# Effects of exploration-exploitation strategies in dynamic Forex markets

**The use of Reinforcement Learning in Algorithmic Trading**

**Serban Mihai-Radu**

**Supervisor(s): Neil Yorke-Smith, Amin Sharifi Kolarijani, Antonis Papapantoleon**

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

## Abstract

This paper examines how different exploration strategies affect the learning behavior and trading performance of reinforcement learning (RL) agents in a custom Foreign Exchange (Forex) environment. By holding all other components constant—including model architecture, features, and reward function—the study isolates the role of exploration in deep Q-learning.

Three strategies were compared: Epsilon-Greedy, Boltzmann, and Max-Boltzmann. The hybrid Max-Boltzmann approach delivered the most stable and profitable outcomes, suggesting that weighted, value-aware exploration can be beneficial in high-risk domains. The results also highlight the impact of non-Markovian structure in financial environments and limitations of equity-based rewards.

Beyond empirical results, this work contributes a modular, reproducible RL framework for trading, and opens new questions about the suitability of exploration techniques in environments with asymmetric risk and irreversible actions.

## 1 Introduction

### 1.1 Context and Motivation

The foreign exchange market (Forex) is one of the most liquid and fast-paced financial markets, with an estimated daily turnover exceeding $7.5 trillion in 2022 [1]. Its global nature ensures 24-hour trading across major financial centers, and its high leverage ratios enable significant capital exposure with relatively small investments. These factors attract a wide spectrum of market participants—from central banks and multinational corporations to institutional investors and retail traders.

At the same time, the Forex market is notoriously volatile and influenced by a wide range of factors, including macroeconomic indicators (e.g., interest rates, inflation reports), geopolitical developments, and unexpected news events. This complexity makes traditional rule-based or static statistical models insufficiently responsive to changing market regimes.

The need for adaptive, data-driven approaches has grown in parallel with advances in computational finance. As algorithmic trading continues to rise in prevalence, there is increasing interest in leveraging machine learning—and particularly Reinforcement Learning (RL)—to develop agents that can learn profitable strategies autonomously and respond to shifting market dynamics. Forex, with its continuous operation, short reaction windows, and abundant data, presents an ideal yet challenging domain for testing such intelligent trading systems. However, as highlighted in recent reviews [8], applying RL to financial markets remains difficult due to non-stationary dynamics, partial observability, and the risk of overfitting to transient market patterns.

### 1.2 Reinforcement Learning in Trading

Reinforcement Learning (RL), which focuses on learning optimal policies through trial-and-error interactions with an environment, has gained increasing interest in quantitative trading. In particular, the combination of RL with deep neural networks, popularized by the Deep Q-Network (DQN) architecture [7], has shown promise in tackling high-dimensional control problems. This hybrid has been applied in financial contexts to learn trading strategies directly from market data.

Carapuço et al. [2] implemented a Deep Q-Network (DQN) to trade the EUR/USD currency pair and demonstrated that RL agents could exploit price trends and achieve profitable returns in backtesting, emphasizing the practicality of neural-network-enhanced RL in Forex markets. Similarly, Théate and Ernst [12] showed that deep RL agents trained using historical limit order book data can learn profitable strategies and adapt to market dynamics, highlighting the feasibility of end-to-end learning in high-frequency trading settings. These studies underscore the growing potential of RL in algorithmic trading and motivate a deeper investigation into the role of exploration strategies within such frameworks.

### 1.3 Research Gap

A fundamental challenge remains underexplored in the literature: the effect of exploration-exploitation strategies on learning stability and trading performance in non-stationary markets like Forex. Unlike stationary environments, the Forex market evolves over time, making naive or fixed exploration schedules potentially suboptimal.

Huang [5] treated financial trading as a game and showed that deep RL agents can outperform heuristic-based strategies in simulated environments, reinforcing the potential of RL for adaptive decision-making in finance. Ghosh et al. [3] combined deep learning models (LSTMs) with tree-based methods to forecast price movements for intraday trading, further illustrating the benefits of model-driven strategies in volatile markets. However, despite these encouraging results, the mechanisms through which exploration choices affect learning dynamics and trading performance—particularly in non-stationary financial environments—remain poorly understood.

### 1.4 Research Question and Contributions

This paper investigates the research question: **How do different exploration-exploitation strategies affect the ability of an RL agent to balance learning and performance in a dynamic Forex market?**

To address this, the study examines how exploration design shapes both learning behavior and trading outcomes under controlled conditions. The following sub-questions are explored through empirical analysis:

- **How do different strategies affect the stability and convergence of policy learning?**
- **What impact does exploration design have on profitability and drawdown risk?**
- **How do agents allocate between long, short, and hold actions under different exploration schemes?**
- **What advantages do hybrid exploration methods offer in high-risk decision environments like trading?**

In addition to these empirical insights, this study contributes a reusable and extensible framework for financial reinforcement learning research.

**The key contributions of this work are:**

- a modular, Gym-compatible Forex trading environment that simulates realistic trading mechanics including long/short positions, bid-ask spread, and transaction logic;
- a systematic comparison of exploration-exploitation strategies for RL agents in financial markets, using DQN and its extensions;
- an evaluation pipeline that integrates financial metrics with learning dynamics to provide practical insights into agent behavior under different exploration schemes;
- extensible code templates for advanced strategies such as Noisy Networks and Curiosity-Driven Exploration, supporting future experimentation.

### 1.5 Paper Structure

The remainder of this paper is structured as follows. Section 2 reviews relevant literature on RL in trading and exploration strategies. Section 3 describes the architecture of the trading environment, the RL agents, and the evaluation setup. Section 4 presents and compares experimental results. Section 5 discusses implications and limitations. Section 6 outlines responsible research practices. Section 7 concludes with key takeaways and directions for future work.

## 2 Background

### 2.1 Reinforcement Learning Foundations

Reinforcement Learning (RL) provides a powerful framework for sequential decision-making under uncertainty. An RL agent learns by interacting with its environment and receiving scalar feedback in the form of rewards. The standard formulation is based on a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ [10], where:

- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- $\mathcal{P}(s'|s, a)$ is the transition probability,
- $\mathcal{R}(s, a)$ is the reward function,
- $\gamma \in [0, 1)$ is a discount factor.

The objective is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative discounted reward:

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

While this framework is mathematically elegant and forms the foundation of RL research, real-world financial environments frequently violate MDP assumptions. In practice, financial markets are partially observable, exhibit non-stationary dynamics, and involve delayed or stochastic rewards. The environment transition function $\mathcal{P}(s'|s, a)$ is not only unknown but also unstable over time due to regime shifts, economic shocks, and latent variables. These violations introduce challenges in applying RL reliably and motivate algorithmic simplifications that increase robustness and interpretability.

### 2.2 Deep Q-Networks as a Design Choice

Given these complexities, this study employs the Deep Q-Network (DQN) algorithm [7] as the base RL agent. DQN is a value-based method that approximates the optimal action-value function $Q^*(s, a)$ using a deep neural network. The network is trained using a temporal difference loss:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]$$

where $\theta$ are the current network weights, and $\theta^-$ are weights of a target network that is periodically updated for stability.

DQN has several properties that make it particularly suitable for controlled studies of exploration strategies:

- It produces **deterministic policies** at evaluation time via greedy action selection, enabling stable and reproducible comparisons across strategies.
- It supports **built-in exploration mechanisms** such as $\varepsilon$-greedy and Boltzmann sampling without requiring probabilistic policies.
- It has a well-understood empirical behavior and serves as a baseline in many prior financial RL studies [2] [12].

Although financial decision-making often involves continuous control (e.g., portfolio allocation or position sizing), this study adopts a discrete action space for both practical and conceptual reasons. The DQN implementation used in this work, based on the Stable Baselines3 framework, supports only discrete actions, since it estimates a separate Q-value for each action. While more complex algorithms like DDPG or SAC support continuous action spaces, they also introduce additional challenges in stability and interpretability. Given the high complexity and non-stationarity of financial environments, this discrete framing is not merely a limitation, but a deliberate simplification that enables clearer analysis and more stable learning.

By framing the problem as choosing among a small, interpretable set of actions—such as opening long, opening short, or holding—we reduce the risk of spurious results driven by noisy gradients or unstable value estimates in continuous domains. This simplification is particularly justified in the presence of MDP violations such as partial observability and non-stationary transitions. Prior work suggests that, in such high-uncertainty environments, discretizing the action space can improve policy stability, accelerate convergence, and support more interpretable exploration dynamics [11]. The resulting design—combining DQN with a minimal yet realistic action set—therefore reflects a pragmatic trade-off between representational richness and experimental tractability.

### 2.3 Exploration-Exploitation Strategies

This study investigates how different exploration strategies influence the learning dynamics and performance of DQN agents in the Forex market. We focus on three widely discussed strategies:

**Epsilon-Greedy.** The $\varepsilon$-greedy strategy is a foundational exploration technique in reinforcement learning [10]. At each timestep, the agent selects a random action with probability $\varepsilon$, and the action with the highest estimated Q-value with probability $1-\varepsilon$:

$$a_t = \begin{cases} \text{random action} & \text{with probability } \varepsilon \\ \arg\max_a Q(s_t, a) & \text{with probability } 1-\varepsilon \end{cases}$$

Typically, $\varepsilon$ is annealed over time. While simple and robust, this approach does not distinguish between poor and moderately promising actions during exploration.

**Boltzmann (Softmax) Exploration.** Boltzmann exploration improves on $\varepsilon$-greedy by converting Q-values into a softmax probability distribution:

$$P(a \mid s_t) = \frac{\exp(Q(s_t, a)/\tau)}{\sum_{a'} \exp(Q(s_t, a')/\tau)}$$

The temperature $\tau$ modulates the degree of exploration: high $\tau$ flattens the distribution, while low $\tau$ sharpens it. This method allocates exploration effort more effectively by favoring actions with relatively high value estimates [6].

**Max-Boltzmann Exploration.** Max-Boltzmann exploration is a hybrid scheme proposed in recent work on deep recurrent RL [13]. It combines $\varepsilon$-greedy structure with Boltzmann sampling:

$$a_t = \begin{cases} \text{sample from softmax}(Q(s_t, \cdot)) & \text{with probability } \varepsilon \\ \arg\max_a Q(s_t, a) & \text{with probability } 1-\varepsilon \end{cases}$$

This strategy aims to leverage the smooth preference scaling of Boltzmann while retaining the simplicity and fallback of greedy selection. Zangirolami and Borrotti [13] showed that Max-Boltzmann exploration improves stability and sample efficiency in uncertain and partially observable settings—properties often encountered in financial environments.

Together, these strategies differ in how they inject stochasticity, prioritize uncertainty, and balance risk. Given the volatility and non-stationarity of the Forex market, understanding these differences is critical to developing agents that are both profitable and adaptable.

## 3 Methods

This section details the trading environment, data processing pipeline, feature engineering components, reinforcement learning setup, and experimental protocol. All aspects unrelated to exploration-exploitation were fixed to ensure controlled comparisons across strategies. The full implementation is available on GitHub [9], with modular code organized by environment design, feature engineering, and model training.

### 3.1 Data and Preprocessing

The historical dataset used in this study consists of EUR/USD candlestick records retrieved from the Dukascopy public API. It includes high-resolution bid and ask prices and spans from January 2, 2022, to May 16, 2025. The data is aggregated at a fixed 15-minute interval, which balances temporal granularity with computational efficiency.

To ensure temporal consistency and realism, the raw data undergoes several preprocessing steps:

- Timestamp alignment and removal of non-trading hours,
- Forward-filling of missing entries,
- Outlier and gap diagnostics,
- Retention of bid/ask OHLC[1] prices for accurate execution modeling.

Each row in the dataset contains synchronized bid and ask OHLC prices and a volume field (which is unused). The data is split chronologically into training and evaluation sets using an 80/20 ratio, with no shuffling, to preserve the temporal structure and avoid data leakage.

### 3.2 Feature Engineering Pipeline

Agent observations include both market-derived and agent-specific features. These are computed using a modular pipeline with per-step execution and strict chronological ordering, ensuring no future leakage. The mathematical definitions of each feature are formally described in Appendix A.

The feature pipeline is implemented in `feature_engineer.py`, where transformations are applied using the `FeatureEngineer.add()` and `run()` methods.

**Market Features.** The following indicators are computed from historical bid prices:

- **Price momentum:** percentage change in the bid close over the last 1 and 5 steps,
- **Trend signals:** EMA(20) and EMA(50) as ratios of the current bid close [2],
- **RSI(14):** normalized to the $[0, 1]$ interval using min-max scaling [3].

**Agent-State Features.** Three dynamic indicators reflect the internal portfolio state:

- **Current exposure:** $(\text{equity} - \text{cash})/\text{equity}$,
- **Target exposure:** current intended position derived from the selected action,
- **Trade duration:** normalized number of steps spent in the current exposure.

### 3.3 Trading Environment Design

Trading is modeled as a discrete-time, episodic environment implemented in `forex_env.py`, built on the OpenAI Gym interface. Each episode walks sequentially through historical market data with no random resets or lookahead.

[1]OHLC stands for Open, High, Low, Close—commonly used price data points in candlestick charts to summarize trading activity within a time interval.

**Action Space.** The agent selects one of three discrete actions:

- **Buy (long):** allocate all capital into a long position,
- **Sell (short):** allocate all capital into a short position,
- **Hold (cash):** remain fully in cash.

These actions map to target exposures in $\{-1, 0, +1\}$ and are executed using real-time bid/ask prices. Trade logic is implemented in `trade.py`, using helper methods such as `execute_trade()` and `calculate_equity()`. Execution is asymmetric: long positions buy at the ask and sell at the bid; short positions reverse this relationship.

**Reward Function.** The agent is rewarded based on the logarithmic change in equity:

$$r_t = \log(E_t) - \log(E_{t-1})$$

This reward formulation captures risk-adjusted growth, is scale-invariant, and has precedent in financial RL literature [5]. It is injected at runtime as a custom reward function defined in `rewards.py`.

**Constraints.** To isolate policy learning, the environment makes the following assumptions:

- Zero transaction costs and slippage,
- No market impact or latency,
- All-in trading (no fractional allocation),
- Deterministic dynamics (no randomness or noise).

### 3.4 Reinforcement Learning Setup

Agents are trained using the Deep Q-Network (DQN) algorithm [7]. All hyperparameters are fixed across exploration variants:

- Two hidden layers with 128 ReLU units,
- Learning rate: $1 \times 10^{-4}$,
- Discount factor: $\gamma = 0.99$,
- Batch size: 64,
- Replay buffer size: 50,000 transitions,
- Target network update every 500 steps,
- Training frequency: every 4 steps.

The DQN models are instantiated and trained through the experimental script `RQ5/main.py`, which also controls reproducibility, environment setup, and logging.

### 3.5 Exploration Strategies and Parameter Selection

We compare three discrete-action exploration strategies:

- **Epsilon-Greedy:** $\varepsilon$ decays linearly from a fixed initial value to a final value over a defined training fraction,
- **Boltzmann Exploration:** actions are sampled from a softmax distribution over Q-values with temperature $\tau$ [6],
- **Max-Boltzmann Exploration:** with probability $\varepsilon$, actions are sampled via Boltzmann; otherwise, the greedy action is chosen [13].

All exploration parameters (e.g., $\varepsilon$ schedule, temperature) are fixed prior to training. The only difference between strategy implementations lies in the action-selection logic. Custom exploration policies are implemented as subclasses of the standard DQN in `boltzmann_dqn.py`.

### 3.6 Training and Evaluation Protocol

Agents are trained for a fixed number of episodes—20 in all experiments presented in this paper—each consisting of a full pass through the training set. This limit was chosen based on preliminary runs showing that agent performance often began to plateau or overfit after 20–25 episodes. A total of 21 models (including the final post-training model) are saved per run. The training and evaluation workflow is managed by `train_model()` and `evaluate_and_analyze_model()` in `train_eval.py`.

During training:

- Model checkpoints are saved after each episode,
- Evaluation is performed on the full test set using deterministic policies (no exploration),
- Logged outputs include episodic rewards, equity progression, and action usage statistics.

To compare performance across strategies, we compute a set of standard financial metrics:

- **Sharpe Ratio:** return per unit of volatility,
- **Maximum Drawdown:** worst historical peak-to-trough drop in equity,
- **Total Equity Change (%):** net percentage growth in portfolio value,
- **Profit Factor:** ratio of gross profits to gross losses,
- **Winning Rate (%):** percentage of trades yielding positive return.

These metrics are computed consistently across agents using a unified analysis pipeline. Mathematical definitions are provided in Appendix B.

## 4 Results

This section presents and compares the performance of three exploration strategies—Epsilon-Greedy, Boltzmann, and Max-Boltzmann—within the custom Forex trading environment. All hyperparameters, market conditions, and evaluation procedures were held constant to ensure that observed differences stem solely from the exploration mechanism.

### 4.1 Epsilon-Greedy Results

The Epsilon-Greedy agent was trained using an exploration fraction of 0.8. Evaluation results indicate limited learning and inconsistent profitability. While the agent occasionally achieved modest returns in intermediate checkpoints, its final performance failed to generalize reliably.

Across 21 training episodes, total evaluation returns mostly remained negative, with drawdowns frequently exceeding \$2,000. Sharpe Ratios fluctuated heavily—many episodes yielded values between –6 and 0, with only a few checkpoint models briefly entering positive territory. Profit Factors stayed close to 1.0, indicating minimal edge over randomness.

The final trained model underperformed in every major metric:

- **Sharpe Ratio:** –0.24,
- **Profit Factor:** 0.995,
- **Equity Gain:** –177.99,
- **Max Drawdown:** –1318.03,
- **Total Trades:** 1.

Notably, the agent executed only one trade during evaluation—an extreme case of policy collapse—suggesting that the learned strategy avoided the market altogether in its final state.

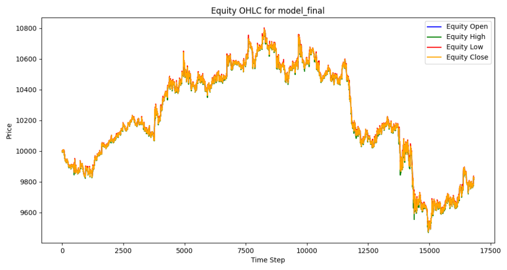
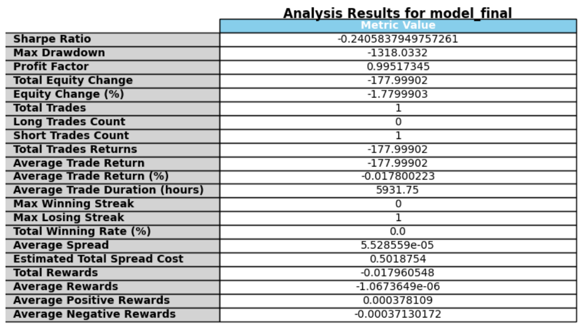


Figure 1: Equity OHLC for Epsilon-Greedy agent (evaluation). Uptrend followed by sharp retreat and stagnation.



| | Analysis Results for model_final Metric Value |
|---|---|
| **Sharpe Ratio** | -0.2405837949757261 |
| **Max Drawdown** | -1318.0332 |
| **Profit Factor** | 0.99517345 |
| **Total Equity Change** | -177.99902 |
| **Equity Change (%)** | -1.7799903 |
| **Total Trades** | 1 |
| **Long Trades Count** | 0 |
| **Short Trades Count** | 1 |
| **Total Trades Returns** | -177.99902 |
| **Average Trade Return** | -177.99902 |
| **Average Trade Return (%)** | -0.017800223 |
| **Average Trade Duration (hours)** | 5931.75 |
| **Max Winning Streak** | 0 |
| **Max Losing Streak** | 1 |
| **Total Winning Rate (%)** | 0.0 |
| **Average Spread** | 5.528559e-05 |
| **Estimated Total Spread Cost** | 0.5018754 |
| **Total Rewards** | -0.017960548 |
| **Average Rewards** | -1.0673649e-06 |
| **Average Positive Rewards** | 0.000378109 |
| **Average Negative Rewards** | -0.00037130172 |

Figure 2: Final evaluation metrics for Epsilon-Greedy agent.

Despite brief moments of promise in mid-training episodes, the Epsilon-Greedy agent ultimately failed to learn a robust trading policy. Its reliance on uniform exploration and lack of value-aware sampling likely hindered convergence in the highly stochastic and non-stationary Forex environment.

## 4.2 Boltzmann Results

The Boltzmann agent, which samples actions via temperature-scaled softmax, showed slightly better trade balance but similarly struggled with performance degradation. Although it achieved a higher win rate than Epsilon-Greedy (53.3%), evaluation results were disappointing overall:

- **Sharpe Ratio:** –0.14,
- **Profit Factor:** 0.99,
- **Equity Gain:** –45.35,
- **Max Drawdown:** –363.28,
- **Total Trades:** 1380.

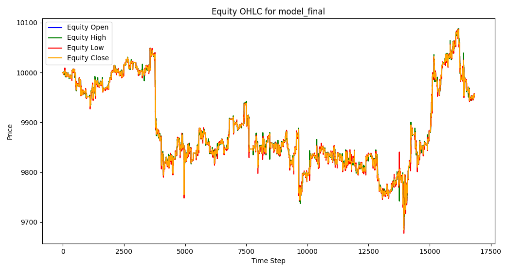
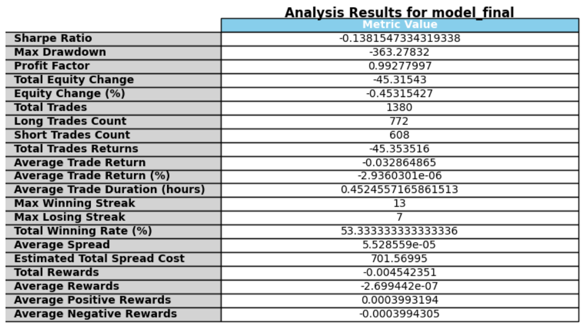


Figure 3: Equity OHLC for Boltzmann agent (evaluation). Volatile behavior with minimal long-term gain.



| | Analysis Results for model_final Metric Value |
|---|---|
| **Sharpe Ratio** | -0.1381547334319338 |
| **Max Drawdown** | -363.27832 |
| **Profit Factor** | 0.99277997 |
| **Total Equity Change** | -45.31543 |
| **Equity Change (%)** | -0.45315427 |
| **Total Trades** | 1380 |
| **Long Trades Count** | 772 |
| **Short Trades Count** | 608 |
| **Total Trades Returns** | -45.353516 |
| **Average Trade Return** | -0.032864865 |
| **Average Trade Return (%)** | -2.9360301e-06 |
| **Average Trade Duration (hours)** | 0.4524557165861513 |
| **Max Winning Streak** | 13 |
| **Max Losing Streak** | 7 |
| **Total Winning Rate (%)** | 53.333333333333336 |
| **Average Spread** | 5.528559e-05 |
| **Estimated Total Spread Cost** | 701.56995 |
| **Total Rewards** | -0.004542351 |
| **Average Rewards** | -2.699442e-07 |
| **Average Positive Rewards** | 0.0003993194 |
| **Average Negative Rewards** | -0.0003994305 |

Figure 4: Final evaluation metrics for Boltzmann agent.

Despite occasional improvements during training, the agent failed to consistently capitalize on market structure. Sharpe ratios for training models ranged widely (as low as –7), with no sustained positive trajectory.

## 4.3 Max-Boltzmann Results

The Max-Boltzmann agent, combining greedy exploitation with occasional Boltzmann sampling, demonstrated markedly improved learning behavior. From mid-training onward, both train and evaluation metrics exhibited sustained improvement:

- **Sharpe Ratio:** 1.04,
- **Profit Factor:** 1.02,
- **Equity Gain:** 641.19,
- **Max Drawdown:** –550.11,
- **Total Trades:** 843.

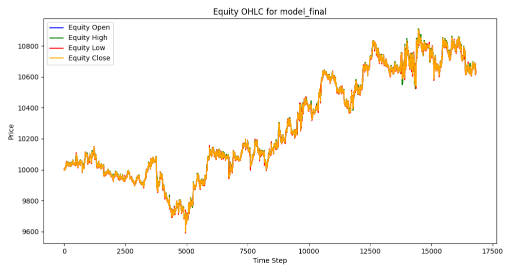
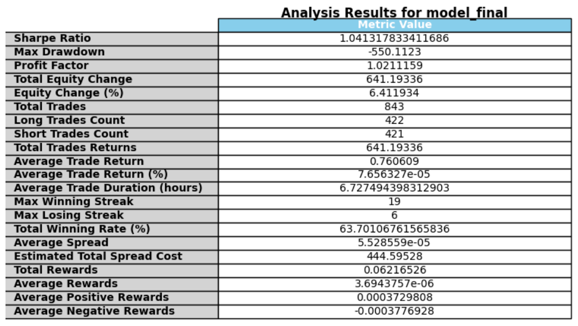


Figure 5: Equity OHLC for Max-Boltzmann agent (evaluation). Clear upward trend and risk-controlled behavior.



| Analysis Results for model_final | Metric Value |
|---|---|
| Sharpe Ratio | 1.041317833411686 |
| Max Drawdown | -550.1123 |
| Profit Factor | 1.0211159 |
| Total Equity Change | 641.19336 |
| Equity Change (%) | 6.411934 |
| Total Trades | 843 |
| Long Trades Count | 422 |
| Short Trades Count | 421 |
| Total Trades Returns | 641.19336 |
| Average Trade Return | 0.760609 |
| Average Trade Return (%) | 7.656327e-05 |
| Average Trade Duration (hours) | 6.727494398312903 |
| Max Winning Streak | 19 |
| Max Losing Streak | 6 |
| Total Winning Rate (%) | 63.70106761565836 |
| Average Spread | 5.528559e-05 |
| Estimated Total Spread Cost | 444.59528 |
| Total Rewards | 0.06216526 |
| Average Rewards | 3.6943757e-06 |
| Average Positive Rewards | 0.0003729808 |
| Average Negative Rewards | -0.0003776928 |

Figure 6: Final evaluation metrics for Max-Boltzmann agent.

Sharpe and profit factors improved steadily over the course of training, with the final evaluation achieving the highest stability and reward efficiency. The agent also executed a balanced mix of long and short trades, suggesting an adaptive policy that responds to market regime shifts.

## 4.4 Learning Curves and Metric Progression

We assess how each strategy evolves during training by tracking the Sharpe Ratio across evaluation checkpoints. While all agents followed the same training protocol, their learning trajectories diverged notably.

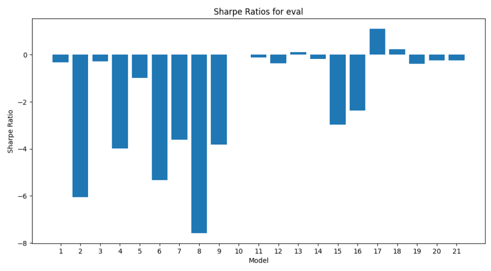


Figure 7: Sharpe Ratio progression — Epsilon-Greedy. High variance and no stable trend.

The Epsilon-Greedy agent exhibited noisy and unstable performance, oscillating between profitable and loss-making policies without converging. Despite occasional peaks, no sustained learning was observed.

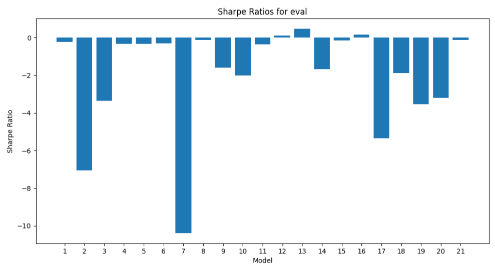


Figure 8: Sharpe Ratio progression — Boltzmann. Slight improvement, but high volatility persists.

Boltzmann exploration showed mild upward movement in Sharpe Ratio early on, but failed to consolidate gains, with later episodes regressing or plateauing.

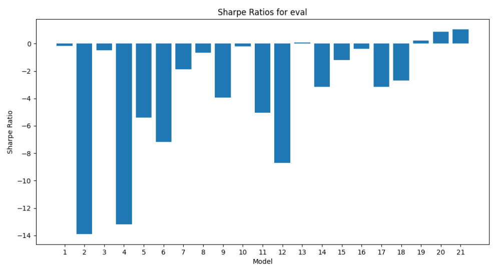


Figure 9: Sharpe Ratio progression — Max-Boltzmann. Clear and consistent learning trend.

In contrast, the Max-Boltzmann agent showed consistent improvement, eventually stabilizing at a positive Sharpe Ratio. This confirms its superior ability to learn and retain profitable trading behavior across episodes.

## 4.5 Summary and Interpretation

The empirical results highlight the advantages of structured exploration in non-stationary environments:

- **Epsilon-Greedy** yielded poor learning signals and high variance, likely due to under-exploration of nuanced trade opportunities.
- **Boltzmann** offered slightly better trade distribution but failed to translate into performance gains.
- **Max-Boltzmann** achieved clear, stable improvement across metrics, demonstrating the value of blending stochasticity and exploitation.

In summary, the Max-Boltzmann strategy not only outperformed both baselines in absolute metrics but also exhibited desirable convergence patterns over time. These findings suggest that hybrid exploration policies may offer a practical advantage in real-world financial RL settings.

Additional plots, including action histograms and episode-wise breakdowns, are available in Appendix C.

## 5 Discussion

The results presented in Section 4 offer empirical evidence of how exploration strategies critically influence learning behavior in RL-based financial agents. This section interprets those results in light of RL theory and financial decision-making, and reflects on broader methodological and practical implications.

### 5.1 Exploration in Financial Contexts

A central insight of this work is that exploration strategies interact differently with financial environments than with typical benchmark tasks (e.g., Atari or robotics). In financial domains, many actions can result in severe and irreversible penalties—such as capital loss or drawdown—making uniform exploration strategies especially hazardous.

The Epsilon-Greedy strategy, though popular, exemplifies this risk. Its uniform sampling from all non-greedy actions treats the second-best action the same as the worst one. In the context of trading, where certain actions (e.g., opening a short position in an upward-trending market) can be catastrophically wrong, such uninformed randomness leads to unstable learning and erratic performance.

In contrast, the Max-Boltzmann strategy mitigates this weakness by attaching meaningful probabilities to each action through softmax sampling over Q-values. Poorly ranked actions receive exponentially less probability mass, thereby reducing the likelihood of collapse-inducing decisions. This property is particularly valuable in financial RL, where:

- The action space is small but asymmetric,
- The cost of poor decisions can dominate long-term returns,
- Recovery from missteps is non-trivial due to position irreversibility and compounding losses.

These challenges reflect structural features of financial environments that depart from standard RL assumptions, such as the Markov property and ergodicity. As highlighted by Hambly et al. [4], financial tasks often involve partial observability, regime switches, and noisy, path-dependent outcomes—all of which amplify the importance of informed and cautious exploration.

This result suggests that **hybrid strategies like Max-Boltzmann are not merely marginally better—they may be fundamentally more appropriate** for domains where action missteps carry disproportionate penalties.

### 5.2 Empirical Interpretation of Results

The Max-Boltzmann agent outperformed its counterparts across Sharpe Ratio, Profit Factor, and equity growth. Its convergence trajectory was both smoother and more stable, suggesting better policy generalization. The stochasticity introduced by Boltzmann sampling allowed it to explore meaningfully, while the fallback to greedy selection preserved performance under uncertainty.

By contrast, the Boltzmann-only strategy struggled to maintain stable returns. Despite its value-aware formulation, it lacked the decisive exploitation behavior needed to consolidate learning in later stages. This supports the interpretation that **exploration alone is not sufficient—exploration must also be risk-aware and anchored by strong exploitation mechanisms**.

Epsilon-Greedy, while occasionally showing intermediate gains, produced erratic behavior and collapsed to nearly inactive policies in some cases. The agent's indifference to action quality during exploration phases led to excessive volatility and poor final performance, despite prolonged training.

### 5.3 Methodological Reflections

This study deliberately fixed all variables unrelated to exploration—such as architecture, reward structure, and features—to ensure that outcome differences were solely attributable to the strategy under test. This constraint strengthened the validity of comparisons but may have limited the agent's capacity to co-adapt more holistically to market dynamics.

The choice to evaluate each checkpoint separately also proved informative. Several agents showed mid-training peaks, underscoring the importance of tracking learning progression over time, rather than relying solely on final models. This is especially relevant in non-stationary domains where the environment itself may shift during training.

### 5.4 Key Takeaways Across Strategies

- **Max-Boltzmann** exhibited the most consistent and profitable behavior. Its softmax-driven stochasticity protected against catastrophic choices while maintaining directed exploration. The results align with prior findings on hybrid exploration improving robustness [13].
- **Boltzmann** failed to deliver meaningful improvements over Epsilon-Greedy. Its inability to consolidate gains highlights the need for balanced exploitation in addition to exploration.
- **Epsilon-Greedy** produced unstable results. Its naive random exploration exposed the agent to high-risk decisions, confirming that **uniform randomness is particularly ill-suited to asymmetric environments like trading**.

### 5.5 Broader Implications

The findings of this study suggest that **domain-aware exploration strategies** may offer advantages in high-uncertainty environments such as financial markets. In particular, strategies that assign action probabilities based on value estimates—rather than selecting randomly among all non-greedy options—appear to mitigate risk more effectively and produce more stable policies.

While this intuition is grounded in the context of trading, it may extend to other domains with similar characteristics: environments where actions can have irreversible consequences, where risk is asymmetric, or where poor decisions can lead to severe penalties. Examples include medical treatment planning, autonomous vehicle control, and resource allocation under uncertainty.

These results do not imply that hybrid strategies like Max-Boltzmann are universally superior, but rather that their structural properties warrant further exploration in domains where naive exploration could be especially costly. RL practitioners may benefit from evaluating such strategies in settings where safety, interpretability, and stability are key concerns.

## 6 Responsible Research

This section outlines the ethical considerations and reproducibility practices adopted in this study, in accordance with the principles of transparent, responsible, and replicable scientific research.

### 6.1 Ethical Considerations

The use of reinforcement learning (RL) agents in algorithmic trading presents several ethical challenges. While the agents developed in this study were created solely for academic purposes and evaluated exclusively in backtesting settings, the potential for real-world application warrants careful reflection.

Algorithmic trading systems can impact market dynamics by increasing volatility, exacerbating liquidity fragmentation, or creating unfair advantages when deployed without sufficient oversight. In particular, RL-based agents—when trained in non-stationary or poorly understood environments—may learn unstable or risky policies that could lead to significant financial losses, or contribute to systemic market instability if scaled improperly.

To mitigate such risks, this research emphasizes that RL agents trained under experimental assumptions—such as perfect execution, no market impact, and idealized bid-ask spreads—should not be considered deployable without extensive validation in production-grade environments. Deployment would require robust safeguards, including:

- Rigorous stress testing across volatility regimes,
- Transparent documentation of risk controls and decision boundaries,
- Alignment with regulatory and compliance frameworks.

This study does not involve any personal, private, or user-generated data. All experiments are conducted using publicly available historical Forex data from the Dukascopy API. The dataset includes only price and volume information and is ethically neutral. Furthermore, no sentiment analysis, social data, or external signals were used, eliminating potential bias from exogenous sources.

### 6.2 Reproducibility

Ensuring reproducibility was a central goal of this work. All code, experiments, and analyses were implemented in Python using open-source libraries such as `gymnasium`, `numpy`, and `stable-baselines3`, with modular design enabling reusability and auditing.

The following reproducibility practices were implemented:

- All hyperparameters—including those governing exploration strategy, learning rate, reward function, and training duration—are explicitly documented in Section 3.
- A fixed random seed was used for all stochastic components to facilitate deterministic runs and result replication.
- No future data leakage is present: all features are computed sequentially and strictly adhere to causal ordering.
- Model checkpoints are saved after every training episode to enable per-episode evaluation and performance tracking.
- Financial metrics such as Sharpe Ratio and Maximum Drawdown are computed using standard definitions detailed in Appendix B.
- All pipeline components—from feature engineering to training—are encapsulated in independently testable modules.

To ensure focused analysis, all experimental conditions unrelated to the core exploration strategy (e.g., environment design, architecture, reward formulation) were held constant. This controlled setup enhances interpretability but limits the scope of optimization—a trade-off further discussed in Sections 5 and 7.

All source code, configuration files, and analysis notebooks are available on GitHub [9]. A reproducibility checklist is provided in Appendix D.2 to guide future researchers.

## 7 Conclusions and Future Work

This thesis investigated how different exploration-exploitation strategies influence reinforcement learning (RL) performance in a dynamic Forex trading environment. Specifically, it evaluated Epsilon-Greedy, Boltzmann, and Max-Boltzmann strategies within a unified and reproducible training framework. All other components—reward function, state representation, model architecture—were fixed to isolate the impact of exploration.

### 7.1 Conclusions

Several key insights emerge from this study:

- **Exploration is a core design decision.** The choice of exploration strategy significantly shaped agent behavior, learning stability, and financial performance. Max-Boltzmann consistently outperformed the other methods, achieving superior Sharpe ratios, smoother equity curves, and lower drawdowns.
- **Weighted exploration prevents catastrophic mistakes.** A critical insight of this work is that in domains like trading—where selecting a poor action (e.g., a badly-timed trade) can lead to large and irreversible losses—simple exploration mechanisms such as Epsilon-Greedy may be ill-suited. Because they assign equal probability to all actions during exploration, they risk sampling catastrophic ones too often. In contrast, hybrid strategies like Max-Boltzmann apply a softmax distribution that emphasizes better actions even when exploring. This nuanced weighting appears vital in environments where the difference between the worst and second-best actions is large and consequential.
- **Intermediate evaluation matters.** Analyzing performance across saved checkpoints showed that final models were not always the best. Some agents peaked mid-training, highlighting the need for periodic evaluation

and model selection based on learning curves, not just endpoint metrics.

Taken together, the findings suggest that hybrid exploration is especially effective in high-risk domains like trading, where naive or flat exploration can expose agents to ruinous decisions.

## 7.2 Limitations

While this research carefully isolated exploration strategies, several constraints affect generality:

- **Frozen pipeline.** By fixing reward functions, input features, and network architectures, this study eliminated many confounding variables but also limited overall performance and flexibility.
- **Non-Markovian structure.** Like most financial environments, the Forex market violates Markov assumptions. Latent factors—such as macroeconomic events or institutional order flows—are unobserved, reducing the effectiveness of standard RL formulations.
- **Simple reward signals.** The agent was rewarded solely on log equity deltas, ignoring factors such as trade duration, drawdown risk, or volatility-adjusted returns. This may have incentivized overly aggressive behavior.

These limitations should be addressed in future extensions that aim to improve generalization or production viability.

## 7.3 Contributions

Despite its constraints, this work makes several notable contributions:

- **A custom RL trading environment**, including a discrete action space, bid-ask spread modeling, and modular reward pipeline.
- **A reproducible experimental pipeline**, supporting per-episode checkpointing, batch evaluation, and per-strategy comparison.
- **A new interpretation of exploration strategy effectiveness**, demonstrating that weighted hybrid methods like Max-Boltzmann can substantially reduce the risk of catastrophic trades while still enabling policy improvement.
- **Early implementation of curiosity and NoisyNet exploration strategies**, available in code form for future tuning and investigation.

## 7.4 Future Work

Building on this foundation, several promising directions remain:

- **Joint optimization.** Future experiments can tune exploration in tandem with reward shaping, state design, or architecture—e.g., adding attention mechanisms or memory models (e.g., LSTM).
- **Risk-aware reward functions.** Introduce asymmetric or utility-based reward signals that account for drawdown sensitivity, regime switching, or risk-adjusted performance.
- **Handling market uncertainty.** Explore partially observable or Bayesian RL techniques, or incorporate tools like Thompson Sampling and UCB to adapt exploration based on epistemic uncertainty.
- **Cross-asset evaluation.** Validate the generality of Max-Boltzmann on other currency pairs, commodities, or stocks under varied volatility regimes.
- **Market impact simulation.** Extend the environment with multi-agent or order book mechanics to study strategy robustness under realistic trading constraints.
- **Fine-tuning advanced strategies.** This work also provides early implementations of two additional exploration strategies—**Noisy Networks** and **Curiosity-Driven Exploration**—which, although not yet yielding strong results, are included for future development. Noisy Networks introduce trainable noise into the Q-network's weights to produce stochastic policies without relying on external schedules. Curiosity-driven exploration rewards the agent for visiting novel or unpredictable states, based on internal prediction error. The modules `noisy_dqn.py`, `curiosity.py`, and `train_eval.py` offer modular starting points for extending and tuning these approaches in financial RL settings.

In summary, this study shows that exploration is not merely a hyperparameter tweak—it is a structural decision. In domains like trading, where bad actions can be disastrous, hybrid strategies that combine curiosity with caution—such as Max-Boltzmann—offer a compelling path forward. This balance may well be the key to stable and profitable RL-based financial agents.

## A Mathematical Definitions of Engineered Features

This appendix provides the precise mathematical definitions of all features used by the agent, grouped by category.

### A.1 Market Features

Let $P_t$ be the closing bid price at timestep $t$.

**Price Momentum.**

$$\text{Momentum}_1(t) = \frac{P_t - P_{t-1}}{P_{t-1}}$$
$$\text{Momentum}_5(t) = \frac{P_t - P_{t-5}}{P_{t-5}}$$

**Exponential Moving Averages (EMA).** Let $\text{EMA}_k(P)_t$ be the EMA of $P$ at time $t$ over a span $k$ (calculated recursively). We define normalized EMAs as:

$$\text{EMA}_{\text{norm},20}(t) = \frac{\text{EMA}_{20}(P)_t}{P_t} - 1$$
$$\text{EMA}_{\text{norm},50}(t) = \frac{\text{EMA}_{50}(P)_t}{P_t} - 1$$

**Relative Strength Index (RSI).** Using the standard RSI calculation over window $N = 14$:

$$\text{RSI}_{14}(t) = 100 \cdot \left(1 - \frac{1}{1 + \frac{\text{AvgGain}_{14}(t)}{\text{AvgLoss}_{14}(t)}}\right)$$

To normalize it:

$$\text{RSI}_{\text{norm}}(t) = 2 \cdot \left(\frac{\text{RSI}_{14}(t) - 0}{100 - 0}\right) - 1$$

### A.2 Agent-State Features

Let: - $C_t$: cash at timestep $t$ - $S_t$: number of shares held at $t$ - $B_t$, $A_t$: bid and ask prices at $t$

**Equity.**

$$E_t = C_t + S_t \cdot \begin{cases} B_t & \text{if } S_t \geq 0 \\ A_t & \text{if } S_t < 0 \end{cases}$$

**Current Exposure.**

$$\text{Exposure}_t = \frac{E_t - C_t}{E_t}$$

**Target Exposure.**

$$\text{Target}_t \in \{-1, 0, +1\} \quad \text{(based on action)}$$

**Trade Duration.** Let $d_t$ be the number of steps since the last change in target exposure. Then:

$$\text{Duration}_t = \frac{d_t}{d_{\text{max}}}$$

where $d_{\text{max}}$ is a fixed normalization constant or the maximum episode length.

## B Evaluation Metrics

This appendix defines the primary evaluation metrics used in this study.

**Sharpe Ratio.** The Sharpe Ratio measures the agent's average return relative to its volatility:

$$\text{Sharpe} = \frac{\mathbb{E}[r_t]}{\sigma[r_t]}$$

where $r_t$ is the per-step reward (or return), $\mathbb{E}[\cdot]$ is the mean, and $\sigma[\cdot]$ is the standard deviation. A higher Sharpe ratio indicates more consistent returns.

**Maximum Drawdown.** Maximum drawdown (MDD) quantifies the worst historical loss from peak equity:

$$\text{MDD} = \min_t \left(\frac{E_t - \max_{\tau \leq t} E_\tau}{\max_{\tau \leq t} E_\tau}\right)$$

where $E_t$ is the equity (portfolio value) at time $t$.

**Total Equity Change (%).** This metric measures the relative change in portfolio equity over the evaluation horizon:

$$\Delta E_{\text{total}} = \frac{E_T - E_0}{E_0} \cdot 100\%$$

where $E_0$ is the initial equity and $E_T$ is the final equity.

**Profit Factor.** Profit Factor is the ratio of cumulative gains to cumulative losses:

$$\text{PF} = \frac{\sum_{i \in \mathcal{W}} r_i}{\left|\sum_{j \in \mathcal{L}} r_j\right|}$$

where $\mathcal{W}$ and $\mathcal{L}$ are the sets of winning and losing trades respectively.

**Winning Rate (%).** Winning rate is the proportion of trades with positive return:

$$\text{Win Rate} = \frac{|\{i : r_i > 0\}|}{|\{i : r_i \neq 0\}|} \cdot 100\%$$

where $r_i$ is the return of trade $i$.

## C Additional Visualizations

While the main results section presents core performance figures, the following supplementary plots offer deeper insight into each agent's behavioral patterns and market interactions. Visualizations are grouped by exploration strategy.

### C.1 Epsilon-Greedy

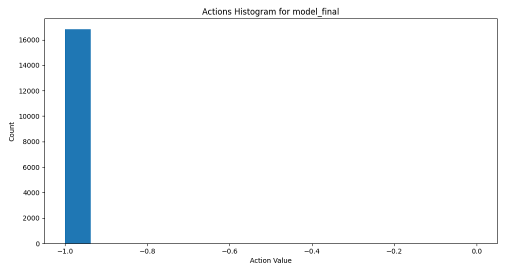


Figure 10: Action distribution histogram for Epsilon-Greedy agent. Long trades dominate behavior.

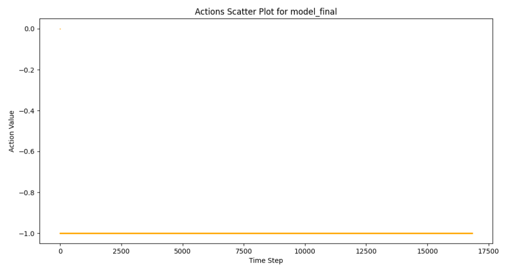


Figure 11: Action selection timeline for Epsilon-Greedy agent. Displays irregular policy switching.

### C.2 Boltzmann

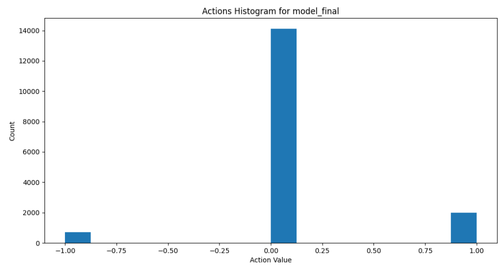


Figure 12: Action distribution histogram for Boltzmann agent. Shows more balanced action spread.

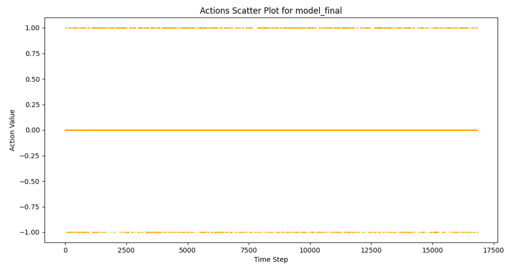


Figure 13: Action selection timeline for Boltzmann agent. Reveals reactive, less structured patterns.

### C.3 Max-Boltzmann

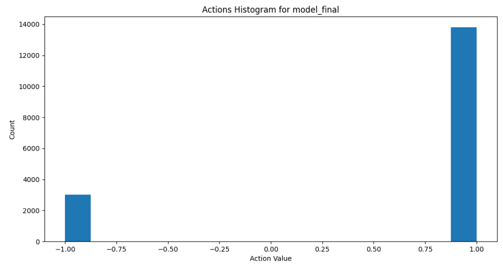


Figure 14: Action distribution histogram for Max-Boltzmann agent. Balanced strategy across all actions.

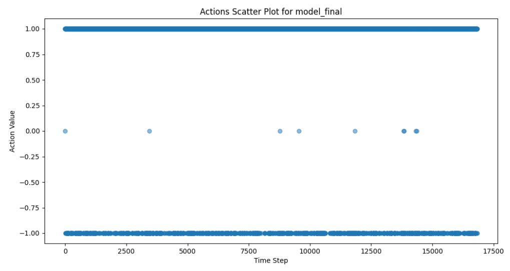


Figure 15: Action selection timeline for Max-Boltzmann agent. Suggests stable and cyclical policy.

*Note:* The histogram shows the number of times each action was taken across all timesteps. In contrast, the trade statistics in the main text group consecutive identical non-zero actions into a single trade. For example, 50 consecutive +1 actions correspond to 1 long trade but 50 histogram counts. This distinction explains why the histogram totals are higher than the number of trades reported in the analysis table.

### C.4 Market Context

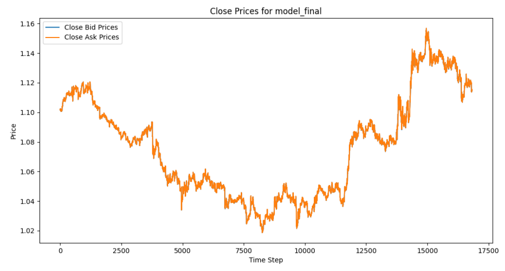


Figure 16: Close prices of EUR/USD market during evaluation. Shared across all agent evaluations.

## D Research Integrity and Reproducibility

### D.1 Use of Generative AI

During this project, generative AI tools (specifically ChatGPT by OpenAI) were used in a responsible and supportive capacity throughout the writing and development process. Their role included:

- Paraphrasing and editing text to improve clarity, fluency, and academic tone;
- Refining structure, transitions, and formatting to align with research writing standards;
- Providing second-opinion validation on technical choices, phrasing, and analytical interpretations;
- Suggesting alternative formulations or perspectives for discussion points, based on user-provided context.

All core scientific content—including the problem formulation, experimental design, implementation, and empirical analysis—was developed independently by the author. No original ideas, hypotheses, or research directions were generated by AI. Instead, generative AI served as an assistive tool to refine articulation, validate reasoning, and improve the professionalism of the written work.

### D.2 Reproducibility Checklist

This checklist summarizes the key practices followed to ensure reproducibility:

- **Environment:** Modular Gym-compatible trading environment with strict chronological feature computation (no future leakage).
- **Codebase:** Modular structure for training, evaluation, and feature engineering. All components are cleanly separated and independently testable.
- **Determinism:** Fixed random seed applied to all training and evaluation routines.
- **Hyperparameters:** All learning and exploration parameters are explicitly reported in the Methods section.
- **Model Checkpoints:** Saved after every training episode, enabling stepwise performance analysis and external verification.
- **Evaluation:** Metrics computed using standard financial definitions (Appendix B); evaluation is decoupled from training.
- **Access:** Full source code, configuration files, and analysis notebooks are available at [9].

## References

[1] Bank for International Settlements. *Triennial Central Bank Survey: Foreign Exchange Turnover in April 2022*. Bank for International Settlements, October 2022.

[2] João Carapuço, Rui Neves, and Nuno Horta. Reinforcement learning applied to forex trading. *Applied Soft Computing*, 73:783–794, Dec 2018.

[3] Pratik Ghosh, Ariel Neufeld, and Jajati Keshari Sahoo. Forecasting directional movements of stock prices for intraday trading using lstm and random forests. *arXiv preprint arXiv:2004.10178*, 2021.

[4] Ben Hambly, Renyuan Xu, and Huining Yang. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 2023.

[5] Chien-Yi Huang. Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787*, 2018.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey, 1996.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb 2015.

[8] Tidor-Vlad Pricope. Deep reinforcement learning in quantitative algorithmic trading: A review. *arXiv preprint arXiv:2106.00123*, 2021.

[9] Radu Serban, Robert Mertens, Finn van Oosterhout, Justas Bertasius, and Yavuz Hancer. Tud-cse-rp-rlinfinance: Reinforcement learning for trading in forex markets. https://github.com/Fo3nix/TUD-CSE-RP-RLinFinance.git, 2025.

[10] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

[11] Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization, 2020.

[12] Thomas Théate and Damien Ernst. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173, 2021.

[13] Valentina Zangirolami and Matteo Borrotti. Dealing with uncertainty: balancing exploration and exploitation in deep recurrent reinforcement learning, 2024.