 hours after 9 a.m. London time), coinciding with 9:30 a.m. New York time, which is when the U.S. foreign exchange FX market commences trading. Hence, the increased volatility towards the close of the trading day directly corresponds to the period when both the European and U.S. FX markets are operational concurrently. This empirical observation is consistent with the conclusions drawn in previous pertinent studies [5].

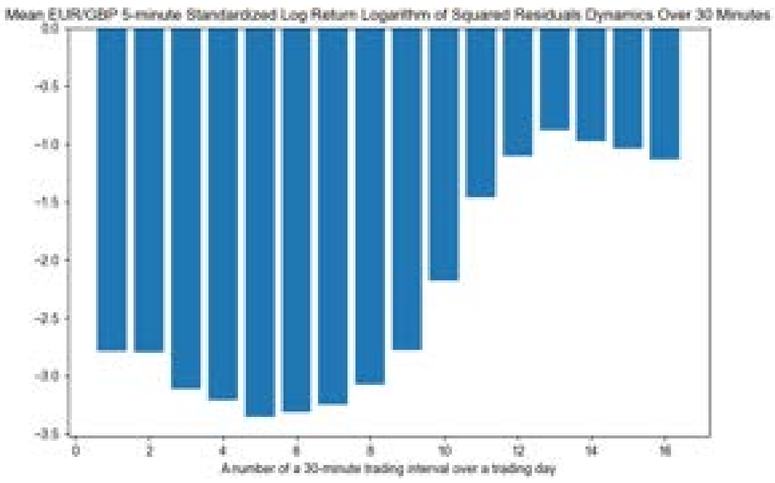


Figure A.5: Mean EUR/GBP 5-minute standardized log return logarithm of squared residuals dynamics over 30 minute intervals during active trading hours. There are sixteen 30 minute intervals over active trading hours. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Formal statistical tests designed to identify intraday seasonality in intraday volatility consistently lead to the rejection of the null hypothesis of no of intraday seasonality at nearly all levels of significance. Note that the chi-squared test statistic exceeds the 5% critical value, as shown in Table A.7. Furthermore, the remarkably low p-values linked with the t-statistics for all the Pure Diurnal Dummy Variable Model coefficients provide additional evidence of the persistent intraday seasonality pattern (refer to Table A.6).

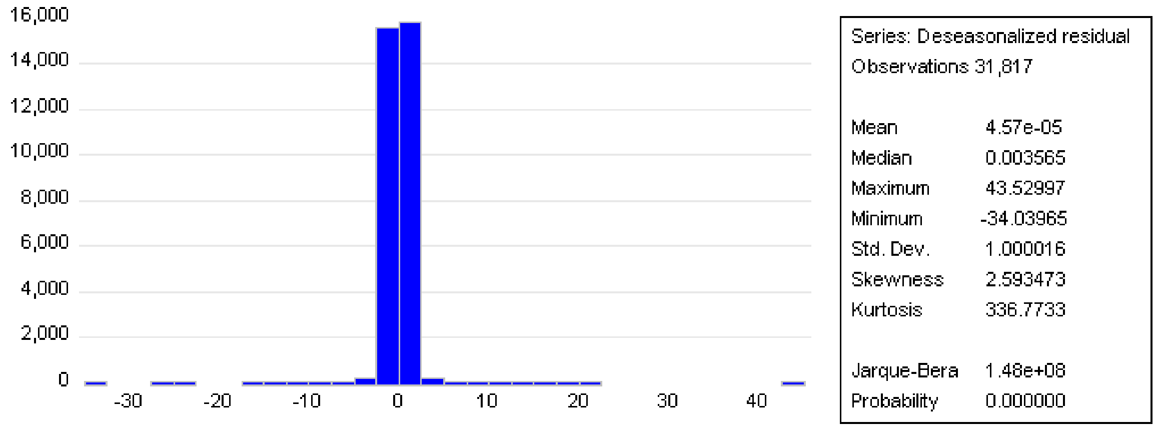
Table A.6: The results from ordinary least squares regression estimation of pure diurnal dummy model 2.52 with HC_3 standard error estimators for logarithm of squared residuals from ARMA(2,1) model 3.63 for deseasonalized EUR/GBP 5-minute log returns during active trading hours. The symbol *** indicates statistical significance at 1% level, the symbol ** indicates statistical significance at 5% level, the symbol * indicates statistical significance at 10% level. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

	Estimate β_i	**HC_3 standard error s_i**	**t-value**	**P-value**
D_1	-2.7835211***	0.05165577	-53.88597	0.00000
D_2	-2.7911662***	0.05257648	-53.08773	0.00000
D_3	-3.1106070***	0.05578124	-55.76439	0.00000
D_4	-3.2084815***	0.05500865	-58.32685	0.00000
D_5	-3.3524203***	0.05421850	-61.83167	0.00000
D_6	-3.3130495***	0.05356828	-61.84723	0.00000
D_7	-3.2457796***	0.05556379	-58.41537	0.00000
D_8	-3.0744087***	0.05289643	-58.12129	0.00000
D_9	-2.7747794***	0.05505828	-50.39713	0.00000
D_{10}	-2.1800401***	0.05398401	-40.38307	0.00000
D_{11}	-1.4612783***	0.05144925	-28.40233	0.00000
D_{12}	-1.1064438***	0.05242335	-21.10594	0.00000
D_{13}	-0.8769276***	0.04972381	-17.63597	0.00000
D_{14}	-0.9780153***	0.05222572	-18.72670	0.00000
D_{15}	-1.0402743***	0.04984343	-20.87084	0.00000
D_{16}	-1.1333055***	0.05940050	-19.07906	0.00000

Table A.7: Chi-squared test for the joint significance of coefficients of the Pure Diurnal Dummy Variable Model 2.52 with HC_3 standard error estimators for logarithm of squared residuals from ARMA(2,1) model 3.63 for deseasonalized EUR/GBP 5-minute log returns during active trading hours. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Observations	31,817
Mean coefficient estimate $\bar{\beta} = \frac{\sum_{i=1}^{16} \beta_i}{16}$	-2.276906
Chi-squared test statistic $\chi^2_{st} = \sum_{i=1}^{16} \left(\frac{\hat{\beta}_i - \bar{\beta}}{s_i} \right)^2$	5,233.297
Critical value $\chi^2_{15}(0.05)$	24.995790

As the logarithm of squared deterministic seasonal volatility component has been estimated $\widehat{\log(S_t^{\epsilon 2})}$ (see Table A.6), an estimate of the squared deterministic seasonal volatility component $S_t^{\epsilon 2}$ is derived using the Formula 2.55. Next, the estimate for the deseasonalized residual ϵ_t^* is derived using Equation 2.56 and its empirical distribution is studied (refer to Figure A.6). It is worth noting that the deseasonalized residual has an approximate mean of zero and a standard deviation of around one. The deseasonalized residual displays a slight positive skewness along with significant kurtosis. Lastly, the Jarque-Bera test statistic's p-value is lower than any reasonable significance level. Therefore, it is not appropriate to assume that the deseasonalized residual follows a normal distribution.



Series: Deseasonalized residual
Observations 31,817

Mean	4.57e-05
Median	0.003565
Maximum	43.52997
Minimum	-34.03965
Std. Dev.	1.000016
Skewness	2.593473
Kurtosis	336.7733
Jarque-Bera	1.48e+08
Probability	0.000000

Figure A.6: Empirical histogram and descriptive statistics for fitted deseasonalized residual $\widehat{\epsilon_t^*}$. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Figures A.7, A.8, A.9, and A.10 depict Quantile-Quantile (QQ) plots illustrating the comparison between empirical quantiles of the fitted deseasonalized residual $\widehat{\epsilon_t^*}$ and the quantiles of the (skewed) generalized error distribution as well as the (skewed) Student-t distribution with 3 and 5 degrees of freedom. This specific number of degrees of freedom has been chosen to accommodate the leptokurtic nature of the data. Quantile-Quantile plots were constructed using the `fGarch` package from `R`. From the visual inspection of the Figures A.7, A.8, A.9, and A.10 it can be concluded that the distribution of the fitted deseasonalized residual $\widehat{\epsilon_t^*}$ can be adequately approximated by the standardized Student-t distribution with 3 degrees of freedom. Based on a visual examination, it appears that the standardized Student-t distribution provides a good description for most of the observed data points in the deseasonalized residual $\widehat{\epsilon_t^*}$ sample (refer to Figure A.9).

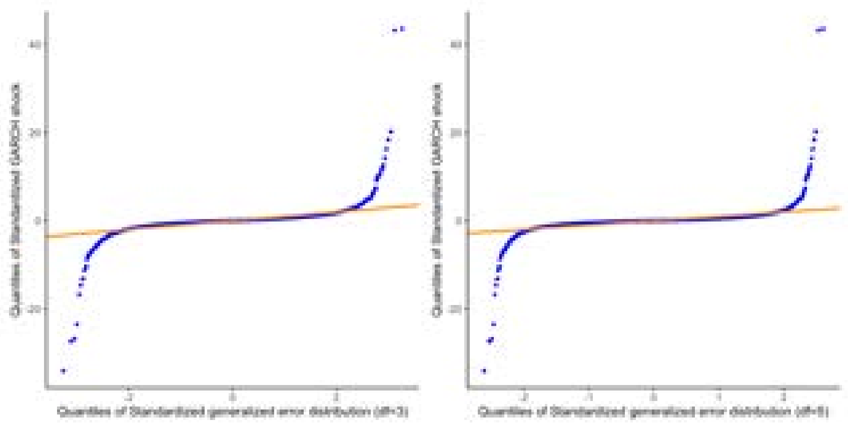


Figure A.7: QQ plots display empirical quantiles of standardized GARCH shock (fitted deseasonalized residual $\widehat{\epsilon_t^*}$) against the quantiles of generalized error distribution with 3 degrees of freedom (left hand-side) and 5 degrees of freedom (right hand-side). The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

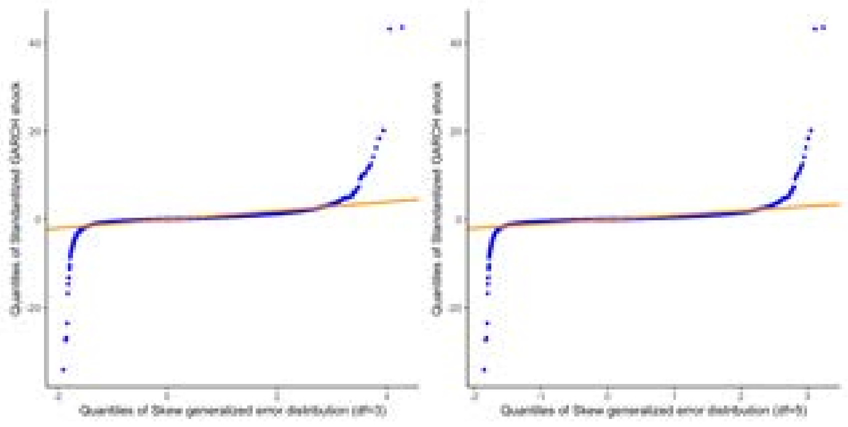


Figure A.8: QQ plots display empirical quantiles of standardized GARCH shock (fitted deseasonalized residual $\widehat{\epsilon_t^*}$) against the quantiles of skew generalized error distribution with 3 degrees of freedom (left hand-side) and 5 degrees of freedom (right hand-side). The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

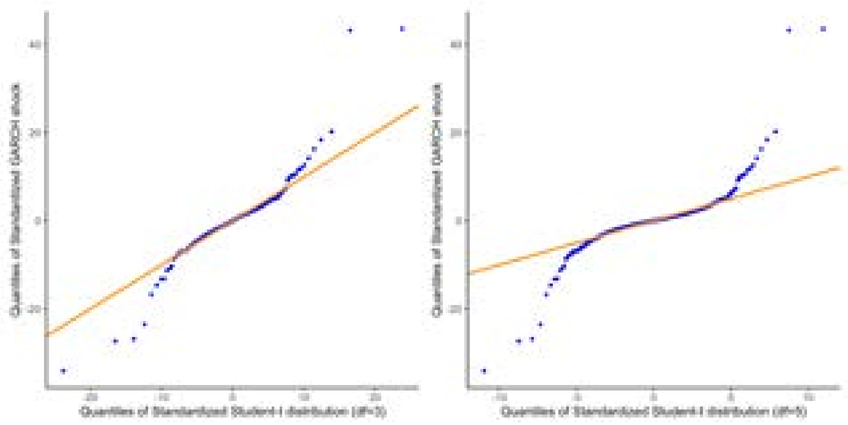


Figure A.9: QQ plots display empirical quantiles of standardized GARCH shock (fitted deseasonalized residual $\widehat{\epsilon_t^*}$) against the quantiles of standardized Student-t distribution with 3 degrees of freedom (left hand-side) and 5 degrees of freedom (right hand-side). The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

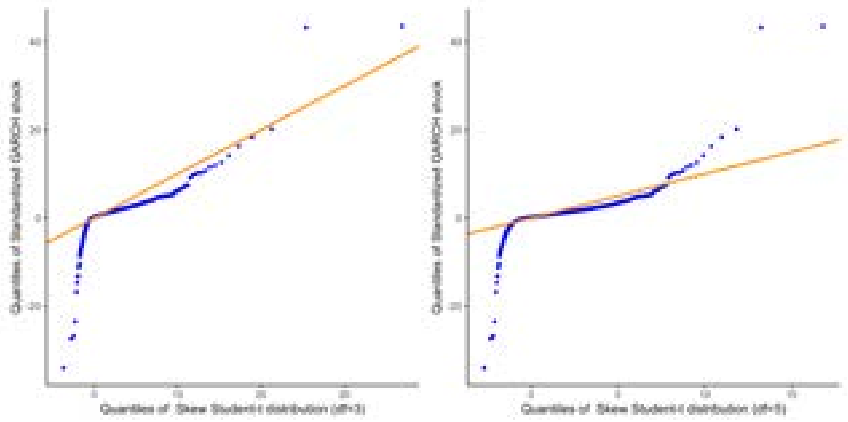


Figure A.10: QQ plots display empirical quantiles of standardized GARCH shock (fitted deseasonalized residual $\widehat{\epsilon_t^*}$) against the quantiles of skew Student-t distribution with 3 degrees of freedom (left hand-side) and 5 degrees of freedom (right hand-side). The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Two ARMA(0,0)-GARCH(1,1) models will be estimated for the deseasonalized residual ϵ_t^*, and the assumptions regarding the conditional de-meaned residual distribution for these models are as follows.

1. ε_t^* is assumed to follow a standardized Student-t distribution with 3 degrees of freedom, denoted as $t_3(0, \sigma_t^{*2})$;

2. ε_t^* is assumed to follow a standardized Student-t distribution with 5 degrees of freedom, denoted as $t_5(0, \sigma_t^{*2})$.

Both ARMA(0,0)-GARCH(1,1) have the following specifications:

$$\begin{aligned} \epsilon_t^* &= \gamma + \varepsilon_t^* \\ \varepsilon_t^* &\sim \text{i.i.d. } t(0, \sigma_t^{*2}) \\ \sigma_t^{*2} &= \omega + \alpha_1 \varepsilon_{t-1}^* + \beta_1 \sigma_{t-1}^{*2}. \end{aligned} \tag{A.7}$$

The deseasonalized residual ϵ_t^* is assumed to follow a Student-t distribution, denoted as $t_{df}(\gamma, \sigma_t^{*2})$, where df denotes the number of degrees of freedom.

The conditions that must be satisfied for a GARCH(1,1) process to be weakly stationary are as follows [37].

1. ω must be greater than 0, and both α and β should be non-negative to ensure that the variance remains greater than 0;
2. For identification purposes, β should equal 0 if α is 0;
3. To maintain weak stationarity, it is necessary that $\alpha + \beta$ be less than 1.

Note that the assumption for weak stationarity does not hold ($\alpha + \beta > 1$) for the ARMA(0,0)-GARCH(1,1) model defined in Equation A.7 with the assumption that $\varepsilon_t^* \sim t_3(0, \sigma_t^{*2})$ (refer to Table A.8). This suggests that the model is unlikely to offer a good fit to the data.

The assumptions for weak stationarity of a GARCH(1,1) process hold for the ARMA(0,0)-GARCH(1,1) model defined in Equation A.7 with the assumption that $\varepsilon_t^* \sim t_5(0, \sigma_t^{*2})$ (refer to Table A.9). Furthermore, this model displays lower AIC and BIC values compared to the previous model (refer to Tables A.8 and A.9), indicating that it is a more suitable model for the data.

Hence, the analysis is going to proceed with the ARMA(0,0)-GARCH(1,1) model defined in Equation A.7 with the assumption that $\varepsilon_t^* \sim t_5(0, \sigma_t^{*2})$.

	Estimate	Std. Error	t value	Pr(>|t|)
γ	0.00005	0.00341	0.01349	0.98924
ω	0.01878***	0.00224	8.37513	0.00000
α	0.12128***	0.00896	13.53293	0.00000
β	0.90703***	0.00689	131.66696	0.00000
$\alpha + \beta$	1.02831			
AIC	2.168601			
BIC	2.169653			

Table A.8: Estimates of the parameters for the ARMA(0,0)-GARCH(1,1) model, as represented in Equation A.7, were obtained. In this model, it is assumed that ε_t^* follows a standardized Student-t distribution with 3 degrees of freedom, which is denoted as $t_3(0, \sigma_t^{*2})$. The Equation was estimated in `R` using the *garchFit* function from the `fGarch` package. The symbol *** indicates statistical significance at 1% level. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

	Estimate	Std. Error	t value	Pr(>|t|)
γ	0.00005	0.00345	0.01323	0.98945
ω	0.01222***	0.00130	9.37336	0.00000
α	0.08107***	0.00525	15.45535	0.00000
β	0.90695***	0.00601	150.82958	0.00000
$\alpha + \beta$	0.988011			
AIC	2.142785			
BIC	2.143837			

Table A.9: Estimates of the parameters for the ARMA(0,0)-GARCH(1,1) model, as represented in Equation A.7, were obtained. In this model, it is assumed that ε_t^* follows a standardized Student-t distribution with 5 degrees of freedom, which is denoted as $t_5(0, \sigma_t^{*2})$. The Equation was estimated in `R` using the *garchFit* function from the `fGarch` package. The symbol *** indicates statistical significance at 1% level. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Deseasonalized and actual EUR/GBP conditional volatility have been estimated for the EUR/GBP 5-minute log returns using the ARMA(0,0)-GARCH(1,1) model described in Equation A.7. These estimations assume that $\varepsilon_t^* \sim t_5(0, \sigma_t^{*2})$. The outcomes are presented in Figure A.11. The distinctive U-shaped pattern is evident in the true volatility throughout each trading day, but none of this pattern is discernible in the deseasonalized volatility.

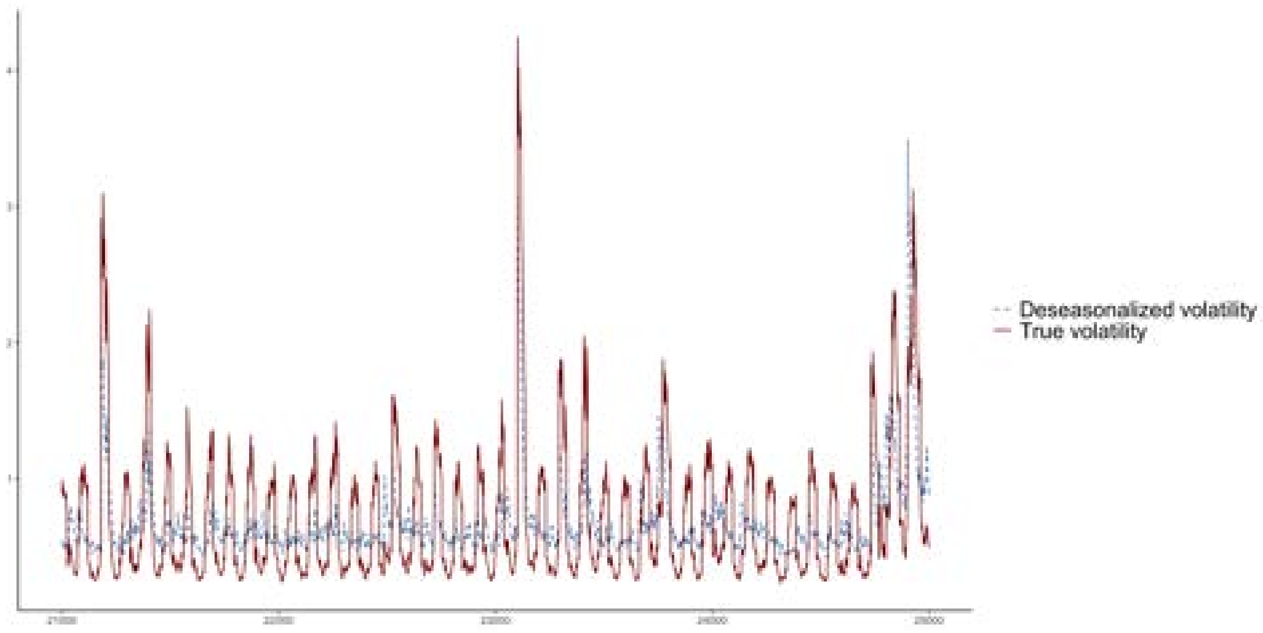


Figure A.11: Deseasonalized (dark red, solid) and true (blue, dashed) volatility estimated with ARMA(0,0)-GARCH(1,1) model defined in Equation A.7 with the assumption that $\varepsilon_t^* \sim t_5(0, \sigma_t^{*2})$. The image shows estimations for only a portion of the training data for visual clarity and convenience. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

Volatility Forecast Test for Optimality

To assess the effectiveness of the volatility forecast $\widehat{\sigma_t}$, it is essential to subject it to optimality testing. However, since the true conditional variance is inherently unobservable, a proxy is employed instead. It is widely recognized that the conditional variance proxy true fitted ARMA residual squared $\widehat{\epsilon_t^2}$ is conditionally unbiased but inherently *noisy*, exhibiting a much higher degree of variability in comparison to the true value [16]. To address this issue, an alternative Mincer-Zarnowitz regression is computed, followed by the performance of a forecast optimality test [31]. Equation A.8 is estimated (refer to Table A.10) and the null hypothesis $\mathcal{H}_0 : \beta_0 = 0$ and $\beta_1 = 1$ (forecast is optimal) is tested against the alternative $\mathcal{H}_1 : \beta_0 \neq 0$ or $\beta_1 \neq 1$.

$$\frac{\widehat{\epsilon_t^2}}{\widehat{\sigma_t^2}} = \beta_0 \frac{1}{\widehat{\sigma_t^2}} + \beta_1 + e_t \tag{A.8}$$

A Wald test for coefficient restrictions has been performed, and the null hypothesis of forecast optimality is not rejected at any reasonable significance level (with p-values for the F-statistic and chi-squared statistic at 0.8489). Therefore, there is no evidence suggesting the non-optimality of the volatility forecast obtained through the ARMA(0,0)-GARCH(1,1) model defined in Equation A.7 with the assumption that $\varepsilon_t^* \sim t_5(0, \sigma_t^{*2})$ in the case of deseasonalized EUR/GBP 5-minute log return.

	Estimate	Std. Error	t value	Pr(>|t|)
β_1	1.0400***	0.0810	12.84	0.0000
β_0	-0.0092	0.0164	-0.56	0.5754
Wald Test - Coefficient Restrictions: $\mathcal{H}_0 : \beta_0 = 0$ and $\beta_1 = 1$ against $\mathcal{H}_1 : \beta_0 \neq 0$ or $\beta_1 \neq 1$				
F-statistics	0.163864		p-value	0.8489
Chi-squared statistic	0.327728		p-value	0.8489

Table A.10: Estimated alternative Mincer-Zarnowitz regression for forecasted volatility obtained through the ARMA(0,0)-GARCH(1,1) model defined in Equation A.7 with the assumption that $\varepsilon_t^* \sim t_5(0, \sigma_t^{*2})$. The symbol *** indicates statistical significance at 1% level. The dataset encompasses the time frame from January 2022 to January 2023 and has been sourced from the Bloomberg.

A.5 R Code for Deseasonalized and True Conditional Volatility Estimation and Forecast

```r
# Load the necessary libraries
library(estimatr)
library(stringr)
library(FinTS)
library(forecast)
library(tseries)
library(moments)
library(fGarch)
library(stats)

# Read the data
data_dummy<-fread("eur_gbp_active.csv")

# Number of observations in the training set should be set to 70%
    of the entire dataset
n_train <- round(dim(data_dummy)[1]*0.7)
```

```

# Filter the training dataset
data_dummy<-data_dummy[1:n_train,]

# Standardize EUR/GBP 5-minute log return
data_dummy$Return1 <- scale(data_dummy$Return, center = TRUE,
   scale = TRUE)

# Estimate robust Pure Diurnal Dummy Variable Model with no
   intercept for standardized EUR/GBP 5-minute log return
model_return_robust<-lm_robust(Return1~D1+D2+D3+D4+D5+D6+D7+D8+D9
   +D10+D11+D12+D13+D14+D15+D16-1, data=data_dummy, se_type = "
   HC3")

# Manually conduct a Chi-squared test for the joint significance
   of coefficients of the robust Pure Diurnal Dummy Variable
   Model with no intercept for standardized EUR/GBP 5-minute log
   returns
mean_est<-mean(model_return_robust$coefficients)
stat_chi<-((model_return_robust$coefficients[1]-mean_est)/(model_
   return_robust$std.error[1]))^2
for (i in 2:16) {
  stat_chi<-stat_chi+((model_return_robust$coefficients[i]-mean_
     est)/model_return_robust$std.error[i])^2
}

# Define residuals of the robust Pure Diurnal Dummy Variable
   Model with no intercept for standardized EUR/GBP 5-minute log
   returns
residuals_model_return_robust <- data_dummy$Return1 - model_
   return_robust$fitted.values

# Estimate an ARMA(2,1) model without a mean term using the
   residuals from the robust Pure Diurnal Dummy Variable Model (
    with no intercept) for standardized 5-minute log returns of
    EUR/GBP; these residuals represent deseasonalized returns
arima12<-Arima(residuals_model_return_robust, order=c(2, 0, 1),
    include.mean = F)

# Save estimated residuals of the ARMA(2,1) model with no mean
    term in the main data source
data_dummy$Residuals <- as.vector(arima12$residuals)

# Conduct an automatic Ljung-Box-Pierce test for ARMA(2,1)
    residuals
Box.test(arima12$residuals, type = "Ljung")

# Conduct a robust regression-based test (alternative to the
    Ljung-Box-Pierce test) for ARMA(2,1) residuals with 10 lags
robust_model_LB10<-lm_robust(Residuals~lag(Residuals, 1)+lag(
    Residuals, 2)+
                        lag(Residuals, 3)+lag(Residuals, 4)+lag(
                            Residuals, 5)+
                        lag(Residuals, 6)+lag(Residuals, 7)+
                        lag(Residuals, 8)+lag(Residuals, 9)+lag(
                            Residuals, 10), data=data_dummy, se_
                            type = "HC3")

# Manually calculate a robust Chi-squared test statistic (an alt)
mean_est<-0
stat_chi<-((robust_model_LB10$coefficients[2]-mean_est)/robust_
    model_LB10$std.error[2])^2
for (i in 3:11) {
  stat_chi<-stat_chi+((robust_model_LB10$coefficients[i]-mean_est
     )/robust_model_LB10$std.error[i])^2
}

```

```
# Estimate a robust Pure Diurnal Dummy Variable Model with no
   intercept for ARMA(2,1) residuals squared
model_volatility_robust<-lm_robust(log(Residuals^2)~D1+D2+D3+D4+
   D5+D6+D7+D8+D9+D10+D11+D12+D13+D14+D15+D16-1, data=data_dummy,
    se_type = "HC3")

# Note that the estimated dependent variable in the robust Pure
   Diurnal Dummy Variable Model with no intercept for ARMA(2,1)
   residuals squared represents the deterministic component of
   conditional volatility
model_volatility_robust$fitted.values

# Save the residual from the robust Pure Diurnal Dummy Variable
   Model with no intercept for the squared residuals of the ARMA
   (2,1) model; these residuals represent the stochastic
   component of conditional volatility that will be modeled using
    the GARCH framework
model_volatility_robust_residuals <- log(data_dummy$Residuals^2)
   - model_volatility_robust$fitted.values

# Calculate unbiased estimate of the squared deterministic
   volatility component
S_squared_epsilon <- exp(model_volatility_robust$fitted.values) *
    sum(exp(model_volatility_robust_residuals)) / length(model_
   volatility_robust$fitted.values)

# Estimate deseasonalized residual
epsilon_star <- data_dummy$Residuals / sqrt(S_squared_epsilon)

# Add calculated deseasonalized residual to the main data source
data_dummy$epsilon_star <- epsilon_star

# Fit a GARCH(1,1) model with a pre-specified conditional
   distribution of deseasonalized residual denoted as epsilon_
   star; note that the choice of a Student's t-distribution with
   5 degrees of freedom was made in advance based on a prior
   statistical analysis
m2 <- garchFit(formula = ~ garch(1, 1), epsilon_star, cond.dist =
    "std", include.shape = F, shape=5, trace=F)

# Derive deseasonalized volatility using the estimated GARCH(1,1)
    model
deseasonalized_volatility <- volatility(m2)

# Derive true volatility by adjusting deseasonalized volatility
   for the identified deterministic volatility component
true_volatility <- sqrt(S_squared_epsilon) * deseasonalized_
   volatility

# Conduct a volatility forecast optimality test via alternative
   Mincer-Zarnowitz regression
mincer_zarnowitz <- as.data.frame(cbind(index, arima12$residuals
   ^2/true_volatility^2, 1/true_volatility^2))
mincer_zarnowitz_model <- lm(y~x, data=mincer_zarnowitz)

# Calculate total number of observations in the data (number of
   volatility estimates and forecasts)
nobs <- dim(data_dummy)[1]

# Calculate a deterministic component of return using the
   estimated robust Pure Diurnal Dummy Variable Model with no
   intercept for standardized EUR/GBP 5-minute log return by
   substituting the predetermined values of dummy variables
S_t_r <- as.numeric(model_return_robust$coefficients[1]) * data_
   dummy$D1 +
  as.numeric(model_return_robust$coefficients[2]) * data_dummy$D2
      +
  as.numeric(model_return_robust$coefficients[3]) * data_dummy$D3
      +
  as.numeric(model_return_robust$coefficients[4]) * data_dummy$D4
      +
  as.numeric(model_return_robust$coefficients[5]) * data_dummy$D5
      +
  as.numeric(model_return_robust$coefficients[6]) * data_dummy$D6
      +
  as.numeric(model_return_robust$coefficients[7]) * data_dummy$D7
      +
  as.numeric(model_return_robust$coefficients[8]) * data_dummy$D8
      +
  as.numeric(model_return_robust$coefficients[9]) * data_dummy$D9
      +
  as.numeric(model_return_robust$coefficients[10]) * data_dummy$
     D10 +
  as.numeric(model_return_robust$coefficients[11]) * data_dummy$
     D11 +
  as.numeric(model_return_robust$coefficients[12]) * data_dummy$
     D12 +
  as.numeric(model_return_robust$coefficients[13]) * data_dummy$
     D13 +
  as.numeric(model_return_robust$coefficients[14]) * data_dummy$
     D14 +
  as.numeric(model_return_robust$coefficients[15]) * data_dummy$
     D15 +
  as.numeric(model_return_robust$coefficients[16]) * data_dummy$
     D16

# Calculate deseasonalized return for the entire dataset by
   subtracting the deterministic component of return from the
   standardized return; note that the deterministic component has
    been predetermined
ReturnDeseasonalized <- data_dummy$ReturnStandardized - S_t_r
```

```
# Create a vector of true residuals with a length equal to the number of observations in the test set, which is the difference between the total number of observations (denoted as 'nobs') and the number of observations in the training set (denoted as 'n_train')
true_epsilon <- c(arima12$residuals, rep(NA, nobs-n_train))
for (i in (n_train+1):nobs) {
  true_epsilon[i] <- ReturnDeseasonalized[i] - arima12$coef["ar1"] * ReturnDeseasonalized[i-1] - arima12$coef["ar2"] * ReturnDeseasonalized[i-2] - arima12$coef["ma1"] * true_epsilon[i-1]
}

# Estimate a deterministic component of conditional volatility using the estimated robust Pure Diurnal Dummy Variable Model with no intercept for ARMA(2,1) residuals squared by substituting the predetermined values of the dummy variables
log_S_epsilon_squared <- as.numeric(model_volatility_robust$coefficients[1]) * data_dummy$D1 +
  as.numeric(model_volatility_robust$coefficients[2]) * data_dummy$D2 +
  as.numeric(model_volatility_robust$coefficients[3]) * data_dummy$D3 +
  as.numeric(model_volatility_robust$coefficients[4]) * data_dummy$D4 +
  as.numeric(model_volatility_robust$coefficients[5]) * data_dummy$D5 +
  as.numeric(model_volatility_robust$coefficients[6]) * data_dummy$D6 +
  as.numeric(model_volatility_robust$coefficients[7]) * data_dummy$D7 +
  as.numeric(model_volatility_robust$coefficients[8]) * data_dummy$D8 +
  as.numeric(model_volatility_robust$coefficients[9]) * data_
     dummy$D9 +
  as.numeric(model_volatility_robust$coefficients[10]) * data_
     dummy$D10 +
  as.numeric(model_volatility_robust$coefficients[11]) * data_
     dummy$D11 +
  as.numeric(model_volatility_robust$coefficients[12]) * data_
     dummy$D12

# Calculate logarithm of deseasonalized residual squared denoted
   as 'u_hat'
u_hat <- log(true_epsilon^2) - log_S_epsilon_squared

# Calculate an unbiased estimate of the deterministic component
   of conditional volatility
S_epsilon_squared <- rep(NA, nobs)
for (t in 1:nobs) {
  S_epsilon_squared[t] <- exp(log_S_epsilon_squared[t]) * sum(exp
     (u_hat[1:t])) / t
}

# Calculate deseasonalized residual
epsilon_star1 <- c(epsilon_star, rep(NA, nobs-n_train))
for (t in (n_train+1):nobs) {
  epsilon_star1[t] <- true_epsilon[t]/sqrt(S_epsilon_squared[t])
     - coef(m2)["mu"]
}

# Estimate deseasonalized conditional volatility
sigma_star_squared <- rep(NA, nobs)
sigma_star_squared[n_train]<-var(data_dummy$ReturnStandardized[1:
   n_train])
for (i in (n_train+1):nobs) {
  sigma_star_squared[i] <- coef(m2)["omega"] + coef(m2)["alpha1"]
      * (epsilon_star1[i-1])^2 + coef(m2)["beta1"] * sigma_star_
     squared[i-1]
}
deseasonalized_volatility <- c(volatility(m2), sqrt(sigma_star_
   squared[(n_train+1):nobs]))

# Compute estimated true conditional volatility
volatility_estimated <- c(volatility(m2), rep(NA, length(sqrt(
   sigma_star_squared[(n_train+1):nobs]))))

# Compute forecasted true conditional volatility
volatility_forecasted <- c(rep(NA, length(volatility(m2))), sqrt(
   sigma_star_squared[(n_train+1):nobs]))
```

A.6 Python Code for Model 1 Implementation with Lagged Returns as an Input

```
# File const_vars.py
# Import the necessary functions
# Define the constants
from torch import pow
INPUT_SIZE: int = 50
OUTPUT_SIZE: int = 10
LEARN_RATE: float = 0.01
BATCH_SIZE: int = 1
TRAIN_FRACTION = 0.7
TEST_FRACTION = 1 - TRAIN_FRACTION
MEAN = 0.0
STD = 1
ETA = 0.01
K_ETA = pow((1 - ETA / 2) / (1 - ETA), 0.5)
NUM_EPOCHS = 1

```

```
# File tools.py
# Import the necessary functions
from torch import nn, sqrt, tensor, sign, pow, square, zeros,
    float32, autograd

# Function for Xavier Initialization
def xavier_init(layer):
    if isinstance(layer, nn.Linear):
        # Calculate the variance for Xavier Initialization
        variance = 2.0 / (layer.in_features + layer.out_features)
        # Calculate the standard deviation
        std_dev = sqrt(tensor(variance))
        # Manually initialize the weights using a normal
            distribution
        layer.weight.normal_(0.0, std_dev)
        # Initialize the bias terms to 1
        layer.bias.fill_(1.0)

# Function for He Initialization
def he_init(layer):
    if isinstance(layer, nn.Linear):
        # Calculate the variance for He Initialization
        variance = 2.0 / layer.in_features
        # Calculate the standard deviation
        std_dev = sqrt(tensor(variance))
        # Manually initialize the weights using a normal
            distribution
        layer.weight.normal_(0.0, std_dev)
        # Initialize the bias terms to zero
        layer.bias.fill_(0.0)

# Function to generate a tensor composed of input vectors from
    the data stream
```

```python
def unfold_the_input_data_manually(data, input_size, n_train,
    batch_size):

    return_standardized = tensor(data['ReturnStandardized'].
        values,
                                 dtype=float32)
    # Number of time steps in input sequence (restricted by the
        input size)
    sequence_length = n_train - input_size + 1

    input_data = zeros((batch_size, sequence_length, input_size))

    for i in range(input_size, n_train + 1):
        # Select input at time step t
        current_input = return_standardized[i - input_size:i]
        input_data[0, i - input_size] = current_input

    return input_data

# Function for the gradient update
def update_gradients(df_t_dtheta_t, df_t_df_prev,
    df_prev_dtheta_prev, model):
    gradient_updates = {}

    # Iterate through the model's parameters, its gradients, and
        the previous gradients
    for name, grad, (name, prev_grad) in zip(
            model.named_parameters(), df_t_dtheta_t,
            df_prev_dtheta_prev.items()):
        gradient_updates[name] = grad + df_t_df_prev * prev_grad

    return gradient_updates

```

```python
# Function to calculate trading model output and realized trading
    return
def model_output_and_gradients(data1, data2, i, f_prev, model):
    current_input1 = data1[:, i]
    f_t = model(current_input1, f_prev)
    df_t_dtheta_t = autograd.grad(f_t, model.parameters(),
                                  retain_graph=True)
    df_t_df_prev = autograd.grad(f_t, f_prev, retain_graph=True)

    delta_t = (data2.iloc[i]['High'] - data2.iloc[i]['Low']) / 2
    r_t = (sign(f_prev) * data2.iloc[i]['ReturnStandardized'] -
           abs(sign(f_t) - sign(f_prev)) * delta_t)

    return f_t, df_t_dtheta_t, df_t_df_prev, delta_t, r_t

# Function to calculate Differential Sharpe ratio and Sharpe
   ratio
def one_time_step(r_t, a_prev, b_prev, eta, k_eta):
    delta_a_t = r_t - a_prev
    delta_b_t = square(r_t) - b_prev

    a_t = a_prev + eta * delta_a_t
    b_t = b_prev + eta * delta_b_t

    # Calculate Differential Sharpe ratio
    differential_sharpe_ratio = (
            (b_prev * delta_a_t - 0.5 * a_prev * delta_b_t) /
            pow(b_prev - square(a_prev), 1.5))
    # Calculate Sharpe ratio
    sharpe_ratio = a_t / (k_eta * pow(b_t - square(a_t), 0.5))
    return a_t, b_t, differential_sharpe_ratio, sharpe_ratio

# Function to calculate partial derivatives for parameter
   optimization
```

```
def optimization_model(i, a_prev, b_prev, f_t, f_prev,
                       delta_t, r_t, train_data):
    dd_t_dr_t = (b_prev - a_prev * r_t) / pow(b_prev - square(
       a_prev), 1.5)
    dr_t_df_t = - sign(f_t - f_prev) * delta_t
    dr_t_df_prev = train_data.iloc[i + 1][
                        'ReturnStandardized'] + sign(f_t - f_prev
                          ) * delta_t
    return dd_t_dr_t, dr_t_df_t, dr_t_df_prev

# File model.py
# Import the necessary functions
from torch import nn, cat, sign
# Define a trading model
class DNN(nn.Module):
    def __init__(self, input_size, output_size):
        super(DNN, self).__init__()

        self.linear1 = nn.Linear(input_size, output_size)
        self.activation1 = nn.LeakyReLU()

    # Forward pass
    def forward(self, x):

        x = self.linear1(x)
        x = self.activation1(x)

        return x

class RecurrentRLTradingSystem(nn.Module):
    def __init__(self, input_size, output_size):
        super(RecurrentRLTradingSystem, self).__init__()
        self.dnn = DNN(input_size, output_size)

        self.linear_rnn = nn.Linear(output_size + 1, 1)
        self.activation_rnn = nn.Tanh()

    # Forward pass
    def forward(self, x, f_prev):
        x = self.dnn(x)

        # Combine current output with lagged input
        recurrent_input = cat((x, f_prev), dim=1)

        # Recurrent layer
        x = self.linear_rnn(recurrent_input)
        f_t = self.activation_rnn(x)

        # Apply sign finction only in evaluation mode
        if not self.training:
            f_t = sign(f_t)

        # Return the output
        return f_t

# File train_and_test.py
# Import the trading model, functions from the supporting files,
   necessary functions and libraries
from model import RecurrentRLTradingSystem
from numpy import round as np_round
from const_vars import *
from tools import (
    xavier_init, unfold_the_input_data_manually, update_gradients
        ,
    model_output_and_gradients, one_time_step, he_init,
       optimization_model
)
import torch
import psutil

# Set the seed
torch.manual_seed(42)

# Initialize the trading model
trading_model = RecurrentRLTradingSystem(INPUT_SIZE, OUTPUT_SIZE)

# Read the data
df = pd_csv('eur_gbp_data.csv')

# Calculate number of observations in training and test datasets
n_train = int(np_round(df.shape[0] * TRAIN_FRACTION))
n_test = int(np_round(df.shape[0] * TEST_FRACTION))

# Specify the training dataset
train_data = df[:n_train]
# Specify the test dataset
test_data = df.iloc[n_train:, ]

# Initialize model's parameters
with torch.no_grad():
    # Initialize the model's layer with Xavier Initialization
    he_init(trading_model.dnn.linear1)
    # Initialize the model's layer with He Initialization
    xavier_init(trading_model.linear_rnn)

# Create a dictionary with the model's parameter initializations
custom_update = {}
for name, layer in trading_model.named_modules():
    if isinstance(layer, torch.nn.Linear):
        custom_update[name + '.weight'] = layer.weight.clone()
        custom_update[name + '.bias'] = layer.bias.clone()

```

```
# Transform the input data
input_data_train = unfold_the_input_data_manually(train_data,
   INPUT_SIZE,
                                                                  n_train,
                                                                      BATCH_SIZE)
input_data_test = unfold_the_input_data_manually(test_data,
   INPUT_SIZE,
                                                                 n_test + n_test,
                                                                      BATCH_SIZE)

# Create empty lists
differential_sharpe_ratios = []
sharpe_ratios = []
trading_signals = []
trading_returns = []

# Initialize the model's previous state
f_prev1 = torch.normal(MEAN, STD, size=(1, 1))
f_prev1.requires_grad = True

# Initialize the gradient of previous model output with respect
   to the model parameters
df_prev_dtheta_prev1 = {}
for name, param in trading_model.named_parameters():
    df_prev_dtheta_prev1[name] = torch.zeros_like(param)

# Initialize parameters necessary for Differential Sharpe ratio
   and Sharpe ratio calculations
a_prev1 = torch.normal(MEAN, STD, size=(1,))
b_prev1 = torch.normal(MEAN, STD, size=(1,))

# Training the model
def train_model(input_data, num_epochs, learn_rate, eta, k_eta):
    # Set the model to training mode
    trading_model.train()

    for epoch in range(num_epochs):
        # Advance by 2 time steps in every iteration
        for t in range(0, input_data.size()[1] - 1, 2):

            # First time step
            f_t1, df_t_dtheta_t1, df_t_df_prev1, delta_t1, r_t1 \
                = model_output_and_gradients(
                    input_data, train_data,
                    t, f_prev1, trading_model)

            a_t1, b_t1, differential_sharpe_ratio_t1,
               sharpe_ratio_t1 \
                = one_time_step(r_t1, a_prev1, b_prev1, eta,
                     k_eta)

            gradient_updates_t1 = update_gradients(df_t_dtheta_t1
               ,
                                                     df_t_df_prev1
                                                        [0][0],
                                                     df_prev_dtheta_prev1
                                                        ,
                                                     trading_model)

            # Update states for next iteration
            a_prev2 = a_t1
            b_prev2 = b_t1
            f_prev2 = f_t1
            df_prev_dtheta_prev2 = gradient_updates_t1

            # Second time step
            f_t2, df_t_dtheta_t2, df_t_df_prev2, delta_t2, r_t2 \
                = model_output_and_gradients(input_data, train_data,
                                                 t, f_prev2, trading_model)

            a_t2, b_t2, differential_sharpe_ratio_t2, sharpe_ratio_t2 \
                = one_time_step(r_t2, a_prev2, b_prev2, eta, k_eta)

            gradient_updates_t2 = update_gradients(
                df_t_dtheta_t2,
                df_t_df_prev2[0][0],
                df_prev_dtheta_prev2,
                trading_model
            )

            dd_t_dr_t, dr_t_df_t, dr_t_df_prev = optimization_model(
                t, a_prev2, b_prev2, f_t2, f_prev2, delta_t2, r_t2, train_data)

            with torch.no_grad():
                for (name, parameter), grad1, grad2 in zip(
                        trading_model.named_parameters(), gradient_updates_t1,
                        gradient_updates_t2):
                    # Custom update formula
                    custom_update[name] += \
                        learn_rate * dd_t_dr_t.squeeze() \
                        * (
                                dr_t_df_t.squeeze() * gradient_updates_t2[name]
                                + dr_t_df_prev.squeeze()
```

```
                                * gradient_updates_t1[name]
                            )
                        parameter.data = custom_update[name]

            # Update states for next iteration
            a_prev1 = a_t1
            b_prev1 = b_t1
            f_prev1 = f_t2
            df_prev_dtheta_prev1 = gradient_updates_t2

            # Append values after the last epoch
            if epoch == num_epochs - 1:
                # Store results
                differential_sharpe_ratios.append(
                   differential_sharpe_ratio_t1.item())
                differential_sharpe_ratios.append(
                   differential_sharpe_ratio_t2.item())
                sharpe_ratios.append(sharpe_ratio_t1.item())
                sharpe_ratios.append(sharpe_ratio_t2.item())
                trading_signals.append(f_t1.item())
                trading_signals.append(f_t2.item())
                trading_returns.append(r_t1.item())
                trading_returns.append(r_t2.item())

# Use the function for training the model
differential_sharpe_ratios_train, sharpe_ratios_train,
   trading_signals_train, trading_returns_train = train_model(
   input_data_train, NUM_EPOCHS, LEARN_RATE, ETA, K_ETA)

# Save the model
torch.save(model.state_dict(), 'model_saved.pth')
# Load model for future use
model = RecurrentRLTradingSystem(input_size, output_size)
model.load_state_dict(torch.load('model_saved.pth'))
```

```

# Create a function for testing the model
def test_model(input_data, eta, k_eta):
    # Set the model to evaluation mode
    model.eval()

    # Initialize parameters
    f_prev = torch.normal(mean, std, size = (1,1))
    a_prev = torch.normal(mean, std, size = (1,))
    b_prev = torch.normal(mean, std, size = (1,))

    trading_signals = []
    sharpe_ratios = []
    differential_sharpe_ratios = []
    trading_returns = []

    with torch.no_grad():  # Ensure no gradients are calculated
        for t in range(input_data.size()[1]):

            current_input = input_data[:, t]
            f_t = model(current_input, f_prev)

            delta_t = (test_data.iloc[t]['High'] - test_data.iloc
               [t]['Low']) / 2
            r_t = f_prev * test_data.iloc[t]['ReturnStandardized'
               ] - torch.abs(f_t - f_prev) * delta_t
            delta_a_t = R_t - a_prev
            delta_b_t = torch.square(R_t) - b_prev
            a_t = a_prev + eta * delta_a_t
            b_t = b_prev + eta * delta_b_t
            # Calculate Differential Sharpe ratio
            differential_sharpe_ratio_t = ( b_prev * delta_a_t -
               0.5 * a_prev * delta_b_t) / torch.pow(b_prev -
               torch.square(a_prev), 1.5 )
            # Calculate Sharpe ratio
            sharpe_ratio_t = a_t / ( k_eta * torch.pow(b_t -
               torch.square(a_t), 0.5 ) )

            # Store results
        differential_sharpe_ratios.append(
           differential_sharpe_ratio_t.item())
            sharpe_ratios.append(sharpe_ratio_t.item())
            trading_signals.append(f_t.item())
            trading_returns.append(r_t.item())

            # Update states for next iteration
            f_prev = f_t
            a_prev = a_t
            b_prev = b_t

    return differential_sharpe_ratios, sharpe_ratios,
       trading_signals, trading_returns

# Use the function for testing the model
differential_sharpe_ratios_test, sharpe_ratios_test,
   trading_signals_test, trading_returns_test = test_model(
   input_data_test, ETA, K_ETA)
```

REFERENCES

[1] Anuj Agarwal. *High Frequency Trading: Evolution and the Future*. 2012. URL: https://www.capgemini.com/wp-content/uploads/2017/07/High_Frequency_Trading__Evolution_and_the_Future.pdf.

[2] Ran El-Yaniv Allan Borodin. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998. URL: http://gen.lib.rus.ec/book/index.php?md5=66226e259275bd6d7ccf3d3fe112eee1.

[3] Torben G. Andersen et al. "The distribution of realized stock return volatility". In: *Journal of Financial Economics* 61.1 (2001), pp. 43–76. DOI: https://doi.org/10.1016/S0304-405X(01)00055-1. URL: https://www.sciencedirect.com/science/article/pii/S0304405X01000551.

[4] Roy E. Bailey. *The Economics of Financial Markets*. Cambridge Books 9780521612807. Cambridge University Press, July 2005. URL: https://ideas.repec.org/b/cup/cbooks/9780521612807.html.

[5] Richard T. Baillie and Tim Bollerslev. "Intra-Day and Inter-Market Volatility in Foreign Exchange Rates". In: *The Review of Economic Studies* 58.3 (May 1991), pp. 565–585. DOI: 10.2307/2298012. URL: https://doi.org/10.2307/2298012.

[6] Richard T. Baillie and Tim Bollerslev. "The Message in Daily Exchange Rates: A Conditional-Variance Tale". In: *Journal of Business Economic Statistics* 7.3 (1989), pp. 297–305. URL: http://www.jstor.org/stable/1391527 (visited on 08/27/2023).

[7] Tim Bollerslev. "Generalized autoregressive conditional heteroskedasticity". In: *Journal of Econometrics* 31.3 (1986), pp. 307–327. DOI: https://doi.org/10.1016/0304-4076(86)90063-1. URL: https://www.sciencedirect.com/science/article/pii/0304407686900631.

[8] George. E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[9] Emma Brunskill. *Introduction to Reinforcement Learning*. https://web.stanford.edu/class/cs234/slides/lecture1.pdf. [Online; accessed 04-July-2023]. 2023.

[10] Ernie Chan. *Quantitative Trading: How to Build Your Own Algorithmic Trading Business*. Wiley Trading. Wiley, 2009. URL: https://books.google.ru/books?id=NZlV0M5Ije4C.

[11] Rama Cont. "Empirical properties of asset returns: stylized facts and statistical issues". In: *Quantitative Finance* 1.2 (2001), pp. 223–236. DOI: 10.1080/713665670. URL: https://doi.org/10.1080/713665670.

[12] Michel Dacorogna et al. *An Introduction to High-Frequency Finance*. Academic Press, New York, Jan. 2001.

[13] Michael Dempster and V. Leemans. "An automated FX trading system using adaptive reinforcement learning". In: *Expert Systems with Applications* 30 (Apr. 2006), pp. 543–552. DOI: 10.1016/j.eswa.2005.10.012.

[14] Yue Deng et al. "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading". In: *IEEE Transactions on Neural Networks and Learning Systems* 28 (2017), pp. 653–664.

[15] David A. Dickey and Wayne A. Fuller. "Distribution of the Estimators for Autoregressive Time Series With a Unit Root". In: *Journal of the American Statistical Association* 74.366 (1979), pp. 427–431. URL: http://www.jstor.org/stable/2286348 (visited on 08/15/2023).

[16] Francis X. Diebold. *Elements of Forecasting*. HG - Cycles and Forecasting Series. South-Western College Pub., 1998. URL: https://books.google.de/books?id=65e3AAAAIAAJ.

[17] Robert F. Engle. "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation". In: *Econometrica* 50.4 (1982), pp. 987–1007. URL: http://www.jstor.org/stable/1912773 (visited on 08/17/2023).

[18] Official Journal of the European Union. *REGULATION (EU) No 236/2012 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 14 March 2012*. https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2012:086:0001:0024:en:PDF.

[19] Eugene F. Fama. "Efficient Capital Markets: II". In: *The Journal of Finance* 46.5 (1991), pp. 1575–1617. DOI: https://doi.org/10.1111/j.1540-6261.1991.tb04636.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1991.tb04636.x.

[20] C. Gold. "FX trading via recurrent reinforcement learning". In: *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.* 2003, pp. 363–370. DOI: 10.1109/CIFER.2003.1196283.

[21] Clive W. J. Granger. "Seasonality: Causation, Interpretation and Implications". In: *Essays in Econometrics: Collected Papers of Clive W. J. Granger*. Ed. by Eric Ghysels, Norman R. Swanson, and Mark W.Editors Watson. Vol. 1. Econometric Society Monographs. Cambridge University Press, 2001, pp. 121–146. DOI: 10.1017/CBO9780511753961.005.

[22] Peter R. Hansen and Asger Lunde. "A forecast comparison of volatility models: does anything beat a GARCH(1,1)?" In: *Journal of Applied Econometrics* 20.7 (2005), pp. 873–889. DOI: https://doi.org/10.1002/jae.800. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jae.800.

[23] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015.

[24] Sebastian Jaimungal José Penalva Álvaro Cartea. *Algorithmic and high-frequency trading*. West Nyack: Cambridge University Press, 2015.

[25] Mehran Katanforoosh and Andrew Kunin. *Initializing Neural Networks*. deeplearning.ai AI Notes. 2019. URL: https://www.deeplearning.ai/ai-notes/initialization/.

[26] G. M. Ljung and G. E. P. Box. "On a Measure of Lack of Fit in Time Series Models". In: *Biometrika* 65.2 (1978), pp. 297–303. ISSN: 00063444. URL: http://www.jstor.org/stable/2335207 (visited on 08/29/2023).

[27] James G. MacKinnon and Halbert White. "Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties". In: *Journal of Econometrics* 29.3 (1985), pp. 305–325. DOI: https://doi.org/10.1016/0304-4076(85)90158-7. URL: https://www.sciencedirect.com/science/article/pii/0304407685901587.

[28] Burton G. Malkiel. “The Efficient Market Hypothesis and Its Critics”. In: *Journal of Economic Perspectives* 17.1 (Mar. 2003), pp. 59–82. DOI: 10.1257/089533003321164958. URL: https://www.aeaweb.org/articles?id=10.1257/089533003321164958.

[29] Harry Markowitz. “PORTFOLIO SELECTION”. In: *Journal of Finance* 7.1 (1952), pp. 77–91. URL: https://EconPapers.repec.org/RePEc:bla:jfinan:v:7:y:1952:i:1:p:77-91.

[30] Patrick McGuire. *2022 BIS Triennial Central Bank Survey of turnover in foreign exchange markets*. Bank for International Settlements, 2022.

[31] Jacob Mincer and Victor Zarnowitz. “The Evaluation of Economic Forecasts”. In: *Economic Forecasts and Expectations: Analysis of Forecasting Behavior and Performance*. National Bureau of Economic Research, Inc, 1969, pp. 3–46. URL: https://EconPapers.repec.org/RePEc:nbr:nberch:1214.

[32] Marvin Minsky. “Steps toward Artificial Intelligence”. In: *Computers Thought*. Cambridge, MA, USA: MIT Press, 1995, pp. 406–450.

[33] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518 (Feb. 2015), pp. 529–33. DOI: 10.1038/nature14236.

[34] John E. Moody and Matthew Saffell. “Learning to trade via direct reinforcement”. In: *IEEE Transactions on Neural Networks* 12.4 (2001), pp. 875–889. DOI: 10.1109/72.935097.

[35] John E. Moody and Lizhong Wu. “Optimization of trading systems and portfolios”. In: *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)* (1997), pp. 300–307.

[36] John E. Moody et al. “Performance functions and reinforcement learning for trading systems and portfolios”. In: *Journal of Forecasting* 17 (1998), pp. 441–470.

[37] A. Patton. *Quantitative finance subject guide for University of London International Programmes*. 2015.

[38] Peter C. B. Phillips and Pierre Perron. “Testing for a Unit Root in Time Series Regression”. In: *Biometrika* 75.2 (1988), pp. 335–346. URL: http://www.jstor.org/stable/2336182 (visited on 08/15/2023).

[39] A. Rao and T. Jelvis. *Foundations of Reinforcement Learning with Applications in Finance*. CRC Press, 2022. URL: https://books.google.de/books?id=n%5C_-VEAAAQBAJ.

[40] William F. Sharpe. "Mutual Fund Performance". In: *The Journal of Business* 39.1 (1966), pp. 119–138. URL: http://www.jstor.org/stable/2351741 (visited on 07/17/2023).

[41] William F. Sharpe. "The Sharpe Ratio". In: vol. 21. 1. 1994, pp. 49–58. DOI: 10.3905/jpm.1994.409501.

[42] Kostas Skouras and Richard Chandler. *Forecasting Course notes, 2017–2018*. 2017–2018.

[43] Elliot Smith. "Pound tanking, massive tax cuts and talk of emergency hikes. Here's what's going on in the UK". In: *CNBC* (Sept. 26, 2022). URL: https://www.cnbc.com/2022/09/26/pound-tanking-massive-tax-cuts-and-talk-of-emergency-rate-hikes.html (visited on 09/26/2022).

[44] Elliot Smith. "The crisis is over for the British pound, but analysts see further weakness ahead". In: *CNBC* (Nov. 3, 2022). URL: https://www.cnbc.com/2022/11/03/british-pound-crisis-is-over-but-analysts-see-further-weakness-ahead.html (visited on 11/03/2022).

[45] Shuo Sun, Rundong Wang, and Bo An. *Reinforcement Learning for Quantitative Trading*. Papers 2109.13851. arXiv.org, Sept. 2021. URL: https://ideas.repec.org/p/arx/papers/2109.13851.html.

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

[47] *Temporary prohibition of short selling - 17 March*. https://www.fca.org.uk/news/news-stories/temporary-prohibition-short-selling-0. Accessed: 2023-07-24.

[48] HM Treasury. *Short Selling Regulation Review*. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/1130913/SSR_CfE_-_Official_Publication__FINAL_.pdf. December 2022.

[49] Ruey S. Tsay. *Analysis of financial time series*. eng. 3rd ed. Wiley series in probability and statistics. Hoboken, N.J.: Wiley, 2010.

[50] Staff of the U.S. Securities and Exchange Commission. *Staff Report on Algorithmic Trading in U.S. Capital Markets*. U.S. Securities and Exchange Commission, 2020. URL: https://www.sec.gov/files/algo_trading_report_2020.pdf.

[51] Paul Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78 (Nov. 1990), pp. 1550–1560. DOI: 10.1109/5.58337.

[52] Jin Zhang and Dietmar Maringer. "Using a Genetic Algorithm to Improve Recurrent Reinforcement Learning for Equity Trading". In: *Computational Economics* 47.4 (Apr. 2016), pp. 551–567. DOI: 10.1007/s10614-015-9490-y. URL: https://ideas.repec.org/a/kap/compec/v47y2016i4d10.1007_s10614-015-9490-y.html.