Feature Engineering in Reinforcement Learning for Algorithmic Trading

Investigating the Effects of State Representation on Trading Agent Performance in the Forex Market

Finn van Oosterhout[1]

Supervisors: Neil Yorke-Smith[1]**, Antonis Papapantoleon**[1]**, Amin Kolarijani**[2]

[1]**EEMCS, Delft University of Technology, The Netherlands**
[2]**Mechanical Engineering, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Finn van Oosterhout
Final project course: CSE3000 Research Project
Thesis committee: Neil Yorke-Smith, Antonis Papapantoleon, Amin Kolarijani, Julia Olkhovskaya

Abstract

This study explores how different features impact a Reinforcement Learning agent's performance in forex trading. Using a Deep Q-Network (DQN) agent and EUR/USD data from 2022-2024, we found that performance is highly sensitive to the information provided. Key findings show that for feature types like momentum and volatility, a single indicator outperformed a combination of them, as the latter tended to introduce noise. Including information about the agent's own status, such as its current trade duration, was beneficial. Counterintuitively, providing more historical data consistently worsened performance, leading to overfitting where the agent memorized training data rather than learning general strategies. The main conclusion is that creating an effective state representation is a trade-off; the complexity of the input data must match the learning algorithm's ability to process it without overfitting.

1 Introduction

The use of artificial intelligence (AI) in financial trading has become increasingly popular in recent years [8; 5]. Specifically, reinforcement learning (RL) has shown a promising future in the field of autonomous trading agents [11; 5; 22; 17]. These agents, in theory, can learn optimal trading strategies through interaction with the market environment [5]. Despite this potential of RL in finance, its adoption remained limited as of 2022. A report by the Dutch Authority for the Financial markets (AFM) highlighted perceived risks associated with RL-based trading agents as a primary barrier, even as firms acknowledged its future potential [5]. This gap between the potential and actual adoption of RL in algorithmic trading shows further research is required into this field. Additionally, reviews of the field, such as that by [22], explicitly call for more comparative studies, particularly examining how different RL agents perform compared to each other under similar conditions. Therefore, this work aims to provide insights for developing more effective, robust, and safer AI-based trading agents by investigating the impact of different state representation designs on agent performance.

1.1 Research Questions

This paper seeks to answer the question: **"What are the impacts of different state representation designs on the performance of an RL-based low-frequency forex trading agent?"** To answer this, the following sub-questions are asked:

1. (SQ1) How does the inclusion or exclusion of specific features (Technical Indicators, Trading Agent Data) affect agent performance?
2. (SQ2) How does the quantity of information impact agent performance?

1.2 Structure of Research

This paper is structured as follows: Section 1 introduces the research, outlining the motivation, the research questions, and related works. Section 2 provides background information on trading and reinforcement learning. Section 3 details the methodology, including the formal problem description, dataset, and the reinforcement learning environment and agent. Section 4 outlines the experimental setup, detailing the data splits, performance metrics, and the protocol for the two phases of experiments. Section 5 presents the results of these experiments. Section 6 discusses the implications of these results, followed by a section on responsible research in Section 7. Finally, Section 8 provides the conclusion and suggestions for future work.

1.3 Related works

To our knowledge, there have not yet been any works published on specifically comparing different state representation designs under similar conditions in RL-based algorithmic trading. However, there are many other works around the general use of reinforcement learning and artificial intelligence in algorithmic trading, containing relevant information for the goal of this study. Below we highlight some of the related works.

Reviews & Surveys

A number of surveys [5; 22; 10; 17; 3] offer insights into the current state of the field of RL in algorithmic trading, discussing various RL approaches, different state features, and comparative performance analysis.

Applications

Several works [12; 14; 26; 11; 9; 30; 20] present a variety of approaches of feature selection and data representation, ranging from the use of raw market data and common technical indicators to more advanced techniques such as dimensionality reduction, clustering, autoencoders, and learned feature weighting. Below are highlighted some of the key insights.

Feature Engineering is Central:

- Many successful approaches do not just use raw prices but incorporate a diverse set of features, including technical indicators [14; 9; 20], derivatives of raw market data like log returns or price evolution [11; 26; 12], and trading agent data like current position, cash, or holding duration [14; 26; 11; 30]. Reference [9] specifically explores novel features from wave theory and K-line patterns, suggesting that domain-specific or advanced pattern recognition can enhance state input.

Data Representation Methods are Diverse and Impactful:

- Normalization (e.g., Z-score, division by last price, log returns) is a common and crucial preprocessing step [12; 11; 20].
- Dimensionality Reduction and Robust Feature Extraction: Techniques like Stacked Denoising Autoencoders (SDAEs in [14]) and offline State Representation Learning (SRL) involving PCA and clustering [20] are used to manage large feature sets, reduce noise, and extract more robust representations.

- Temporal Feature Extraction: LSTMs are frequently employed within the RL architecture to process sequences of state features and capture temporal dependencies [14; 20; 12].
- Specific Encodings: Sinusoidal encoding for time data and one-hot encoding for categorical agent states (like position) are used in forex trading agents [11].
- Discretization: For certain RL algorithms like Q-learning in noisy markets, creating a finite state space through discretizations and categorization of variables (including agent-specific data like holding time) is a viable strategy [30].

Quantity and Quality of Information Matter:

- Reference [9] advocates for multi-category features to provide richer market information.
- Reference [26] shows a practical example of starting with a broad set of potential observations and then defining a more focused, reduced observation space for the actual agent.

Context and Market-Specific Design:

- References [11; 30] provide forex-specific state design examples, including intermarket features and multi-timeframe analysis.
- Reference [20] demonstrates that their SRL (State Representation Learning) components (dimensionality reduction, clustering, gate structure for feature weighting) each contribute to improved performance, emphasizing the importance of a tailored state representation pipeline.

2 Background

This section provides an overview of trading concepts, market analysis techniques, and reinforcement learning (RL), laying the foundation for our research.

2.1 Trading

This research focuses on trading in the foreign exchange (forex) market, a decentralized global marketplace where currencies like the Euro and U.S. Dollar are exchanged [4]. Algorithmic trading automates the process of executing asset exchanges, using computer programs that act based on predefined criteria such as timing, price movements, market data, or other quantitative inputs [24]. Over time these exchanges of currencies can result in profits or losses.

Exchange rates

In the forex market, currencies are exchanged in pairs, meaning that to buy one currency, another has to be sold. Exchange rates represent the value of the quote currency in terms of the base currency, showing the amount of the quote currency that is received by selling 1 unit of the base currency [15].

Currency prices are typically quoted with a bid price and an ask price. The bid price is the rate at which a dealer (or the market) is willing to buy a currency, while the ask price is the rate at which they are willing to sell that same currency [21].

The difference between the ask price and the bid price is known as the 'bid-ask spread' or simply the 'spread'. A narrower spread is generally more advantageous for traders as it indicates lower transaction costs [21]. The spread commonly increases during periods of low liquidity, and decreases during periods of high liquidity [26].

Price movements in financial markets are often described using the term 'ticks'. A tick represents the smallest possible price change for a financial instrument [6].

Tick data refers to the most granular level of market information, recording every individual price change (tick) and the time it occurred. This high-resolution data is crucial for many algorithmic trading strategies that rely on capturing small, short-term price fluctuations.

Japanese Candlesticks

This research focuses on low-frequency trading, where decisions are made at discrete time steps (e.g. 5-minute intervals). The raw tick data is aggregated into Japanese candlesticks, which represent the Open, High, Low, and Close (OHLC) prices for a given period [5]. Figure 1 shows how candlesticks are formed, and highlights the difference between an upward/bullish and downward/bearish candle.

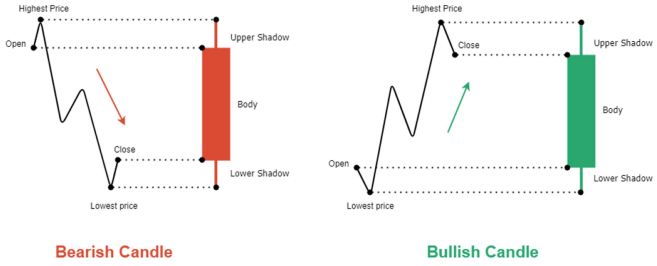

Figure 1: Formation of candlesticks. Figure reproduced from [5].

Volume

Trading volume refers to the total quantity of an asset traded during a specific period. It serves as an indicator of market activity and liquidity. Because of the decentralized nature of forex markets, absolute volume data is often unavailable. Instead, tick volume is used as a proxy. Tick volume measures the number of ticks that occur within a specific period. Although it doesn't capture total trade sizes, tick volume strongly correlates with actual trading volume [16; 28].

Analysis

Financial market analysis is broadly categorized into fundamental analysis, which examines external economic factors, and technical analysis, which this work employs. Technical analysis uses historical price and volume data to estimate future price movements, often using technical indicators [5]. The different types of analysis are shown in Figure 2. Technical indicators are categorized into three main types: trend indicators, momentum indicators, and volatility indicators. Trend indicators are designed to identify the general direction of price in recent history. Momentum indicators, also known as oscillators, measure the speed and strength of price movements. They can help identify whether an asset is gaining or losing strength [5]. Volatility indicators measure the rate and magnitude of price fluctuations. They help traders

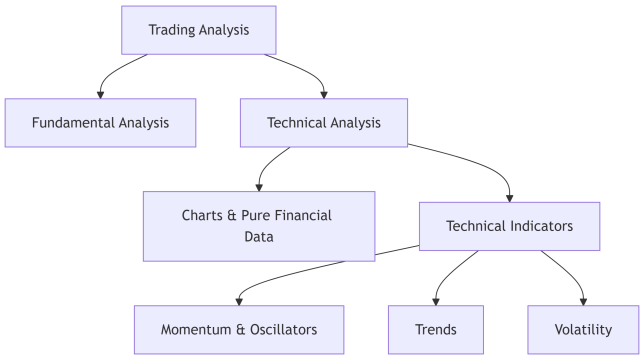

Figure 2: Financial Market Analysis Types. Figure reproduced and adjusted from [5].

understand the degree of market risk and potential for significant price swings. High volatility indicates large and rapid price changes, while low volatility suggests a more stable market. Complementing these categories are volume-based indicators. These indicators incorporate trading volume into their calculations to provide insight into the market participation and conviction driving a price trend. For instance, a price trend accompanied by high volume is generally considered more significant than one with low volume [28].

Costs of Trading

Trading profitability is directly impacted by two main costs: transaction costs and slippage. Transaction costs are explicit fees for executing trades. Transaction costs are subdivided into two main components: trading platform commissions, and spread costs (crossing the spread). Commissions are fees charged by a trading platform for the execution of trades, typically a percentage on the trading volume. As an example, `FOREX.com` charges a commission of 0.005% for users with a "`RAW Spread account`" [1]. Spread costs arise from the gap between the bid and ask prices, which incurs a minor loss upon the execution of a trade [8; 11; 9; 30]. Slippage is an implicit cost defined as the difference between the expected and actual execution price of a trade. This typically occurs due to market volatility during execution delays or the price pressure from the trade's own volume (market impact) [26; 20].

Trade Types

A trade is defined as a continuous period of holding a long or short position [13]:

- A Long position is initiated by buying an asset, speculating that its price will rise, allowing it to be sold later for a profit.
- A Short position involves selling an asset (typically borrowed), speculating that its price will fall, allowing it to be bought back at a lower price for a profit.

2.2 Reinforcement Learning

Reinforcement Learning is a machine learning paradigm for sequential decision-making. An agent learns to operate within an environment by performing actions and receiving feedback in the form of numerical rewards. The agent's objective is to develop a policy, a strategy for choosing actions, that maximizes the cumulative reward over time, effectively learning optimal behavior through trial-and-error [10; 17; 20; 11]. Figure 3 shows a visual representation of this process.

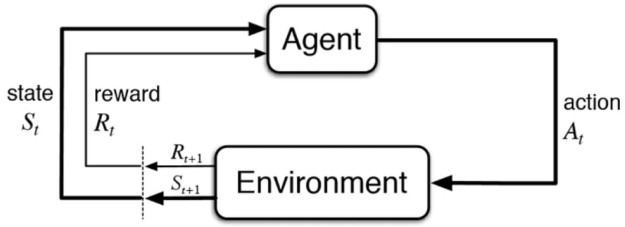

Figure 3: Process of reinforcement learning. Figure reproduced from [25].

RL algorithms can be classified in several ways [22; 3; 10]:

- **By what they learn:** Algorithms can be critic-only (learning a value function to estimate the expected future reward of states), actor-only (learning a policy directly), or a hybrid actor-critic method (learning both simultaneously).
- **By how they learn:** On-policy algorithms improve the same policy that is used to make decisions. In contrast, off-policy algorithms can learn an optimal policy using data generated by a different, exploratory policy, making them more sample-efficient by allowing the reuse of past experiences.

Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) combines RL with deep learning, using neural networks (e.g. MLPs, CNNs, RNNs, LSTMs) as function approximators to handle complex, high-dimensional state spaces like those found in financial data [22; 17; 14; 9].

Markov Decision Process

A Markov Decision Process (MDP) is a mathematical framework for modeling sequential decision problems where outcomes are partly random and partly controlled by a decision-maker. Most reinforcement learning problems are framed as MDPs. An MDP is formally defined by a tuple (S,A,T,R,γ) [10; 17; 20], which contains:

- S: A set of all possible states the environment can be in.
- A: A set of all possible actions the agent can take.
- $T(s'|s,a)$: The transition probability of moving to state s' from state s after taking action a.
- $R(s,a)$: The reward received after taking action a in state s.
- $\gamma \in [0,1]$: A discount factor that balances the importance of immediate versus future rewards.

A key assumption of the MDP framework is the Markov property: the probability of transitioning to the next state depends only on the current state and the action taken, and is

independent of the history of states and actions that led to the current state [17; 14].

Partially Observable Markov Decision Process

In many real-world applications, including finance, the agent cannot fully perceive the environment's true state, violating the core assumption of an MDP. A Partially Observable Markov Decision Process (POMDP) extends the MDP for these scenarios where the agent cannot observe the true state. A POMDP adds a set of observations Ω and an observation function $O(o|s', a)$, meaning the agent receives an observation o that is a probabilistic function of the true underlying state s. This makes the POMDP a more realistic framework for algorithmic trading [11; 22; 3; 12; 26; 14].

3 Methods

3.1 Formal Problem Description

We model the low-frequency forex algorithmic trading task as a portfolio optimization problem, framed as a POMDP. The portfolio consists of two assets: a base currency (cash) and a quote currency (shares). The agent's goal is to learn a policy $\pi(o_t)$ that maps observations to actions to maximize the expected cumulative discounted reward, which correlates to maximizing the total final value of the portfolio. The POMDP is defined by the tuple (S,A,T,R,Ω,O,γ):

- **State Space (S):** The state $s_t \in S$ consists of the portfolio state p_t, and the unobserved market state m_t. The portfolio state p_t is defined by its cash (C_t, base currency) and shares (H_t, quote currency). A negative H_t denotes a short position. Let P_t represent the exchange rate of the currency pair at time t. We consider both the bid price ($P_{bid,t}$) and the ask price ($P_{ask,t}$). The portfolio's equity (E_t) liquidation value, is calculated using Equation 1.

$$E_t = \begin{cases} C_t + H_t * P_{bid,t} & \text{if } H_t \geq 0 \\ C_t + H_t * P_{ask,t} & \text{if } H_t < 0 \end{cases} \quad (1)$$

- **Action Space (A):** The action space A is a finite set of discrete actions the agent can take, $A = \{a_0, a_1, \ldots, a_{N-1}\}$. Each action maps to a different configuration of the assets in the portfolio, as seen in Section 3.4
- **Reward Function (R):** The reward function $R(s_t, a_t)$ provides the immediate feedback signal to the agent. The reward is a function of the change in the agent's portfolio equity, guiding it toward profitable behavior. The specific mathematical formulation of the reward is a critical design choice detailed in Section 3.7.
- **Transition Function (T):** The market state transition is not modeled but follows historical data, as is standard in backtesting environments (Section 3.4).
- **Observation Space (Ω) and Function (O):** The observation vector o_t is composed of features derived from historical market data and the agent's portfolio states (Section 3.3).

3.2 Dataset

The data source for this research is Dukascopy. We gathered tick-by-tick historical data for the EUR/USD currency pair, covering the full calendar years of 2022, 2023, and 2024. EUR/USD was chosen for its major use in previous research [5] and being one of the most traded currency pairs in general [15].

Resampling

The tick data is resampled into Japanese candlesticks using the resample functionality within the Pandas library [19]. The resampling is done both on the `bid` and `ask` prices to generate Open, High, Low, and Close (OHLC) values for each. Regarding trading volume, Dukascopy provides `bid_volume` and `ask_volume` fields with their tick data, but their precise definition is not clearly documented. Therefore, we adopt the count of ticks occurring within each candle as a proxy for the volume. To account for non-trading hours, the resulting data is filtered on having a tick count greater than 0.

The final resampled dataset consists of candlestick data at the chosen granularity for the EUR/USD pair, with distinct OHLC values for bid and ask prices, and a volume figure representing the number of ticks per candle.

Analysis on the data in 15-minute japanese candlesticks can be seen in Appendix A.

3.3 Feature Engineering

Due to data availability and time constraints, this research focuses solely on features directly calculated from the market and agent data. The five types of features we recognize are: time, trend, momentum, volatility, and agent features. The trend, momentum, and volatility features are defined the same as the technical indicators discussed in Section 2. We define time features as features that give a quantitative indication of time, for example, the hour of the day. Agent features are calculated based on the agent's current state, including its cash, amount of shares, and previous actions.

Features

Within these categories, we recognize the following features:

- **Time:** `SIN(interval)`, `COS(interval)`, and `LIN(interval)`.
- **Trend:** `Parabolic SAR`, `Volume Weighted Average Price (VWAP)`, and `Kaufman's Adaptive Moving Average (KAMA)`.
- **Momentum:** `MACD`, `MFI`, and `CCI`.
- **Volatility:** `Bollinger Bands`, `Average True Ranges (ATR)`, and `Ease of Movement (EOM)`.
- **Agent:** `CURRENT TRADE DURATION`, and `CURRENT EXPOSURE`.

Calculations of these features and our normalization, scaling, and transformation techniques can be found in our GitHub repository [2].

Core Feature Set

To establish a consistent baseline for both hyperparameter optimization and our primary experiments, seen in Section 4, we define a "Core Feature Set". This set is designed to be a balanced representation, including one feature from each of our defined categories. The trend, momentum and volatility features have been provided with an additional historical lookback of 4 timesteps. All features have been normalized to approximately fall into the range of -1 and 1. The core feature set consists of the following: `SIN(24H), Parabolic SAR, MACD, Bollinger Bands, Current Exposure`. This feature set forms the basis of the hyperparameter optimization detailed in Appendix C, and serves as a starting point for the experiments described in Section 4.

3.4 Simulated Forex Environment

We developed a custom backtesting environment, built on the Gymnasium API [27], to implement the POMDP described in Section 3.1. The agent's discrete action $i \in \{0, 1, \ldots, N-1\}$ is mapped to a desired target exposure (χ), which represents the intended ratio of the shares' value to the total portfolio equity. The target exposure for action i is calculated as $\chi(i) = -1 + \frac{2i}{N-1}$, creating a uniform spread of positions from $[-1, 1]$, or full short to full long. The agent's current exposure is calculated using the agent's portfolio state: $\chi_t = (E_t - C_t)/E_t$ At each step, the environment executes the necessary trades at the open price of the current candle to align the current exposure with the target, disregarding small differences to avoid a large number of transactions. This process of buying and selling accounts for the bid-ask spread and commissions. Following the transaction, a reward and next observation are calculated and returned to the agent. An episode concludes if the agent's equity is depleted or the end of the dataset has been reached, after which it is reset. Visual representations of this environment are given in Figures 6 and 7 in Appendix B.

As we are simulating the highly complex forex markets, some assumptions and simplifications have been made:

- **Zero Market Impact Hypothesis:** we assume that the agent's trades have no impact on the price of the traded asset.
- **Action Execution:** we assume actions get executed at exactly the open prices of the current candlestick, without any slippage.

3.5 RL Algorithm

We use the Deep Q-Network (DQN) algorithm, implemented via the Stable-Baselines3 library [23], due to its demonstrated success and frequent use in previous studies [5; 10; 26]. DQN is an off-policy, critic-only algorithm that uses a deep neural network to approximate the optimal action-value function, $Q^*(s, a)$. The Q-network takes as input an observation, and outputs the Q-values of each possible action. The agent's policy is ϵ-greedy, meaning it selects the action with the highest Q-value most of the time but chooses a random action with a probability of ϵ to ensure exploration. To stabilize training, DQN utilizes an experience replay buffer, which stores past experiences that are sampled in mini-batches to update the Q-network. This process breaks the correlation between consecutive samples. Additionally, a separate, periodically updated target network is used to provide a stable target for calculating the loss, further improving learning stability [23; 18; 17; 14; 26; 11]. The DQN algorithm involves a number of key hyperparameters that were tuned during our research (Section 4.4). A detailed list and explanation of these parameters can be found in Appendix C.3.

3.6 Action Space

We use a discrete action space of N=3, where the agent chooses from the actions $\{0, 1, 2\}$. These actions map to a target portfolio exposure, χ, of -1 (full short), 0 (neutral), and +1 (full long), respectively.

3.7 Reward Function

We define the reward, R_t, as the percentage change in the portfolio's equity from the moment just before an action is taken at time step t, to the beginning of the next time step, $t + 1$.

Let $E_{pre-action,t}$ be the equity calculated using the open price at time t ($P_{open,t}$), before executing the action given by the agent for timestep t. The reward is calculated as $R_t = (\frac{E_{pre-action,t+1}}{E_{pre-action,t}} - 1) * 100$.

Using pre-action equity at both t and $t + 1$ ensures that the reward reflects the full impact of price movement and transaction costs over that interval. This approach encourages the agent to select actions that maximize portfolio growth in a way that aligns with actual trading outcomes.

4 Experimental Setup

4.1 Data Splits

To ensure rigorous evaluation and to prevent data leakage, the dataset covering 2022-2024 is divided chronologically into two equally sized blocks. The first block (Jan 2022 - June 2023) was used exclusively for the multi-phase hyperparameter optimization detailed in Section 4.4. The second block is used in the main experiments, and split chronologically into a training set (70%), a validation set (15%) for model selection, and a final, untouched evaluation set (15%) for final evaluation.

4.2 Performance Metrics

We evaluate each agent model using standard financial metrics to assess profitability, risk, and risk-adjusted performance [5; 22; 10; 14; 20]:

- **Annualized Sharpe Ratio (ASR):** Our primary metric, measuring risk-adjusted return. It is the sample Sharpe Ratio of portfolio returns annualized for comparison, calculated as $ASR = \frac{\overline{r}}{\sigma_r} * \sqrt{N}$. Here $\overline{r}$ and σ_r are

the mean and standard deviation of periodic portfolio returns ($r_t = \frac{E_t}{E_{t-1}} - 1$). N is the annualization factor, representing the number of trading periods in a year.

- **Total Profit and Loss (PnL %):** The overall strategy profitability, calculated as the percentage change in equity from start to finish.
- **Maximum Drawdown (Max DD):** A measure of downside risk, representing the largest peak-to-through decline in portfolio equity, calculated as $MDD = \min_t(E_t - \underset{\tau \leq t}{max} E_\tau)$.
- **Number of Trades**: The total count of executed trades, offering insight into trading frequency and transaction costs. A trade is a continuous period of holding a long or short position.

4.3 Benchmarks

We compare the results against several benchmarks to assess whether the created RL-based strategies are viable. We use four benchmarks:

1. **Long Only:** This model performs one long trade across the entire dataset
2. **Short Only:** This model performs one short trade across the entire dataset
3. **Trend Following:** The Trend Following benchmark is a model designed to follow market trends by making trades based on moving average crossovers [26]. It utilizes two Kaufman's Adaptive Moving Average (KAMA) indicators, each configured with a different time window. A long trade is initiated when the shorter-term moving average crosses above the longer-term one, and conversely, a short trade is opened when it crosses below.
4. **Perfect Trading:** This model takes near perfect actions at each timestep, calculated beforehand using a dynamic programming approach.

4.4 Hyperparameter Optimization

Prior to the main experiments, the hyperparameters were determined through a systematic optimization process. This was conducted in phases using a dedicated, non-overlapping portion of the dataset to prevent data leakage and ensure that the final model was not overfit to the test data. The comprehensive details of this phased approach are described in Appendix C. This process led to the selection of the parameters that can be seen in Table 1.

4.5 Experimental Protocol

Our experimental protocol is structured in two distinct phases, each designed to systematically address one of the sub-questions outlined in Section 1. This approach allows us to first isolate the impact of different types of information before investigating the effect of the quantity of that information. All experiments are conducted using the final model architecture and hyperparameters determined during the optimization phase, highlighted in Table 1.

Table 1: Experimentation Parameters

Configuration Name	**Value**
Learning Rate	5×10^{-5}
γ	0.95
Buffer Size	$60,000$
Batch Size	512
Tau	0.0025
Train Frequency (Steps)	64
Target Update Interval	$2,500$
Exploration Fraction	0.40
Exploration Initial Epsilon	1.0
Exploration Final Epsilon	0.02
DQN Policy	MlpPolicy
Network Architecture	[32,16]
Activation Function	Leaky ReLU
Optimizer	Adam
Data Granularity	1-hour
Initial Capital	$10,000$
Transaction Cost	0.005%

To ensure robustness and reproducibility of our results, all experiments described below are executed on five distinct random seeds: $42, 43, 44, 45, 46$. The performance metrics reported in Section 5 are the mean and standard deviation across these five runs, providing insight into the stability of our findings.

Furthermore, we employ a systematic model selection strategy to balance performance with generalization and mitigate overfitting. During the training of an agent for a specific experimental setup, the model's state is saved at the end of every episode. After the training process concludes, each of these saved models is evaluated on both the training set and the validation set. The definitive model for that experiment is then selected based on a score calculated as: $score = ASR_{val} - |ASR_{val} - ASR_{train}|$. ASR_{val} and ASR_{train} represent the primary performance metric (annualized sharpe ratio) on the validation and training sets, respectively. This selection criterion was designed to reward high performance on unseen validation data, while also penalizing models that exhibit a large performance gap between training and validation data, which indicates overfitting or "getting lucky". The single model with the highest score is chosen as the representative for that experiment and is the only one used for the final evaluation on the out-of-sample evaluation dataset.

Phase 1: Impact of Feature Categories (SQ1)

The primary objective of the first phase is to answer our first sub-question (SQ1): "How does the inclusion or exclusion of specific features (Technical Indicators, Trading Agent Data) affects agent performance?"

To achieve this, we conduct a series of experiments based on the feature categories defined in Section 3.3: Time, Trend, Momentum, Volatility, and Agent features. For each category, we systematically alter the core feature set's features (Section 3.3), within that specific category. The experiments for each category generally include:

- No features included.

- A single non-volume-based feature.
- A single volume-based feature.
- A model combining one non-volume-based and one volume-based feature.
- A model using all available defined features within that category.

This structured approach allows us to attribute changes in performance directly to the presence or absence of specific feature types. A comprehensive list of the 28 unique experimental configurations for this phase is detailed in Table 7 in Appendix D.1.

Phase 2: Impact of Information Quantity (SQ2)

The second phase addresses SQ2: "How does the quantity of information impact agent performance?". This phase investigates the impact of historical information depth across three predefined feature configurations of increasing complexity:

- **Small:** The core feature set as defined in Section 3.3.
- **Medium:** An enhanced set combining two features for each category, specifically the 'COMBO' configurations from Phase 1, defined in Appendix D.1.
- **Large:** The complete set of all engineered features investigated in this research.

For each of these three configurations, we systematically vary the historical lookback window that is applied only to the Trend, Momentum, and Volatility features. We test lookback values of 0, 1, 2, 4, 8, 16, and 32. A lookback of k means the agent receives data from the current time step t and the previous k steps, for a total of $k+1$ datapoints for the applied feature. This allows us to assess performance sensitivity to historical context across different levels of feature complexity. A comprehensive list of the 21 unique experimental configuration for this phase is detailed in Table 8 in Appendix D.2.

5 Results

This section presents the empirical results from our two-phase experimental protocol. All metrics are reported as the mean and standard deviation across five distinct random seeds to ensure a robust evaluation. We first establish benchmark performance, followed by a general overview of agent performance, and then detail the results corresponding to our two sub-questions on feature categories and information quantity. Additionally, Appendix E shows the performance graphs of each model saved during training, evaluated on the training and validation datasets.

5.1 Benchmark Results

Table 2 reports the performance of the four benchmark strategies on the evaluation dataset. The passive 'Long Only' strategy yielded a Sharpe ratio of -2.9, and the 'Short Only' strategy achieved a positive Sharpe ratio of 2.9. The rule-based 'Trend Following' strategy produced a Sharpe ratio of 0.41. The 'Perfect Trading' model, representing a theoretical upper bound for performance, obtained a Sharpe ratio of 56.1.

Table 2: Results of the benchmarks

Configuration	Sharpe	PnL (%)	# Trades	Max DD
Long Only	-2.9	-5.3	1	549
Short Only	2.9	5.3	1	207
Trend Following	0.41	0.68	43	231.5
Perfect Trading	56.1	121.3	585	25.0

5.2 General Results

A primary observation across the majority of experiments is a negative mean Sharpe ratio, as shown in Tables 3 and 4. The experiments consistently exhibit high variance, with standard deviations that are often close to or exceed the mean values, indicating a significant sensitivity to the random seed used for agent initialization.

The highest mean Sharpe ratio observed across all configurations was 1.64 (`S2_MEDIUM_0`) , and the lowest was -2.60 (`S2_LARGE_32`). All experiments performed better than the 'Long Only' benchmark, and some surpassed the 'Trend Following' strategy. None of the experiments' mean performance surpassed the 'Short Only' strategy's performance. A recurring pattern in the results is that simply increasing the number of features does not consistently lead to improved agent performance.

A notable observation is the performance variance among experiments that used the same set of features but in a different input order. Due to the experimental design, the baseline 'Core Feature Set' was constructed under different experiment IDs (e.g., `S1_MO_NV, S1_VO_NV, and S2_SMALL_4`). This resulted in a different ordering of features in the agent's input vector. This seemingly minor difference led to disparate performance outcomes: `S1_MO_NV` reported a mean Sharpe ratio of 0.58, while `S1_VO_NV` and S2_SMALL_4 both reported -0.37. This discrepancy shows the training process's high sensitivity to input structure and initial network weights.

5.3 Phase 1: Impact of Feature Categories (SQ1)

This phase investigated how the inclusion or exclusion of specific feature categories affects agent performance. The complete results are summarized in Table 3.

Time Features

The configuration with a single linear encoding of the 24-hour cycle (`S1_TM_L24`) resulted in the highest mean Sharpe ratio in this category at 1.12. In contrast, using all available time features (`S1_TM_ALL`) led to a negative mean Sharpe ratio of -1.02.

Trend Features

For this category, the agent configured with all available trend features (`S1_TR_ALL`) produced the best result within the group, with a mean Sharpe ratio of -0.46. Excluding trend features entirely (`S1_TR_NONE`) or using a limited combination (`S1_TR_COMBO`) resulted in lower Sharpe ratios of -0.99 and -1.33, respectively.

Momentum and Volatility Features

For both the momentum and volatility categories, including a single feature yielded superior performance compared to

other configurations within the group. For momentum, a single volume-based feature (S1_MO_V) achieved a Sharpe ratio of 0.70. For volatility, a single non-volume-based feature (S1_VO_NV) performed best with a Sharpe ratio of -0.37. In both categories, models using no features, two features, or all available features significantly underperformed compared to the single-feature configurations.

Agent Features

Providing the agent with information about its own state resulted in a performance increase. The configuration with no agent features (S1_AG_NONE) registered the lowest Sharpe ratio in this group (-2.48). Including the agent's current trade duration (S1_AG_DT) resulted in the group's best performance, with a mean Sharpe ratio of -0.03.

Table 3: Results of the SQ1 experiments displayed as a mean with the standard deviation in brackets

Configuration	Sharpe	PnL (%)	# Trades	Max DD
S1_TM_NONE	-0.73(2.22)	-0.97(3.55)	103.00(40.15)	-407.88(163.89)
S1_TM_L24	1.12(2.27)	1.83(3.88)	90.8(15.97)	-290.25(83.13)
S1_TM_L24L7	-0.83(2.41)	-0.83(3.90)	119.00(67.46)	-405.98(201.08)
S1_TM_S24	0.29(2.19)	0.56(3.91)	94.80(31.56)	-328.91(128.67)
S1_TM_SC24	-1.11(2.46)	-1.99(4.11)	180.40(111.58)	-469.26(212.62)
S1_TM_SC24SC7	0.04(2.81)	0.33(4.81)	74.00(56.44)	-338.51(167.63)
S1_TM_COMBO	-0.48(3.02)	-0.57(5.08)	81.8(72.16)	-399.29(204.29)
S1_TM_ALL	-1.02(2.29)	-1.46(3.52)	164.00(46.84)	-410.72(168.39)
S1_TR_NONE	-0.99(1.21)	-1.60(2.00)	75.20(49.97)	-306.55(124.05)
S1_TR_NV	-0.82(2.66)	-1.63(4.46)	97.00(45.55)	-438.55(245.32)
S1_TR_V	-1.11(2.18)	-1.93(3.76)	107.80(47.08)	-450.82(190.94)
S1_TR_COMBO	-1.33(2.42)	-2.23(4.14)	76.8(45.15)	-407.28(209.17)
S1_TR_ALL	-0.46(1.98)	-0.79(3.31)	87.00(45.76)	-358.98(122.176)
S1_MO_NONE	-1.62(1.43)	-2.87(2.43)	24.4(46.80)	-336.06(267.44)
S1_MO_NV	0.58(2.42)	0.95(4.23)	123.00(67.10)	-369.84(216.90)
S1_MO_V	0.70(2.24)	1.00(3.85)	57.8(44.63)	-295.10(111.71)
S1_MO_COMBO	-2.15(1.17)	-3.47(1.67)	124.60(42.21)	-516.09(112.08)
S1_MO_ALL	-2.44(3.26)	-3.98(5.34)	134.80(22.78)	-591.01(353.47)
S1_VO_NONE	-1.85(2.35)	-3.21(4.01)	84.00(63.83)	-558.47(136.99)
S1_VO_NV	-0.37(2.50)	-0.36(4.06)	103.20(37.45)	-355.71(202.66)
S1_VO_V	-0.09(2.94)	0.59(4.32)	60.00(25.65)	-317.73(178.15)
S1_VO_COMBO	-1.73(1.26)	-2.46(1.96)	106.4(77.30)	-407.50(107.09)
S1_VO_ALL	-2.25(0.89)	-3.09(1.42)	128.8(51.61)	-410.86(124.47)
S1_AG_NONE	-2.48(1.19)	-3.68(1.85)	165.40(94.91)	-466.25(103.13)
S1_AG_CE	-0.38(2.51)	-0.37(4.06)	103.20(37.46)	-355.71(202.67)
S1_AG_DT	-0.03(1.82)	0.15(3.04)	151.80(62.83)	-321.65(152.94)
S1_AG_ALL	-0.94 (2.26)	-1.45(3.60)	127.60(43.97)	-352.90(134.61)
S1_COMBO_COMBO	-0.57(2.56)	-0.67(4.33)	90.00(61.20)	-367.89(131.34)
S1_COMBO_ALL	-0.66(2.58)	-0.74((4.14)	157.8 (46.2)	-370.40(142.94)

5.4 Phase 2: Impact of Information Quantity (SQ2)

Phase 2 evaluated the impact of historical information depth by varying lookback windows across feature sets of increasing size. The results are presented in Table 4.

Small Feature Set

Using the core 'Small' feature set, performance peaked with short lookback windows. A lookback of 2 (S2_SMALL_2) achieved the highest mean Sharpe ratio for this set at 1.21. Further increasing the lookback window generally corresponded with a decrease in performance.

Medium Feature Set

The 'Medium' feature set achieved the highest Sharpe ratio across all experiments (1.64) with no historical lookback (S2_MEDIUM_0). The introduction of any historical context ($k > 0$) resulted in a significant performance degradation, with all other configurations yielding negative mean Sharpe ratios.

Large Feature Set

Similar to the other sets, the 'Large' feature set performed best with no historical lookback (S2_LARGE_0), yielding a mean Sharpe ratio of 0.39. Increasing the lookback depth generally resulted in progressively lower performance, culminating in a Sharpe ratio of -2.60 for the largest lookback of 32.

Table 4: Results of the sq2 experiments displayed as a mean with the standard deviation in brackets

Configuration	Sharpe	PnL (%)	# Trades	Max DD
S2_SMALL_0	0.75(2.47)	1.34(4.22)	55.40(32.37)	-316.42(182.73)
S2_SMALL_1	0.41(1.64)	1.19(2.28)	51.00(54.61)	-217.41(121.40)
S2_SMALL_2	1.21(2.09)	2.06(3.59)	62.60(76.84)	-278.76(142.31)
S2_SMALL_4	-0.37(2.50)	-0.36(4.06)	103.20(37.45)	-355.71(202.66)
S2_SMALL_8	-0.01(1.68)	-0.03(2.67)	82.40(48.95)	-242.03(172.78)
S2_SMALL_16	0.28(2.81)	0.52(4.54)	100.00(78.40)	-317.41(155.12)
S2_SMALL_32	-0.68(2.13)	-0.98(3.35)	96.00(57.37)	-355.83(146.50)
S2_MEDIUM_0	1.64(1.51)	1.36(2.56)	40.00(40.16)	-182.70(160.16)
S2_MEDIUM_1	-1.84(2.08)	-2.92(3.46)	87.00(47.09)	-446.66(154.92)
S2_MEDIUM_2	-0.33(1.72)	-0.59(2.68)	113.80(61.80)	-294.75(201.17)
S2_MEDIUM_4	-0.56(2.55)	-0.66(4.33)	90.00(61.20)	-367.89(131.33)
S2_MEDIUM_8	-1.43(2.77)	-1.84(4.56)	120.40(70.34)	-416.66(234.75)
S2_MEDIUM_16	-1.54(1.65)	-2.59(2.76)	132.30(70.98)	-426.61(144.59)
S2_MEDIUM_32	-1.30(2.05)	-1.71(2.86)	149.20(81.19)	-382.90(127.66)
S2_LARGE_0	0.39(2.25)	0.46(3.78)	81.6(32.11)	-279.33(163.24)
S2_LARGE_1	-0.81(1.13)	-1.19(1.83)	90.4(55.6)	-300.19(166.35)
S2_LARGE_2	-2.43(2.25)	-3.41(3.30)	148.00(8.48)	-509.99(182.96)
S2_LARGE_4	-0.65(2.57)	-0.74(4.14)	157.80(46.20)	-370.39(142.94)
S2_LARGE_8	-0.91(2.75)	-1.86(3.94)	151.40(50.61)	-429.81(247.66)
S2_LARGE_16	-1.67(2.90)	-2.45(4.84)	97.60(87.77)	-410.28(274.03)
S2_LARGE_32	-2.60(1.17)	-3.51(2.29)	99.60(61.91)	-479.10(298.07)

6 Discussion

The results reveal significant challenges in applying a DQN agent to low-frequency forex trading. Across most experimental configurations, the agent achieved negative mean Sharpe ratios with high standard deviations, which often exceeded the mean values themselves (Tables 3 and 4). This indicates that despite finding profitable strategies on certain random seeds, the agent's performance lacked stability and was not reliably positive.

This high variance points to a significant sensitivity to initial conditions, such as network weights and exploration trajectory. For instance, merely altering the input order of an identical feature set resulted in mean Sharpe ratios of 0.58 and -0.37, respectively. Such instability occurred despite a prolonged exploration phase (40% of training) and complicates efforts to draw firm conclusions about the value of specific features.

6.1 Phase 1: The Impact of Feature Categories

The first phase of experiments, designed to answer SQ1, revealed that the relationship between feature types and agent performance is not straightforward. The results suggest a balancing act between providing sufficient information and introducing excessive complexity or noise.

For the Momentum and Volatility categories, a "less is more" approach appeared most effective. Configurations with

a single, well-chosen feature (S1_MO_V and S1_VO_NV) significantly outperformed those with no features, a combination of two, or all available features. This suggests that adding indicators introduced confounding noise rather than valuable signals, making it harder for the agent to learn a coherent policy.

Conversely, for Trend features, the agent performed best when provided with all available indicators (S1_TR_ALL), achieving the highest Sharpe ratio within its group. This may indicate that the different trend indicators (Parabolic SAR, VWAP, KAMA) captured complementary aspects of the market's direction, providing a richer and more robust view that the agent could leverage.

Another notable finding was the impact of Agent features. Providing the agent with information about its own state, specifically the duration of its current trade (S1_AG_DT), led to an improvement in performance compared to when this context was absent (S1_AG_NONE). This is logical within a POMDP framework, as this feature provides the agent with a crucial piece of its own history, helping it to contextualize its current position. For instance, it could learn to exit a trade that has been held for an extended period without generating profit. The Current Exposure feature was less impactful, likely because its value was a direct consequence of the action space ($\{-1, 0, 1\}$), offering little new information beyond what the agent had just decided to do.

6.2 Phase 2: The Peril of Too Much Information

The second phase, addressing SQ2, investigated the impact of historical information depth. The results were remarkably consistent and counter-intuitive: across all three feature set sizes (Small, Medium, and Large), performance peaked with little to no historical lookback ($k = 0$ or $k = 2$). For the Medium and Large sets, the introduction of any significant historical lookback led to a consistent and severe degradation in performance.

The most probable explanation for this phenomenon is overfitting. As the dimensionality of the observation space increases (either through more features in Phase 1 or more historical steps in Phase 2), the DQN agent, with its complex neural network, has a greater capacity to "memorize" specific patterns in the training data rather than learn generalizable strategies. This is corroborated by the training graphs in Appendix E, where models with larger feature sets often converge to higher Sharpe ratios on the training set but fail to translate this performance to the validation or evaluation sets. The agent is incentivized to find complex, high-dimensional correlations that, despite yielding high rewards during training, are merely noise when tested on unseen data.

This issue also highlights a potential weakness in our model selection methodology. The criterion, $score = ASR_{val} - |ASR_{val} - ASR_{train}|$, was designed to penalize overfitting. However, the high variance in performance across episodes may have allowed "lucky" models to still achieve a high score, even if they were not truly generalized. This could have occurred simply because they performed well enough on both training and validation sets by chance. The observation that poor convergence on the training data sometimes led to better validation performance further supports this concern.

6.3 Answering the Research Questions

Overall, this research paints a picture of a delicate balancing act in designing state representations for an RL trading agent. The agent's extreme sensitivity to initial conditions and input structure highlights the need for robust experimental procedures, including the use of more random seeds and more advanced model selection techniques, to confidently isolate the effects of specific parameters.

With this crucial context in mind, we answer the research questions as follows:

- **SQ1: How does the inclusion of specific feature categories affect performance?** Within our setup, the impact was category-dependent. Providing all available Trend indicators was beneficial, whereas for Momentum and Volatility categories, a single feature yielded better results, suggesting that additional features in these groups introduced more noise than signal for the agent. Most significantly, providing the agent with its own state information, led to improvements in relative performance, highlighting the value of contextual features.
- **SQ2: How does the quantity of information impact performance?** For our agent, increasing the quantity of information, both in terms of the number of features and the depth of historical lookbacks, was consistently detrimental. Performance peaked with little to no historical context, and the introduction of lookback windows almost always resulted in significant performance degradation, which we attribute to the agent's inability to generalize from a high-dimensional state space and its tendency to overfit.

However, this should not be interpreted as a definitive statement that simpler feature sets are universally superior. It is more likely a reflection of the limitations of the chosen agent and methodology. The DQN agent's inability to effectively process a high-dimensional state space does not mean the information within it is not valuable. It is plausible that a more advanced agent architecture, perhaps with attention mechanisms to weigh feature importance, or a more sophisticated training and model selection process, could successfully extract the signal from the noise in a larger feature set and benefit from deeper historical context.

Therefore, the final takeaway is not that complex state representations should be avoided, but that the design of the state representation is closely linked with the capabilities of the learning algorithm and the robustness of the experimental procedure. The challenge lies in the co-design of these elements: developing agents and training protocols that are capable of effectively exploiting rich, complex information without succumbing to high variance and overfitting.

7 Responsible Research

This research was conducted with a strong commitment to ethical considerations, reliability, and reproducibility. We acknowledge the complexities and potential risks associated with applying advanced AI techniques to financial markets and have taken deliberate steps to address them throughout our methodology and reporting.

7.1 Ethical Considerations

The application of Reinforcement Learning (RL) in algorithmic trading raises several ethical questions, primarily centered on transparency, risk, and market fairness.

A significant concern is the "black box" nature of many RL models. This lack of transparency makes it difficult to fully understand how an agent arrives at its decisions, creating uncertainty about whether it might be exploiting data or market mechanics in unintended or unethical ways. This challenge is recognized by regulatory bodies like the Dutch Authority for the Financial Markets (AFM), which highlights the risks of market manipulation and the lack of explainability as potential barriers to the adoption of RL in trading. The AFM stipulates that if a trading algorithm is suspected of causing disorderly conditions, the firm must be able to explain how its trading decisions were made. Furthermore, the AFM points to a potential knowledge gap between machine learning experts and organizational supervisors, which could impede effective oversight [8].

Trading is inherently risky, and the development of autonomous agents carries the responsibility of considering potential financial losses for users. If commercialized, such agents could enable users with limited knowledge to suffer significant losses. A core part of this risk is the potential for an RL agent to perform unexpectedly when faced with new, unseen market data, a factor that requires further research to be fully understood.

Finally, the issue of fairness arises. RL agents that can execute trades faster and potentially more reliably than manual traders could create market inequalities. Beyond individual fairness, the widespread adoption of similar RL agents could pose a systemic risk, where correlated trading strategies might amplify market volatility. While this research aims to develop "more effective, robust and potentially safer trading agents", we acknowledge that the underlying technologies could be misused if not deployed responsibly.

7.2 Reliability

To make sure our results were as reliable as possible, we built several checks into our process and have been upfront about the limitations of our study.

A core practice was to test our experimental configurations across five distinct random seeds to account for the stochastic nature of RL training. Our results confirm the necessity of this approach, as experiments consistently exhibited high variance, with performance being highly sensitive to the agent's initialization. In one notable instance, a simple change in the input order of the same feature set resulted in significantly different performance outcomes, with Sharpe ratios of 0.58 and -0.37, underscoring the training process's high sensitivity.

In addition, the quality of the data is important for the reliability of the results. While the tick data from Dukascopy is generally considered reliable [29], our analysis revealed potential data gaps. We identified a total of $1,758$ potentially missing 15-minute candles across the dataset, as explained in Appendix A. We also acknowledge the possibility of data source bias; since the experiments were conducted on data from a specific source, an agent might learn patterns that are not generalizable to the broader market.

Our simulated environment makes several simplifying assumptions. We operate under a "Zero Market Impact Hypothesis" and do not model slippage costs. While the market impact of an agent with an initial capital of 10,000 USD is likely minimal in the highly liquid forex market, both market impact and slippage are realistic concerns that would become significant with larger trading volumes.

7.3 Reproducibility

In response to the general lack of reproducibility and comparability in the field, this study was designed to be fully transparent and replicable. We believe this is a critical step toward building a reliable body of knowledge on the application of RL in finance.

To this end, we have made the following available:

- **Public Codebase:** Our complete Python codebase is publicly accessible on GitHub, as seen at [2]. The specific commit hash used for the final experiments is provided to ensure that the exact version of the code can be replicated. All experiments can be found and identically run in the folder named `RQ2`.
- **Detailed Methodology:** We have thoroughly documented our experimental protocol, data processing steps, and the two-phase hyperparameter optimization process. The final hyperparameters are explicitly listed.
- **Specified Seeds:** The random seeds used for all experiments are specified to allow for exact replication of the training runs.

We also maintain transparency about our limitations. The specific parameters for the technical indicators themselves were not optimized; instead, they were set to default values based on common usage in other research and libraries. While their calculation is available in the codebase, we did not test the significance of these specific parameters. By providing open access to our code and a detailed account of our methods, we hope to provide a fully reproducible foundation that other researchers can use to verify, critique, and extend our findings.

8 Conclusion

This paper investigated the critical role of state representation in the performance of a Deep Q-Network (DQN) agent developed for low-frequency algorithmic trading in the forex market. Our primary goal was to systematically determine how different feature designs impact agent profitability and stability.

To answer our research questions, we developed a simulated trading environment and conducted two phases of experiments. The first phase focused on the impact of including or excluding specific categories of features: Time, Trend, Momentum, Volatility, and Agent-specific data. The second phase investigated how the quantity of information, manipulated by varying the historical lookback window across feature sets of different sizes, affected performance. This

structured approach allowed for a methodical analysis of the agent's sensitivity to both the type and volume of input data.

Our findings reveal a delicate balance in state representation design. In response to our first sub-question (SQ1), the impact of feature categories was highly dependent on the type of information. We found that providing the agent with all available trend indicators was beneficial, likely offering a more robust view of market direction. Conversely, for momentum and volatility features, a "less is more" approach was superior; single indicators outperformed larger sets, suggesting that additional features introduced confounding noise. Most significantly, providing the agent with its own historical context, specifically its current trade duration, led to a notable relative improvement, underscoring the importance of such features in a Partially Observable Markov Decision Process (POMDP) framework.

The answer to our second sub-question (SQ2) was counter-intuitive but clear: increasing the quantity of information, both through more features and deeper historical lookbacks, was consistently detrimental to the agent's performance. The best results were achieved with little to no historical context. We attribute this phenomenon to the DQN agent's tendency to overfit in high-dimensional state spaces, where it memorizes training data patterns rather than learning generalizable strategies. The agent's performance is not necessarily a definitive verdict on the value of complex features, but rather a reflection of the specific learning algorithm's limitations in processing them effectively.

A primary challenge that emerged was the agent's extreme sensitivity to initial conditions and input structure, resulting in high variance across experiments with identical configurations but different random seeds. This highlights the difficulty of isolating the incremental value of specific features and achieving deterministic performance. This instability, coupled with the agent's propensity to overfit, underscores the core problem: the design of the state representation is inextricably linked to the capabilities of the learning algorithm and the robustness of the experimental protocol.

8.1 Future Work

Based on the challenges and insights gained, future work should proceed in several key directions. A more extensive hyperparameter search is needed, and the experimental procedure must be made more robust, primarily by utilizing a larger number of random seeds and developing more sophisticated model selection criteria to ensure results are stable and not due to chance. Exploring more advanced agent architectures could be crucial for effectively processing higher-dimensional state spaces and benefiting from the potential rich information within larger feature sets and deeper historical contexts. Furthermore, investigating automated feature extraction techniques could help in creating more condensed and potent state representations. Finally, future research should focus on ensuring generalization not only across different random seeds but also across different RL algorithms, financial instruments, and market conditions to build truly robust and reliable trading agents.

9 Acknowledgements

First of all, it is important to note that this thesis was conducted as part of a larger body of work investigating the application and optimization of RL techniques in algorithmic trading. While this specific paper focuses on feature engineering and state representation, it contributes to a broader project aimed at understanding how reinforcement learning can be effectively and safely used for algorithmic trading.

I would like to express my sincere gratitude to the people who supported this research and made this thesis possible. My foremost thanks goes to my responsible professor, Neil Yorke-Smith. I am deeply grateful for his willingness to enable this self-proposed research topic. I am also incredibly grateful to my supervisors, Antonis Papapantoleon and Amin Kolarijani. Their consistent guidance and insightful feedback during our weekly meetings were key in shaping the direction and quality of this research. Finally, I wish to thank my fellow collaborators. I want to give special thanks to Robert Mertens, with whom I developed a major part of the codebase. My thanks also go to the rest of our research group, Justas Bertašius, Yavuz Hançer, and Mihai Radu Şerban, for the discussions and shared learning experience.

This work would not have been possible without the contributions of this entire team.

References

[1] Trading costs - Trading charges (FOREX.com). https://www.forex.com/en/about-us/financial-transparency/trading-costs/. Accessed: 2025-06-10.

[2] TUD CSE RP Q4 2025 Group 77. TUD-CSE-RP-RLinFinance. GitHub repository, 2025. Commit: c75542a91e254856cadcb6d13557802c5dd118f1. Accessed: June 22, 2025.

[3] Yahui Bai, Yuhe Gao, Runzhe Wan, Sheng Zhang, and Rui Song. A review of Reinforcement learning in Financial Applications. *arXiv (Cornell University)*, 10 2024.

[4] James Chen. Forex (FX): How trading in the foreign exchange market works, 11 2024.

[5] Fatima Dakalbab, Manar Abu Talib, Qassim Nasir, and Tracy Saroufil. Artificial intelligence techniques in financial trading: A systematic literature review. *Journal of King Saud University - Computer and Information Sciences*, 36(3):102015, 3 2024.

[6] Greg DePersio. Pips vs. Points vs. Ticks: What's the Difference?, 7 2024.

[7] Dukascopy Bank SA. Forex historical data feed, 2025. Accessed: 2025-06-01.

[8] The Dutch Authority for the Financial Markets. Machine learning in Algorithmic Trading. Technical report, 9 2023.

[9] Kui Fu, Yidong Yu, and Bing Li. Multi-feature supervised reinforcement learning for stock trading. *IEEE Access*, 11:77840–77855, 2023.

[10] Ben Hambly, Renyuan Xu, and Huining Yang. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33(3):437–503, 4 2023.

[11] Chien Yi Huang. Financial Trading as a Game: A deep reinforcement learning approach. *arXiv (Cornell University)*, 1 2018.

[12] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the Financial Portfolio Management problem. *arXiv (Cornell University)*, 1 2017.

[13] Leslie Kramer. Long Position vs. Short Position: What's the Difference?, 5 2025.

[14] Yang Li, Wanshan Zheng, and Zibin Zheng. Deep robust reinforcement learning for practical algorithmic trading. *IEEE Access*, 7:108014–108022, 1 2019.

[15] Daniel Liberto. Forex (FX): Definition, How to Trade Currencies, and Examples, 5 2025.

[16] Caper Marney. Are price updates a good proxyfor actual traded volume in FX? 4 2011.

[17] Adrian Millea. Deep Reinforcement Learning for Trading—A Critical Survey. *Data*, 6(11):119, 11 2021.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv (Cornell University)*, 1 2013.

[19] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[20] Deog-Yeong Park and Ki-Hoon Lee. Practical algorithmic trading using state representation learning and imitative reinforcement learning. *IEEE Access*, 9:152310–152321, 1 2021.

[21] Elvis Picardo. Bid-Ask spreads in the foreign currency exchange market, 9 2024.

[22] Tidor-Vlad Pricope. Deep Reinforcement Learning in Quantitative Algorithmic Trading: a review. *arXiv (Cornell University)*, 1 2021.

[23] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[24] Shobhit Seth. Basics of Algorithmic Trading: Concepts and Examples, 12 2023.

[25] AltexSoft Editorial Team. Reinforcement Learning explained: Overview, comparisons and applications in business, 1 2019.

[26] Thibaut Théate and Damien Ernst. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632, 1 2021.

[27] Mark Towers, Ariel Kwiatkowski, Jordan K Terry, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Jin Shen Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments.

[28] Alexandra Twin. Volume of Trade: How it Works, What it Means, and Examples, 3 2021.

[29] Tomas Vanek. Preparing accurate data for ALGO Trading: Broker Data Feed Differences - StrategyQuant, 9 2023.

[30] Hasna Haifa Zahrah and Jimmy Tirtawangsa. *Algorithmic forex trading using Q-learning*. 1 2023.

A Data Analysis

In this section, we analyze the EUR/USD exchange rate data, resampled to 15-minute candles from Dukascopy tick data spanning the years 2022, 2023, and 2024. We define unusual gaps as time differences between two consecutive candles that are neither equal to the expected 15-minute interval nor to the duration of a weekend, which typically indicates non-trading hours.

The 15-minute candle data, with the unusual gaps highlighted, is shown in Figure 4. At first glance, $1,758$ candles appear to be missing, representing approximately 4% of the full dataset. However, further analysis suggests that most of these gaps correspond to global holidays or daylight saving time changes, both of which affect trading schedules. After excluding these known anomalies, the number of genuinely missing candles drops to approximately 122, a negligible amount relative to the overall dataset.

The distribution of tick volume across the dataset is illustrated in Figure 5.

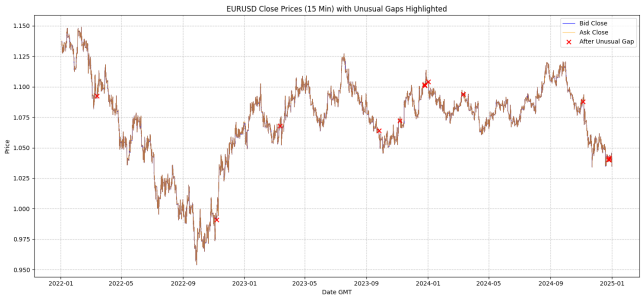

Figure 4: Exchange rates of EURUSD 15M candles resampled from tick data from dukascopy, including highlights of unusual gaps

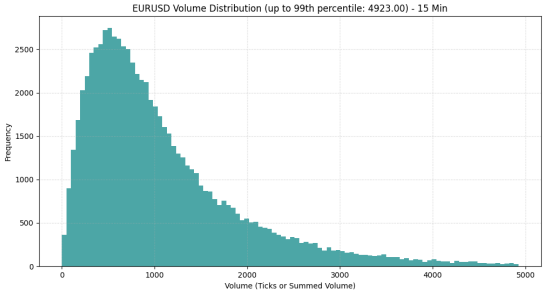

Figure 5: Volume distribution of EURUSD 15M candles resampled from tick data from dukascopy

The text output of the analysis can be seen below:

```
Analysis of time differences for EURUSD at 15 min granularity.
Expected regular time diff: 15 minutes.
Approximate weekend gap expectation (time diff for first candle after weekend): 2895 minutes.
Rows listed below are those where the time difference from the previous candle was neither the standard granularity nor a typical weekend gap.
This suggests potential missing data or unusual market closure.

Unusual gap before 2022-03-13 21:00:00 (local time in data): 2835.00 minutes. Estimated ~188.00 missing 15-min candles.
Unusual gap before 2022-11-06 22:00:00 (local time in data): 2955.00 minutes. Estimated ~196.00 missing 15-min candles.
Unusual gap before 2023-03-12 21:00:00 (local time in data): 2835.00 minutes. Estimated ~188.00 missing 15-min candles.
Unusual gap before 2023-09-25 11:00:00 (local time in data): 60.00 minutes. Estimated ~3.00 missing 15-min candles.
Unusual gap before 2023-11-05 22:00:00 (local time in data): 2955.00 minutes. Estimated ~196.00 missing 15-min candles.
Unusual gap before 2023-12-24 23:00:00 (local time in data): 2955.00 minutes. Estimated ~196.00 missing 15-min candles.
Unusual gap before 2023-12-24 23:30:00 (local time in data): 30.00 minutes. Estimated ~1.00 missing 15-min candles.
Unusual gap before 2023-12-25 22:00:00 (local time in data): 855.00 minutes. Estimated ~56.00 missing 15-min candles.
Unusual gap before 2024-01-01 22:00:00 (local time in data): 4335.00 minutes. Estimated ~288.00 missing 15-min candles.
Unusual gap before 2024-03-10 21:00:00 (local time in data): 2835.00 minutes. Estimated ~188.00 missing 15-min candles.
Unusual gap before 2024-11-03 22:00:00 (local time in data): 2955.00 minutes. Estimated ~196.00 missing 15-min candles.
Unusual gap before 2024-12-25 02:30:00 (local time in data): 30.00 minutes. Estimated ~1.00 missing 15-min candles.
Unusual gap before 2024-12-25 05:00:00 (local time in data): 30.00 minutes. Estimated ~1.00 missing 15-min candles.
Unusual gap before 2024-12-25 22:00:00 (local time in data): 855.00 minutes. Estimated ~56.00 missing 15-min candles.
Unusual gap before 2024-12-26 19:00:00 (local time in data): 75.00 minutes. Estimated ~4.00 missing 15-min candles.

Total estimated number of missing candles from these unusual gaps: 1758.00
```

Additional Notes

Dukascopy also allows direct downloads of candlestick data on their website [7]. However, for us, this resulted in up to 30% of missing data. Therefore, we opted for the tick-by-tick data download, which also allowed us to use a proxy for the volume.

B Simulated Forex Environment

This appendix provides a visual walkthrough of the custom forex environment's operational flow, as described in Section 3.4. Figure 6 illustrates the sequence of events within a single discrete timestep of the simulation. Each box represents one market data candle. At the start of a timestep (e.g., $t = 1$), the environment first calculates the "Pre-Action Equity" based on the candle's open price. The agent's chosen action is then executed at this same price. The price subsequently evolves over the remainder of the candle's duration. The reward for the action taken at timestep t is calculated based on the change in equity between the pre-action state at t and the pre-action state at the beginning of the next timestep, $t+1$. This ensures the reward fully captures the outcome of the price movement during the interval. Figure 7 presents a flowchart of the environment's pipeline for a single step. The process begins when the environment receives an action from the agent. This input triggers a series of internal calculations:

1. **Transform Action:** The discrete action is mapped to a target portfolio exposure.
2. **Execute Action:** The environment performs the necessary buy or sell trades based on market data to align the portfolio with the target exposure.
3. **State Calculation:** The agent's internal state (e.g., cash, shares held) is updated.
4. **Calculate Reward:** The reward is computed based on the change in portfolio equity.
5. **Retrieve Observation:** A new observation vector is assembled for the next timestep, incorporating market data and updated agent features.

The environment then returns the reward and the new observation to the agent, completing the step loop.

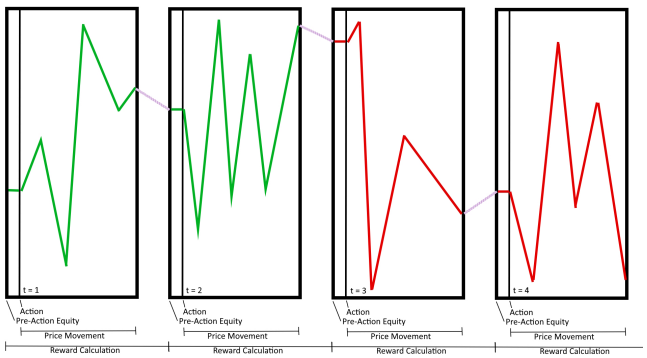

Figure 6: Visual representation of the discrete time steps and simulation steps within. Each box represents a candle, and the lines represent the price movement within that candle.

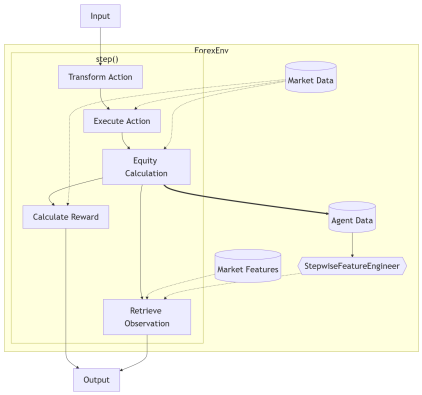

Figure 7: Visual representation of the pipeline of a single step execution of the forex environment.

C Hyperparameter Optimization

As mentioned in Section 3, we optimize our parameters on a subset of our full dataset in phases. Adjusting and performing new experiments as we go.

C.1 Data split

As explained in Section 4.1, the first 9 months of the full dataset is used for hyperparameter optimization. This dataset is chronologically split up into a training set, containing 70% of the data, and a validation set, containing the remaing 30%. We use the validation set to analyze the reaction of the models to out-of-sample data.

C.2 Baseline

Before starting training models we set the following parameters:

- **Seed:** the random seed for all used libraries and packages is set to `42`.
- **Initial Capital:** the agent starts with 10,000 U.S. dollars, the base currency.
- **Transaction Cost:** the commission of the environment is set to 0.005% to support realism.
- **DQN Policy:** the `MlpPolicy` is used with the `Adam` optimizer and `LeakyReLU` activation function. The neural network has two hidden layers of sizes 32 and 16. Other parameters are set to their default values [23], unless specified otherwise.
- **Gamma (γ):** the gamma parameter is set to 0.95.

Features

The hyperparameter optimization experiments were conducted exclusively using the Core Feature Set as defined in Section 3.3. This ensures that the tuning process was performed on a consistent and representative baseline before the main experiments were conducted.

C.3 DQN Parameters

The most notable parameters of the DQN algorithm are [23]:

- **Policy**: The neural network architecture used to approximate the Q-function. In this case, the `MlpPolicy` is a multi-layer perceptron.
  - **Q Network**: The neural network that approximates the Q-values.
  - **Optimizer**: The algorithm used to update the weights of the Q-network to minimize the loss, such as the Adam optimizer.
  - **Activation Function**: The non-linear function used in the hidden layers of the neural network, like Leaky ReLU.
- **Learning Rate**: The step size at which the optimizer updates the network's weights.
- **Buffer Size**: The maximum number of past experiences stored in the replay buffer.
- **Batch Size**: The number of experiences sampled from the replay buffer for each training update.
- **Tau (τ)**: The parameter for the soft update of the target network's weights.
- **Gamma (γ)**: The discount factor for future rewards, balancing immediate and long-term gains.
- **Train Frequency**: The number of steps in the environment between training updates.
- **Gradient Steps**: The number of gradient descent steps to perform after each new experience.
- **Target Update Interval**: The frequency (in terms of environment steps) at which the target network is updated with the weights of the main Q-network.
- **Exploration Fraction**: The fraction of the total training time over which the exploration rate (ϵ) decreases.
- **Exploration Initial Epsilon**: The starting value of ϵ for the ϵ-greedy exploration strategy.
- **Exploration Final Epsilon**: The final, minimum value of ϵ.

C.4 Phase 1

Phase one is meant to optimize the convergence of training, and to generate as close to non-random results as possible. The parameters that influence these most can be seen in Table 5. We define four different types of agents called 'aggressive', 'balanced', 'cautious', and 'patient'. Respectively, they range from fast unstable learning to slow more stable learning, as seen in Table 5. Each of these configurations are run on three granularities: 15-minute, 30-minute, and 1-hour market data, following the idea that higher granularity data carries more noise. The experiments are run for 100 episodes.

Discussion

The 'aggressive' models proved too unstable for effective learning, while the 'patient' models converged too slowly. The 'balanced' and 'cautious' configurations demonstrated the most stable and promising learning patterns across all data granularities (15-min, 30-min, and 1-hour). Therefore, we selected these two configurations to move forward into Phase 2 for a more detailed investigation of network architecture.

Table 5: Hyperparameter Configurations for Initial Convergence Experiments

Hyperparameter	Aggressive	Balanced	Cautious	Patient
Learning Rate	10^{-3}	10^{-4}	5×10^{-5}	10^{-5}
Buffer Size	5,000	30,000	60,000	100,000
Batch Size	64	256	512	512
Tau	0.01	0.005	0.0025	0.001
Train Frequency (Steps)	4	16	64	128
Target Update Interval	500	1,000	2,500	5,000
Exploration Fraction	0.25	0.33	0.40	0.50
Exploration Initial Epsilon	1.0	1.0	1.0	1.0
Exploration Final Epsilon	0.1	0.05	0.02	0.01

C.5 Phase 2

In this phase we investigate the network architecture. As such we experiment with 6 additional network architectures, adjusted from the initial architecture to either be larger or smaller. The experiments can be seen in Table 6. These experiments are run on the 15-minute, 30-minute, and 1-hour candlestick datasets similar to Phase 1, with the model types `cautious` and `balanced`.

Table 6: Phase 2: Network Architecture Configurations for Generalization Testing

Configuration Name	Hidden Layers	Rationale
Baseline	[32, 16]	The original network.
Reduced	[16, 8]	Overfitting Mitigation
Minimalist	[16]	Minimal Parameter Search
Medium Symmetric	[32, 32]	Symmetric vs. Funnel
Increased	[64, 32]	Complex Pattern Recognition
Symmetric	[64, 64]	Complex Pattern Recognition
High	[128, 64]	Maximal Parameter Search

Discussion

After evaluating all network architectures with the 'cautious' and 'balanced' configurations, a clear pattern emerged. The experiments using 1-hour candlestick data consistently yielded higher and more stable Sharpe ratios on the validation set compared to the 15-minute and 30-minute data. Among these, the 'cautious' configuration paired with the 'Baseline' network architecture ([32, 16]) provided the best balance of performance and generalization. This combination was therefore selected as the final configuration for the main experiments detailed in Section 4.

C.6 Final Configuration

The systematic, two-phase optimization process detailed in this appendix led to the selection of our final model configuration. Based on our experiments, the 'cautious' hyperparameter set (Table 5) combined with the baseline network architecture of [32, 16] (Table 6) and 1-hour data granularity was chosen. This configuration demonstrated the most robust performance on our validation dataset. The complete set of final parameters used for the main experiments is listed in Table 1 in Section 4.4.

D Experiments

D.1 Phase 1 Experiments

The unique experimental configurations explained in Section 4 for Phase 1 are shown in Table 7.

Table 7: Complete Experimental Design for Phase 1 Experiments

Category	Experiment ID	Features Investigated
Time	S1_TM_NONE	*Baseline*: No time features included.
	S1_TM_L24	Linear encoding of the 24-hour cycle.
	S1_TM_L24L7	Linear encoding of the 24-hour cycle, and 7-day cycle.
	S1_TM_S24	Sinusoidal encoding of the 24-hour cycle.
	S1_TM_SC24	Sinusoidal and Cosine encoding of the 24-hour cycle.
	S1_TM_SC24SC7	Sinusoidal and Cosine encoding of the 24-hour cycle, and 7-day cycle.
	S1_TM_ALL	All available time features (Linear, SIN, COS for 24h and 7d).
Trend	S1_TR_NONE	*Baseline*: No trend features included.
	S1_TR_NV	One non-volume-based feature: Parabolic SAR.
	S1_TR_V	One volume-based feature: VWAP.
	S1_TR_COMBO	Combination of Parabolic SAR and VWAP.
	S1_TR_ALL	All trend features: Parabolic SAR, VWAP, and KAMA.
Momentum	S1_MO_NONE	*Baseline*: No momentum features included.
	S1_MO_NV	One non-volume-based feature: MACD.
	S1_MO_V	One volume-based feature: MFI.
	S1_MO_COMBO	Combination of MACD and MFI.
	S1_MO_ALL	All momentum features: MACD, MFI, and CCI.
Volatility	S1_VO_NONE	*Baseline*: No volatility features included.
	S1_VO_NV	One non-volume-based feature: Bollinger Bands.
	S1_VO_V	One volume-based feature: Ease of Movement (EOM).
	S1_VO_COMBO	Combination of Bollinger Bands and EOM.
	S1_VO_ALL	All volatility features: Bollinger Bands, EOM, and ATR.
Agent	S1_AG_NONE	*Baseline*: No agent features included.
	S1_AG_CE	Agent's current market exposure.
	S1_AG_DT	Duration of the agent's current trade.
	S1_AG_ALL	All agent features: Current Exposure and Trade Duration.
Combinatory	S1_COMBO_COMBO	A combination of select features from each category.
	S1_COMBO_ALL	The complete set of all engineered features.

D.2 Phase 2 Experiments

The unique experimental configurations explained in Section 4 for Phase 2 are shown in Table 8.

Table 8: Complete Experimental Design for SQ2: Investigating the Impact of Historical Lookback

Configuration	Experiment ID	Features and Historical Lookback Investigated
SMALL	S2_SMALL_0	Core Feature Set. Lookback of 0 on Trend, Momentum, and Volatility features.
	S2_SMALL_1	Core Feature Set. Lookback of 1 on Trend, Momentum, and Volatility features.
	S2_SMALL_2	Core Feature Set. Lookback of 2 on Trend, Momentum, and Volatility features.
	S2_SMALL_4	Core Feature Set. Lookback of 4 on Trend, Momentum, and Volatility features.
	S2_SMALL_8	Core Feature Set. Lookback of 8 on Trend, Momentum, and Volatility features.
	S2_SMALL_16	Core Feature Set. Lookback of 16 on Trend, Momentum, and Volatility features.
	S2_SMALL_32	Core Feature Set. Lookback of 32 on Trend, Momentum, and Volatility features.
MEDIUM	S2_MEDIUM_0	Medium Feature Set (COMBO for Trend/Momentum/Volatility, ALL for Time/Agent). Lookback of 0.
	S2_MEDIUM_1	Medium Feature Set (COMBO for Trend/Momentum/Volatility, ALL for Time/Agent). Lookback of 1.
	S2_MEDIUM_2	Medium Feature Set (COMBO for Trend/Momentum/Volatility, ALL for Time/Agent). Lookback of 2.
	S2_MEDIUM_4	Medium Feature Set (COMBO for Trend/Momentum/Volatility, ALL for Time/Agent). Lookback of 4.
	S2_MEDIUM_8	Medium Feature Set (COMBO for Trend/Momentum/Volatility, ALL for Time/Agent). Lookback of 8.
	S2_MEDIUM_16	Medium Feature Set (COMBO for Trend/Momentum/Volatility, ALL for Time/Agent). Lookback of 16.
	S2_MEDIUM_32	Medium Feature Set (COMBO for Trend/Momentum/Volatility, ALL for Time/Agent). Lookback of 32.
LARGE	S2_LARGE_0	All engineered features. Lookback of 0 on Trend, Momentum, and Volatility features.
	S2_LARGE_1	All engineered features. Lookback of 1 on Trend, Momentum, and Volatility features.
	S2_LARGE_2	All engineered features. Lookback of 2 on Trend, Momentum, and Volatility features.
	S2_LARGE_4	All engineered features. Lookback of 4 on Trend, Momentum, and Volatility features.
	S2_LARGE_8	All engineered features. Lookback of 8 on Trend, Momentum, and Volatility features.
	S2_LARGE_16	All engineered features. Lookback of 16 on Trend, Momentum, and Volatility features.
	S2_LARGE_32	All engineered features. Lookback of 32 on Trend, Momentum, and Volatility features.

E Experiment Result Graphs

The graphs in the following sections visualize the performance of the various experimental configurations throughout the training process. During the training of each experimental setup, the state of the model was saved at the conclusion of every episode. Each of these saved models was evaluated on the training and validation dataset, showed in the first two subsections. The plots show the mean Sharpe ratio (and its standard deviation across five random seeds) for the models saved at each episode, offering insight into learning stability, convergence, and generalization. The final subsection shows performance of the final selected models, which are also shown in Table 3 and 4 in Section 5.

E.1 Training Dataset

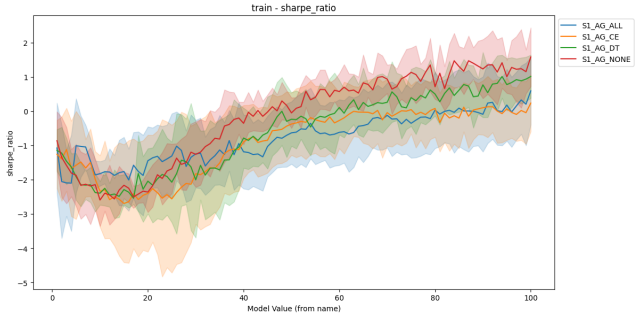

Figure 8: Results for experiments in group [FE_S1_Agent] on the train dataset.

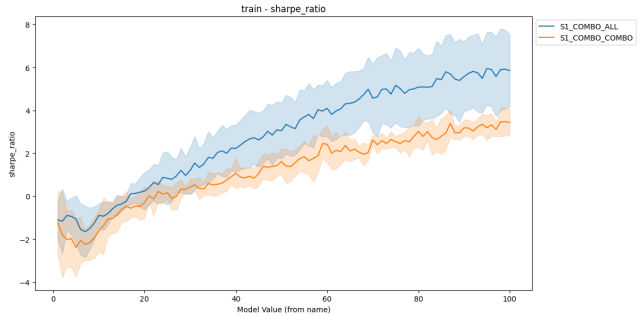

Figure 9: Results for experiments in group [FE_S1_Combinatory] on the train dataset.

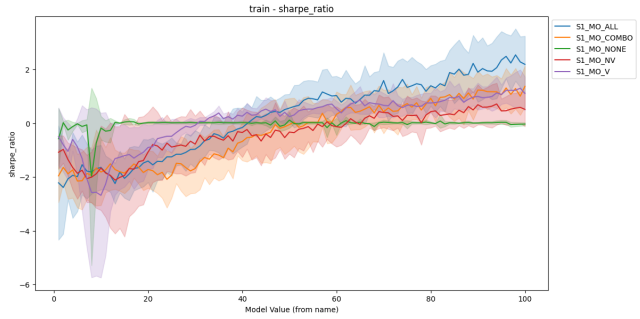

Figure 10: Results for experiments in group [FE_S1_Momentum] on the train dataset.

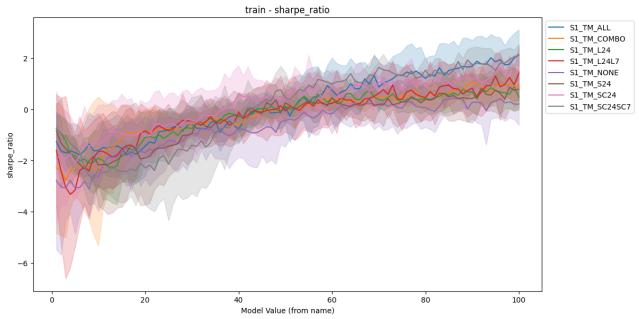

Figure 11: Results for experiments in group [FE_S1_Time] on the train dataset.

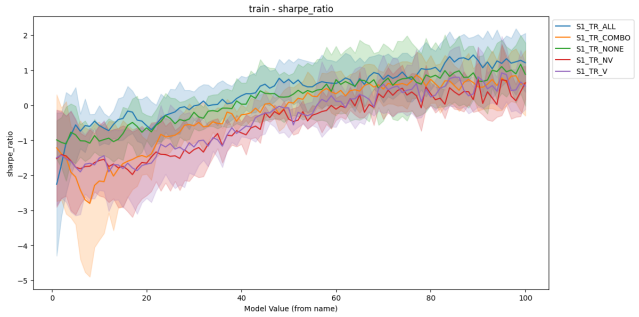

Figure 12: Results for experiments in group [FE_S1_Trend] on the train dataset.

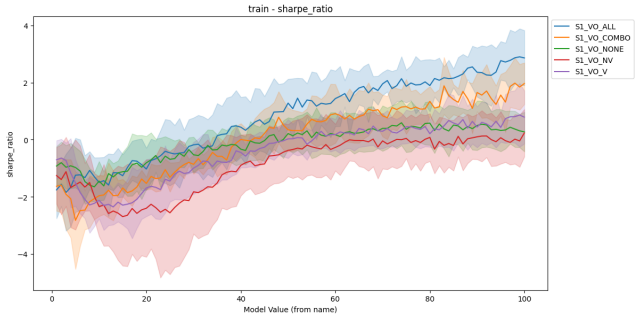

Figure 13: Results for experiments in group [FE_S1_Volatility] on the train dataset.

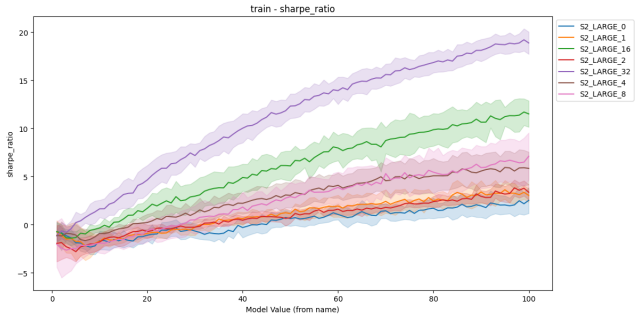

Figure 14: Results for experiments in group [FE_S2_LARGE] on the train dataset.

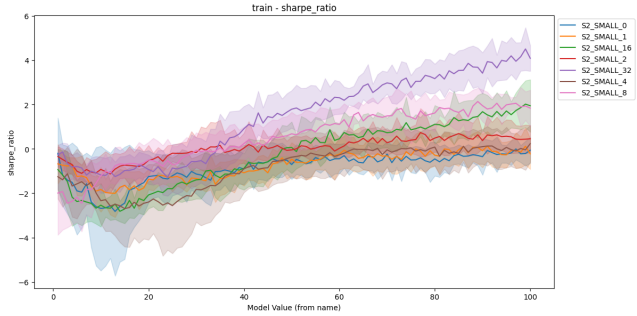

Figure 15: Results for experiments in group [FE_S2_SMALL] on the train dataset.

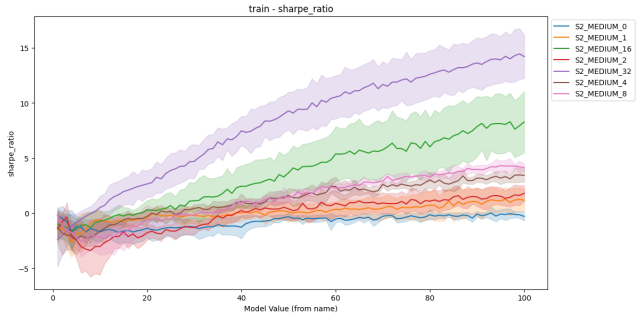

Figure 16: Results for experiments in group [FE_S2_MEDIUM] on the train dataset.

E.2 Validation Dataset

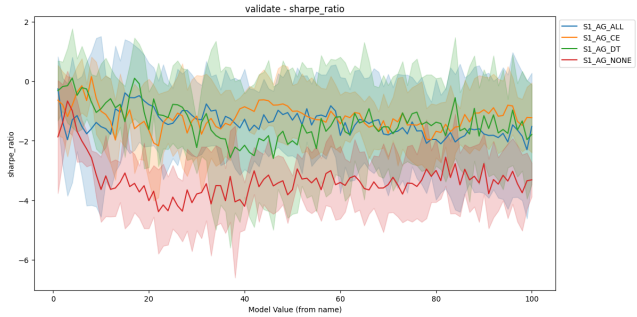

Figure 17: Results for experiments in group [FE_S1_Agent] on the validate dataset.

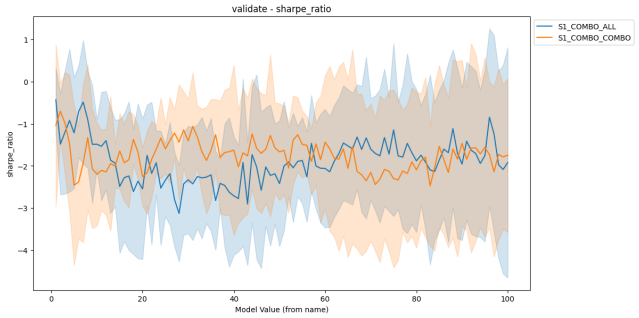

Figure 18: Results for experiments in group [FE_S1_Combinatory] on the validate dataset.

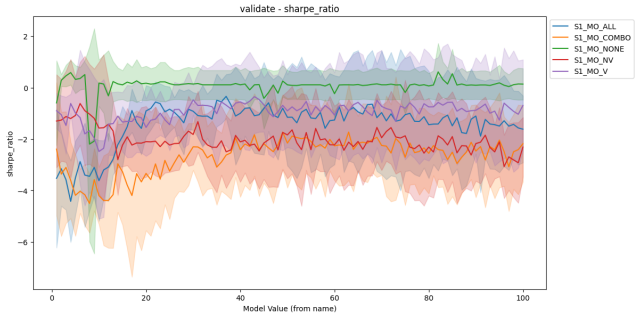

Figure 19: Results for experiments in group [FE_S1_Momentum] on the validate dataset.

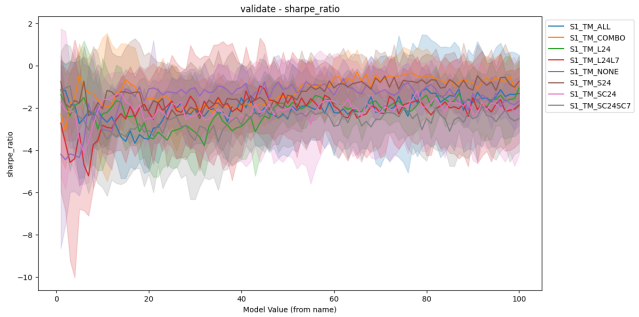

Figure 20: Results for experiments in group [FE_S1_Time] on the validate dataset.

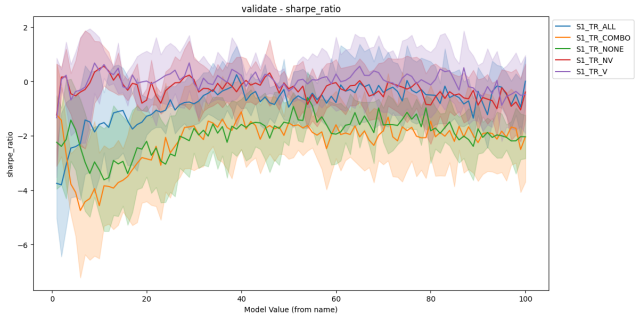

Figure 21: Results for experiments in group [FE_S1_Trend] on the validate dataset.

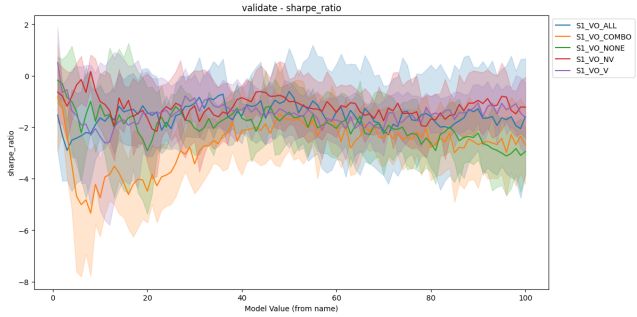

Figure 22: Results for experiments in group [FE_S1_Volatility] on the validate dataset.

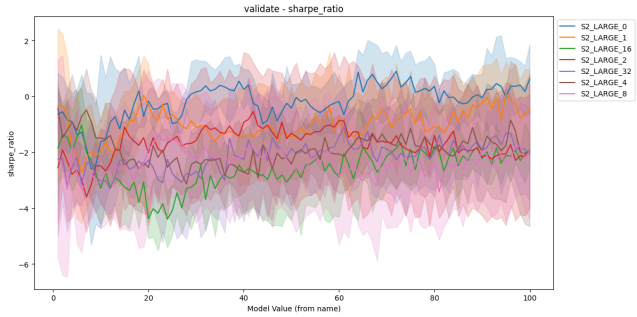

Figure 23: Results for experiments in group [FE_S2_LARGE] on the validate dataset.

Figure 24: Results for experiments in group [FE_S2_MEDIUM] on the validate dataset.

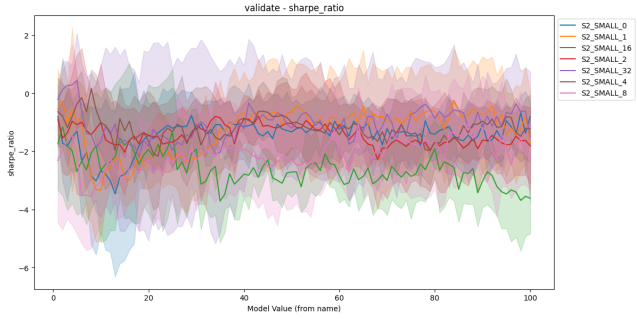

Figure 25: Results for experiments in group [FE_S2_SMALL] on the validate dataset.

E.3 Evaluation Models & Dataset

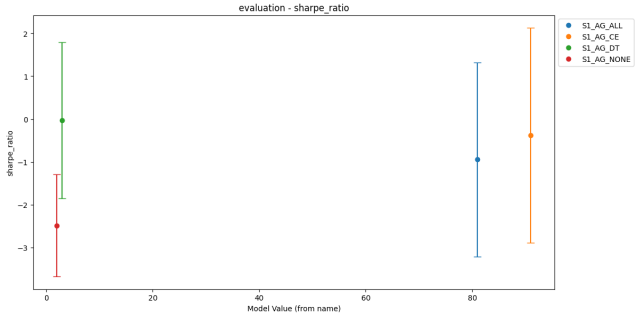

Figure 26: Results for experiments in group [FE_S1_Agent] on the evaluation dataset.

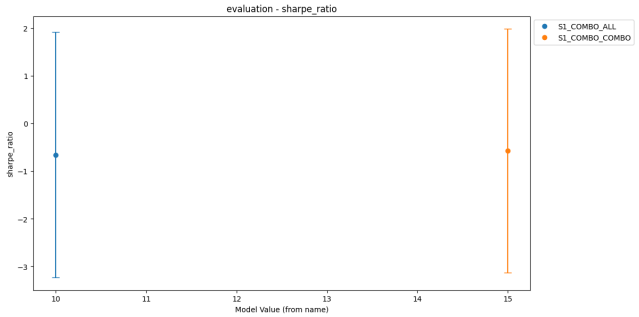

Figure 27: Results for experiments in group [FE_S1_Combinatory] on the evaluation dataset.

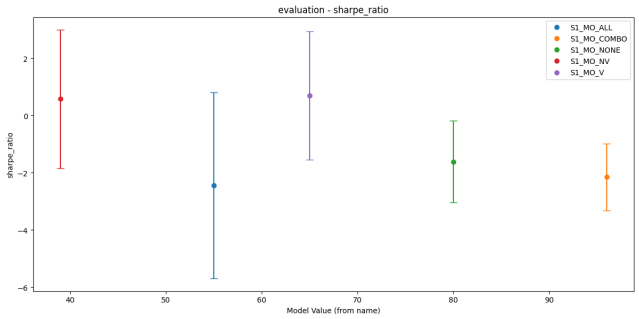

Figure 28: Results for experiments in group [FE_S1_Momentum] on the evaluation dataset.

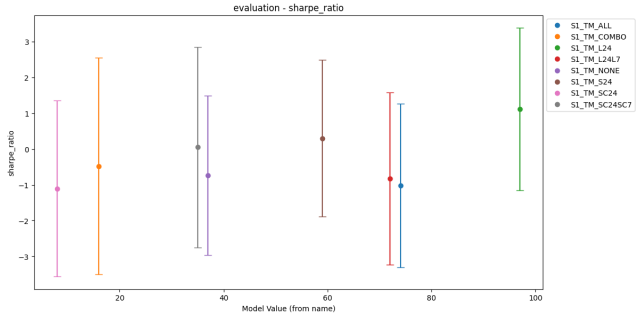

Figure 29: Results for experiments in group [FE_S1_Time] on the evaluation dataset.

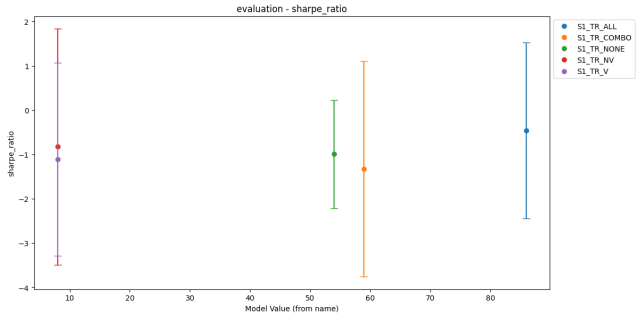

Figure 30: Results for experiments in group [FE_S1_Trend] on the evaluation dataset.

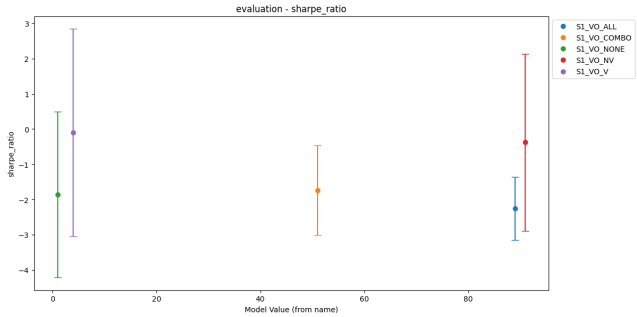

Figure 31: Results for experiments in group [FE_S1_Volatility] on the evaluation dataset.

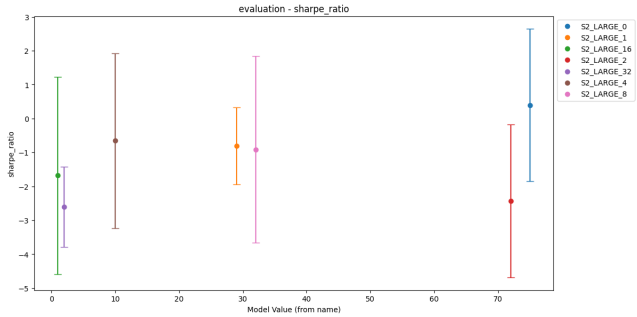

Figure 32: Results for experiments in group [FE_S2_LARGE] on the evaluation dataset.

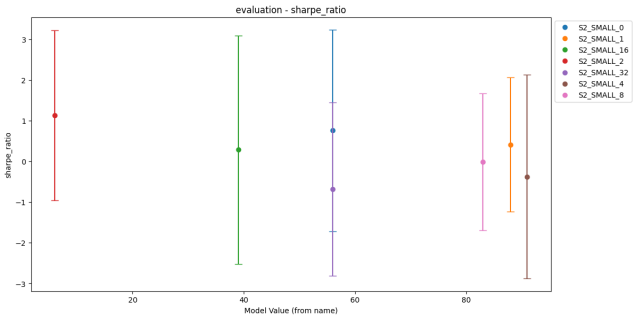

Figure 33: Results for experiments in group [FE_S2_SMALL] on the evaluation dataset.

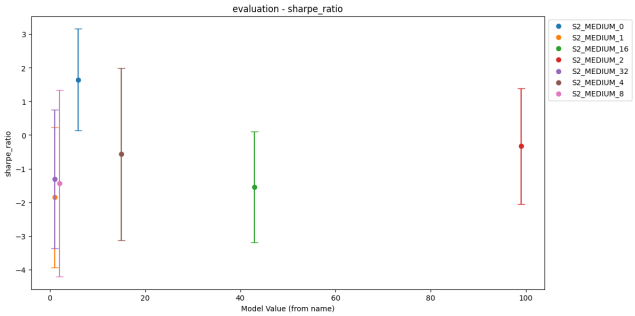

Figure 34: Results for experiments in group [FE_S2_MEDIUM] on the evaluation dataset.