The use of Reinforcement Learning in Algorithmic Trading

What are the impacts of different possible reward functions on the ability of the RL model to learn, and the performance of the RL Model?

Justas Bertašius
Supervisor(s): Neil Yorke-Smith, Amin Sharifi Kolarijani, Antonis Papapantoleon
EEMCS, 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: Justas Bertašius
Final project course: CSE3000 Research Project
Thesis committee: Neil Yorke-Smith, Amin Sharifi Kolarijani, Antonis Papapantoleon, Julia Olkhovskaia

Abstract

Algorithmic trading already dominates modern financial markets, yet most live systems still rely on fixed heuristics that falter when conditions change. Deep reinforcement learning agents promise adaptive decision making, but their behaviour is driven entirely by the reward function - a design choice that remains poorly researched. Given the critical importance of reward function design in RL problems, this paper investigates the impact of different choices of reward functions while trading in the Forex market and focusing on the EUR/USD pair. We explore different rewarding methods such as profit-only, risk-adjusted, multi-objective, imitation-learning based, and self-rewarding mechanisms. Overall, we demonstrate the impact of careful reward engineering and whether it can boost performance and efficiency, highlighting reward design as a critical and previously under-examined part of RL for deploying reliable trading models.

1 Introduction

Algorithmic trading now drives much of global market flow, reacting to news and order book changes in milliseconds, crucial in the $7.5 trillion per day Forex market, whose 24/5 cadence and sensitivity to macro events create extreme volatility [2; 13]. Yet most live systems still depend on hand-crafted rules or static supervised models that falter when regimes shift, forcing costly manual re-tuning [10].

Reinforcement learning (RL) offers a more adaptive alternative: by treating trading as a sequential decision task, RL agents optimize long-term rewards and adjust online to changing conditions. Recent surveys and case studies confirm RL's promise as the next step in quantitative finance, particularly for fast-moving markets like Forex [11; 6; 1].

1.1 Motivation

A pivotal and often neglected design choice in RL trading is the reward function, which supplies the learning signal that shapes agent behavior. Rewards can target raw profit, risk-adjusted return, drawdown, Sharpe ratio, or more complex utilities, and this choice strongly affects training stability and eventual performance. Poorly aligned rewards invite over-fitting or unstable policies, yet most studies adopt them heuristically with little justification. To close this gap, we systematically compare profit-based, risk-adjusted, multi-objective, self-rewarding and imitation learning based reward schemes for an RL agent in the Forex market, aiming to establish practical guidelines for robust, interpretable, and effective reward design.

1.2 Research questions

The main research question is: **What are the impacts of different possible reward functions on the ability of the RL model to learn, and the performance of the RL Model?** It is divided into four sub questions:

SQ1: Does substituting PnL type reward functions with risk-adjusted ones yield better results?

SQ2: Does using multi-objective rewards (weighted combination of profit, risk, transaction costs and drawdown penalty) improve the perfor- mance of the model and how do these components of the multi-objective reward function impact the performance?

SQ3: Does imitation learning [5] helps to compute a reward function which would improve the performance of the model?

SQ4: Does a self-rewarding mechanism [8] improve adaptability of the model, thus resulting in better results?

1.3 Related Work

Early work by Moody and Saffell [12] showed that RL can learn profitable trading policies, sparking a line of research that replaces hand-crafted rules with sequential decision agents. Deep learning has since enabled end-to-end RL frameworks: Jiang et al. [9] optimise multi-asset portfolios, Tsantekidis et al. [18] exploit micro-price momentum, and Huang [7] casts trading as a Markov game.

Reward engineering.
Recent studies stress that the reward signal, more than network design, dictates robustness. Cornalba et al. [4] use multi-objective blends; Rodinos et al. [15] embed Sharpe ratio in the reward; Goluža et al. [5] stabilise learning with imitation signals; and Huang et al. [8] introduce a self-rewarding network that adapts online. Direct comparisons across reward types, however, remain scarce.

RL in Forex.
Carapuço et al. [3] demonstrated RL outperformance on intraday currency pairs; Silva and Rodrigues [17] added risk overlays; Marimuthu [10] highlighted liquidity advantages for RL agents in FX.

Surveys.
Hambly et al. [6] review sample-efficient RL in finance; Bai et al. [1] and Napate et al. [13] argue that RL mitigates the brittleness of rule-based systems in evolving markets.

Gap.
Collectively, these works show RL's promise but also reveal an ad-hoc approach to reward design. Our study closes this gap by benchmarking profit, risk, multi-objective, imitation, and self-reward formulations in a unified Forex setting, analysing their impact on learning stability and trading performance.

1.4 Contributions

This research aims to advance the understanding and practical application of reinforcement learning (RL) in algorithmic trading within the foreign exchange (Forex) market, with a

specific focus on the role of reward function design. The key contributions of this work are as follows:

- **Systematic reward study:** Side-by-side evaluation of profit, risk-adjusted, multi-objective, self-reward, and imitation rewards.
- **Real-world FX benchmark:** High-frequency EUR/USD data for realistic agent testing.
- **Rich metrics:** Returns plus risk and behavioural stability indicators.
- **Open framework:** Fully reproducible code, templates, and tools.
- **Design rules:** Practical guidelines for choosing reward signals in RL trading.

1.5 Paper Structure

The remainder of the paper is organised as follows. Section 2 reviews background of the research. Section 3 details the experimental method. Sections 4 and 5 report and discuss the results. Section 6 addresses ethical considerations, and Section 7 concludes, states possible improvements and current limitations.

2 Background

Before conducting this research we had to cover many aspects of algorithmic trading, reinforcement learning and also formally describe the problem.

2.1 Algorithmic Trading

Algorithmic trading [13] uses software to scan real-time data and submit orders at latencies unreachable by humans. Traditional rule-based or supervised systems assume stationary markets and struggle in the noisy, regime-shifting Forex arena, where macro-news and policy shocks cause rapid price swings. Adaptive methods are therefore essential. Core attributes of algorithmic trading are:

- **Speed** – Sub-millisecond execution.
- **Scalability** – Single algorithmic trading program (one set of rules and parameters running in software) can monitor and trade many assets simultaneously.
- **Consistency** – Rule-driven decisions free of emotion.
- **Backtesting** – Strategies validated on historical data before deployment.

2.2 Reinforcement Learning

Reinforcement learning (RL) trains an *agent* to interact with an *environment*, choosing actions that maximise long-run reward via trial and error. The formalism is a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ [14]:

- $\mathcal{S}$ – state space, $\mathcal{A}$ – action set
- $P(s'|s,a)$ – transition probability
- $R(s,a)$ – immediate reward
- $\gamma \in [0,1]$ – discount factor

At each step t the agent in state s_t selects a_t, receives $r_t = R(s_t, a_t)$, and observes $s_{t+1} \sim P(\cdot|s_t, a_t)$. It seeks a policy $\pi(a|s)$ that maximizes expected discounted return

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

Key RL elements:

- **Agent** – decision-maker
- **Environment** – market simulator
- **State** s_t – feature vector at t
- **Action** a_t – trade (e.g. Buy, Sell, Hold)
- **Reward** r_t – profit, risk metric, etc.

RL is well suited to sequential, non-stationary tasks such as high-frequency Forex trading, where agents must adapt online and optimize long horizon objectives [6; 1].

2.3 Formal Problem Description

The trading problem is cast as a discrete-time MDP. At each step t, the agent observes a feature vector $o_t \in \mathbb{R}^n$, chooses an action $a_t \in 0 : \text{Buy}, 1 : \text{Sell}, 2 : \text{Hold}$, receives a reward r_t, and the environment moves to s_{t+1}. The goal is to learn a policy $\pi_\theta(a|s)$ that maximises the discounted return $\sum_t \gamma^t r_t$.

Focus on reward design.
All experiments keep data, features, and a fixed DQN constant; only the reward function varies: profit, risk-adjusted, multi-objective, self-reward, and imitation. We evaluate how each choice affects learning stability and convergence, risk-taking behavior and final trading performance.
The study thus isolates the impact of reward engineering in noisy, non-stationary Forex markets, offering practical guidance for future RL trading systems.

3 Methodology

This section details the experimental workflow designed to answer our core research question: *What are the impacts of different possible reward functions on the ability of the RL model to learn, and the performance of the RL Model?* All source code, data pipelines, and configuration files are open-sourced for full reproducibility.

3.1 Data Acquisition and Pre–trade Processing

Market Feed and Time Horizon
Historical quotations for the EUR/USD currency pair were downloaded from the **Dukascopy** public archive [16]. Ticks covering the period *2 Jan. 2022 – 16 May 2025* were retrieved and consolidated into *15-minute* candles. Using tick data rather than pre-aggregated bars guarantees that the resulting series is internally consistent (open, high, low, close, volume) and free from vendor-side rounding artifacts. The data was then split into training and evaluation sets with 0.7 and 0.3 ratios respectively. Training data can be seen in Figure 1 and evaluation data can be seen in Figure 2.

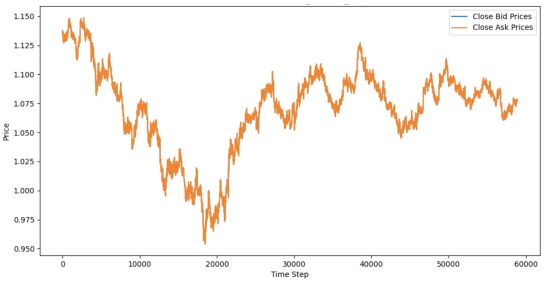

Figure 1: Market train data

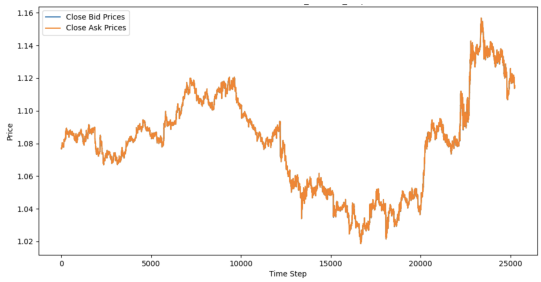

Figure 2: Market evaluation data

Cleaning Pipeline

a) **Holiday & weekend removal.** Intervals in which the bid, ask, high and low were identical or trading volume was zero are discarded.

b) **Aggregation.** Remaining ticks are resampled to 15-minute OHLCV records with volume taken as the sum of tick volumes.

c) **Gap handling.** Fewer than 2% of candles are absent after aggregation (three weeks out of three years). Missing bars are left *blank*—no forward–fill or interpolation is applied—to avoid leaking artificial structure that a learning agent might exploit.

Feature Construction

Two independent stages generate inputs for the RL agent:

a) **Static market features** Computed once for the entire dataset.

- $\Delta p_{t,1}$ and $\Delta p_{t,5}$ — one- and five-bar log returns,
- normalised RSI_{14},
- relative trend strength $\frac{\mathrm{EMA}_{20}}{p_t}$, $\frac{\mathrm{EMA}_{50}}{p_t}$.

b) **Dynamic agent features** Re-evaluated each step.

- cash-to-equity ratio,
- current exposure $(-1, 0, +1)$.

All real-valued columns are z-score normalised on the training split; the learned mean and variance are reapplied to the validation and test partitions to preclude look-ahead bias.

3.2 Trading Environment

A custom newly created `gymnasium` environment, `ForexEnv`, emulates leveraged spot trading under realistic frictions while retaining analytical clarity.

State and Observation Space

The full *state* maintained by the environment includes all market data and agent portfolio information (e.g., prices, volumes, cash, positions, equity). However, the agent is provided only with a partial *observation* $\mathbf{o}_t$ at each step, which includes just the information needed to make decisions, in accordance with the Markov assumption. This observation includes:

1. **Market Features** – Derived from market history (e.g., RSI, log returns).
2. **Agent Features** – Portfolio-related variables computed by the environment (e.g., cash-to-equity ratio), not internally tracked by the agent.

The full observation is obtained by concatenating these two components,

$$\mathbf{o}_t = \big[\ \mathrm{MarketFeatures}_t \parallel \mathrm{AgentFeatures}_t \big],$$

which supplies all information judged necessary for informed action while remaining a *partial* view of the underlying environment.

Action Specification

A key modelling choice is to let the agent output a *target exposure* rather than an explicit buy–or–sell command. This exposure can be specified in two alternative ways:

- **Continuous scale:** a single real number $a_t \in [-1, 1]$, where $a_t = -1$ denotes a fully short position, $a_t = 1$ a fully long position, and $a_t = 0$ a flat (all–cash) stance.
- **Discrete grid:** a finite set of integer labels that are mapped to evenly spaced exposure levels, e.g. $\{-1.0, -0.5, 0.0, 0.5, 1.0\}$.

Execution Model

At every simulation tick the environment receives the agent's *desired exposure* and determines the trade size needed to rebalance the current portfolio accordingly. Execution then follows a realistic micro-structure defined by four rules:

1. **Transaction fee.** A proportional commission is subtracted on each trade, diminishing sale proceeds or increasing purchase cost.
2. **Leverage cap.** Gross exposure may not exceed 100% of account equity; the agent is barred from entering positions that would require borrowing cash or stock.
3. **Bid–ask spread.** Long entries transact at the prevailing *ask*; short entries at the *bid*, properly accounting for spread costs.
4. **Zero price impact.** Trades are assumed too small to influence market prices. The asset prices evolve independently of the agent's actions, and executing trades does not affect the bid, ask, or mid prices.

Reward Signal
By default the instantaneous reward equals the change in ledger equity,

$$r_t = E_t - E_{t-1},$$

but the environment allows to easily change reward functions (e.g. log return, Sharpe-adjusted, self-reward mechanism, etc.) to be injected without modifying other mechanics. This helps for us to examine how different rewards impact the model's performance in trading.

Episode Termination
An episode finishes either when the candle stream ends (natural truncation) or when equity depletes to zero (bankruptcy termination). Upon termination the complete trajectory—states, actions, rewards and portfolio values—is persisted for later analysis.

Mark-to-Market Accounting
Equity is re-marked at every step:

$$E_t = \text{cash}_t + \text{position}_t \times \text{midprice}_t,$$

where the mid-price is the mean of bid and ask quotes.

The above design reproduces the core micro-structure of real foreign-exchange trading while abstracting away latency, market depth and counter-party risk. Consequently, any observed performance differentials can be attributed to the agent's learning behaviour and, crucially for this study, to the chosen reward formulation rather than to idiosyncratic simulator quirks.

3.3 Agent and Learning Algorithm

All experiments employ the same Deep Q-Network (DQN) to isolate reward-related effects.

- *Network.* Two dense layers, 128 ReLU units each.
- *Hyper-parameters.* Learning rate 10^{-4}, replay buffer 50 000, batch 64, target net sync every 500 steps, $\gamma = 0.99$.
- *Exploration strategy.* Max-Boltzmann exploration strategy is used with epsilon 0.1 and temperature as 1.

3.4 Experimental Design

Five types of reward functions are used:

R1: Profit-based. Net equity change, percentage return, log-equity change.

R2: Risk-adjusted. Sharpe ratio, mean–variance, CVaR adjusted.

R3: Multi-objective. Uses a weighted sum of profit, risk, transaction costs and drawdown penalty.

R4: Self-reward mechanism. Self-rewarding mechanism is constructed and tested.

R5: Imitation learning. Imitation-learning-based function is implemented and tested.

The mathematical definitions and more details about these reward functions can be found in Appendix A. These reward functions are evaluated under identical fixed training settings, using 50 episodes per run (more episodes are not used since the model tends to overfit after approximately 30 episodes or in case there is no convergence at all, the amount of episodes proved to be irrelevant) over 5 different seeds. Intermediate models are saved after each episode to track convergence behavior. These are then individually evaluated on a single test episode to track learning progression. In addition the best performing model over all the episodes is taken and analyzed further with predetermined evaluation metrics 3.5.

3.5 Evaluation Metrics

Each definition is given in closed form for an equity series $\{E_t\}_{t=0}^{T}$ and single-period returns $r_t = \frac{E_t - E_{t-1}}{E_{t-1}}$.

(1) Cumulative Return Shows the total percentage change in the portfolio's value from the previous time step to the current.

$$\text{CR} = \prod_{t=1}^{T}(1 + r_t) - 1.$$

(2) Sharpe Ratio The Sharpe ratio is a measure of risk-adjusted return, indicating how much an investment compensates for the risk taken.

$$\text{SR} = \frac{\overline{r} - r_f}{\sigma_r}, \quad \overline{r} = \frac{1}{T}\sum_{t=1}^{T} r_t, \quad \sigma_r = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(r_t - \overline{r})^2},$$

where r_f is the risk-free rate (set to 0 for intraday analysis).

(3) Maximum Drawdown Measures the maximum fall in the value of the investment, as given by the difference between the value of the lowest trough and that of the highest peak before the trough.

$$\text{MDD} = \max_{1 \le t \le T}\Big(\max_{0 \le u \le t} E_u - E_t\Big).$$

(4) Profit Factor Ratio of gross profit to gross loss across trades.

$$\text{PF} = \frac{\sum_{t:r_t>0} r_t}{\sum_{t:r_t<0} |r_t|}.$$

(5) Win Rate Fraction of profitable trades.

$$\text{WR} = \frac{N_{\text{wins}}}{N_{\text{trades}}}, \quad N_{\text{wins}} = \#\{t \mid r_t > 0\}.$$

(6) Reward Evolution Average reward per episode across training. Episode-level average reward:

$\bar{R}^{(k)} = \frac{1}{L_k}\sum_{t=1}^{L_k} r_t^{(k)}$ for episode k of length L_k. Tracking $\bar{R}^{(k)}$ across k visualises convergence.

(7) Action Distribution Frequency of each action type (long, short, hold). For each action $a \in \{0, 1, 2\}$, $f_a = \frac{1}{T}\sum_{t=1}^{T} 1\{a_t = a\}$ measures policy bias towards long, short, or flat positions.

These metrics help to compare the impact of different types of reward functions and how the performance changes because of them.

4 Results

This section compares five reward-design paradigms—profit-centred, risk-adjusted, multi-objective, self-rewarding, and imitation learning. For each paradigm we plot episode-level learning curves, inspect the equity trajectory of the best model, and report the mean and standard deviation of all key metrics across training seeds.

4.1 Profit-based

Three profit signals were tested: *equity change*, *log-equity change*, and *percentage equity change*. Figure 3 shows that the equity-change agent learns most smoothly, whereas the log- and percentage variants display substantial early-stage volatility. After episode 30, the equity-change curve plateaus, indicating possible over-fitting.

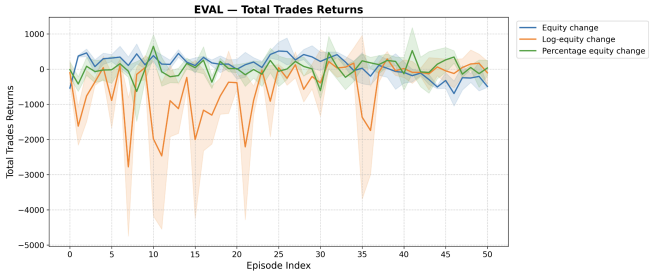

Figure 3: Total trade returns over episodes for profit-based reward variants.

After selecting the top model for each profit signal, we plot their equity curves in Figure 4. The equity-change model is the most stable and best model, even though the profit is slightly less than percentage equity-change model. Its multi-seed statistics are listed in Table 1.

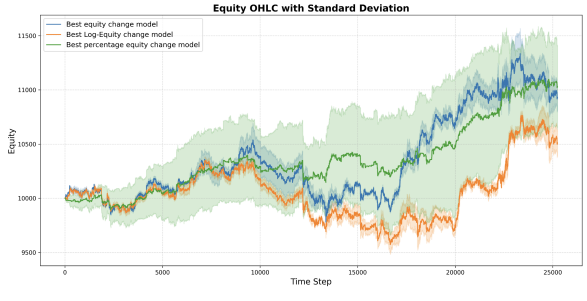

Figure 4: Equity OHLC curves for the best model of each profit-based reward.

Metric	Mean	Std
Sharpe Ratio ↑	1.141	0.174
Max Drawdown ↑	-764.86	108.37
Profit Factor ↑	1.024	0.0038
Equity Change (%) ↑	9.27	1.47
Total Trades Returns ↑	926.89	146.97
Total Trades	186	76
Long Trades Count	92	32
Short Trades Count	94	44
Avg Trade Return ↑	5.67	2.51
Win Rate (%) ↑	66.17	3.69
Avg Rewards ↑	0.037	0.0058

Table 1: Performance metrics for the best equity-change reward model.

4.2 Risk-adjusted

Three risk-aware signals were examined: *CVaR-adjusted*, *Sharpe-adjusted*, and *Sortino-adjusted*, each with a look-back window of 100. Figure 5 reveals that the CVaR agent is highly erratic, whereas the Sharpe- and Sortino-adjusted agents learn more smoothly and exhibit markedly narrower confidence bands.

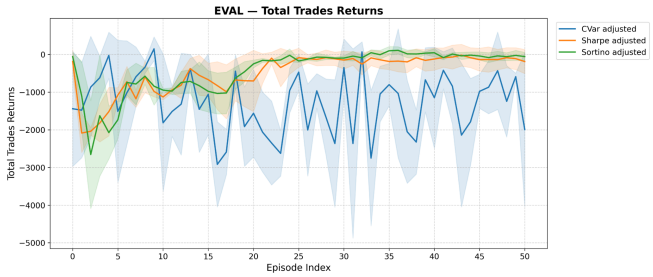

Figure 5: Total trade returns over episodes for risk-adjusted reward variants.

After selecting the top model for each variant, we plot their equity curves in Figure 6. Although the CVaR model occasionally achieves larger returns, its instability makes the Sortino-adjusted model the preferred choice. Multi-seed statistics for this model are compiled in Table 2.

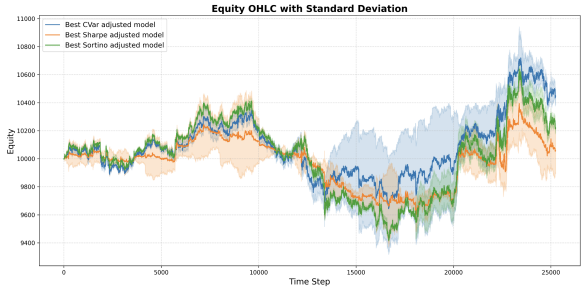

Figure 6: Equity OHLC curves for the best risk-adjusted models.

Metric	Mean	Std
Sharpe Ratio ↑	0.3028	0.0420
Max Drawdown ↑	-1000.53	15.04
Profit Factor ↑	1.0064	0.0005
Equity Change (%) ↑	2.17	0.38
Total Trades Returns ↑	217.03	37.59
Total Trades	273	46
Long Trades Count	224	98
Short Trades Count	49	56
Win Rate (%) ↑	37.62	2.83
Avg Rewards ↑	0.178	0.003

Table 2: Performance metrics for the best Sortino-adjusted model.

Metric	Mean	Std
Sharpe Ratio ↑	0.956	0.279
Max Drawdown ↑	-395.83	38.69
Profit Factor ↑	1.025	0.011
Equity Change (%) ↑	6.58	1.11
Total Trades Returns ↑	658.31	111.08
Total Trades	959	123
Long Trades Count	501	86
Short Trades Count	458	50
Win Rate (%) ↑	65.26	0.88
Avg Rewards ↑	0.0145	0.0087

Table 3: Performance metrics for the best multi-objective model.

4.3 Multi-objective

The multi-objective reward combines profit, risk, transaction-cost, and drawdown penalties using fixed weights of 1:10:1:5. As shown in Figure 7, training is profitable and relatively stable until roughly episode 30, at which point returns deteriorate, suggesting sensitivity to the fixed weight vector.

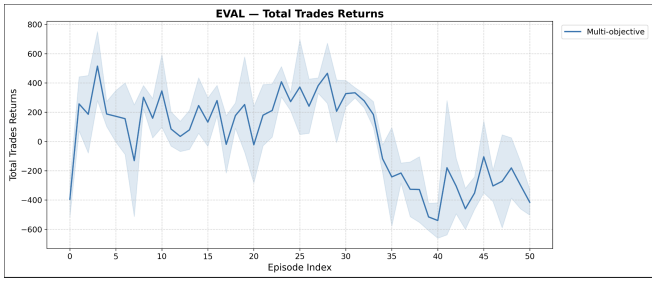

Figure 7: Total trade returns over episodes for the multi-objective reward.

Figure 8 depicts the equity path for the best multi-objective model. Although its cumulative profit is below that of the pure profit agent, drawdowns are substantially lower. Table 3 summarises the corresponding multi-seed statistics.

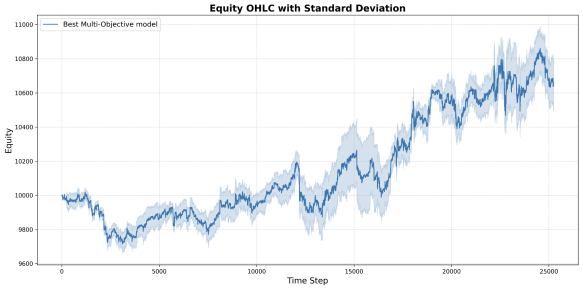

Figure 8: Equity OHLC curve for the best multi-objective model.

4.4 Self-rewarding Mechanism

Three self-reward networks were trained—*min-max*, *return*, and *Sharpe* experts—each with a two-layer MLP (128 / 64 units, ReLU). Figure 9 shows that the min-max agent converges most smoothly; the other two variants remain volatile throughout.

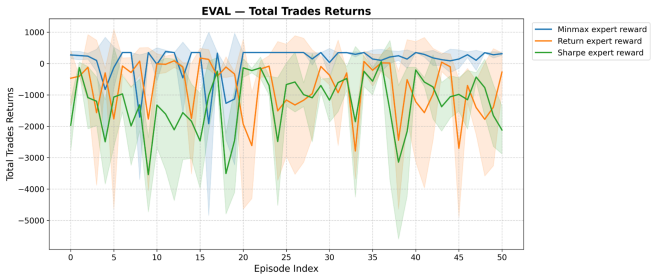

Figure 9: Total trade returns over episodes for self-rewarding variants.

Figure 10 displays equity curves for the best model of each variant. Despite its instability, the return-expert agent delivers the highest terminal equity and is therefore examined in Table 4.

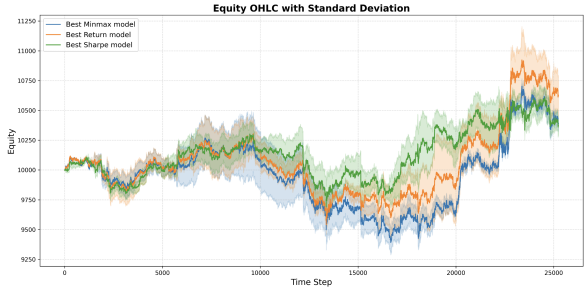

Figure 10: Equity OHLC curves for the best model of each self-rewarding variant.

Metric	Mean	Std
Sharpe Ratio ↑	0.774	0.227
Max Drawdown ↑	-862.07	143.68
Profit Factor ↑	1.016	0.005
Equity Change (%) ↑	6.26	2.00
Total Trades Returns ↑	626.12	200.04
Total Trades	353	589
Long Trades Count	177	294
Short Trades Count	176	293
Win Rate (%) ↑	70.24	3.83
Avg Rewards ↑	0.458	0.074

Table 4: Performance metrics for the best self-reward model (return expert).

Metric	Mean	Std
Sharpe Ratio ↑	-0.3403	0.1175
Max Drawdown ↑	-1150.61	268.44
Profit Factor ↑	0.9939	0.0017
Equity Change (%) ↑	-3.07	1.19
Total Trades Returns ↑	-306.72	122.89
Total Trades	53	65
Long Trades Count	5	5
Short Trades Count	48	60
Win Rate (%) ↑	50.33	0.49
Avg Rewards ↑	-2.87×10^{-6}	3.62×10^{-6}

Table 5: Performance metrics for the best imitation-learning model.

4.5 Imitation Learning

An imitation-learning reward derived from an expert policy [5] was also evaluated. Figure 11 shows large fluctuations during the first 20 episodes, followed by convergence to consistently small, often negative, returns.

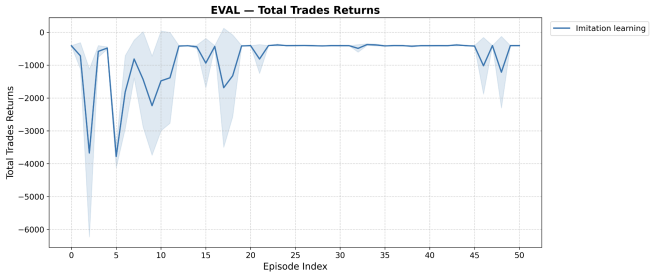

Figure 11: Total trade returns over episodes for the imitation-learning reward.

The equity curve of the best imitation model is given in Figure 12, and its multi-seed metrics appear in Table 5. The negative Sharpe ratio underscores the difficulty of surpassing the expert purely through behavioural cloning.

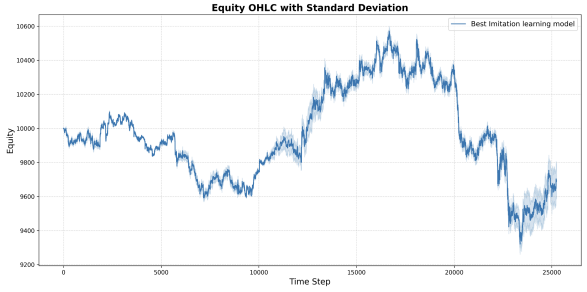

Figure 12: Equity OHLC curve for the best imitation-learning model.

5 Discussion

This section discusses the empirical findings in Section 4, connects them to the four research sub-questions (SQ1–SQ4), and highlights both practical and methodological implications.

5.1 Profit-based versus Risk-adjusted Rewards (SQ1)

Figures 3 and 5 show that both reward types exhibit unstable behavior and it can also be seen that profit-based reward learning process yields much larger returns. Furthermore, if we look at the best models from both of these types it can be observed from Tables 1 and 2 that purely profit-based model is in almost every metric superior yielding higher Sharpe ratio (1.141 to 0.3028), Max Drawdown (-764.86 to -1000.83), Total Trades Returns(926.89 to 217.03), etc. With lower Sharpe ratio it means that risk-adjusted model does not do its job succesfully by minimizing risk to reward ratio. However, it should be noted that standard deviation of risk-adjusted model is smaller for metrics such as Sharpe Ratio, Max Drawdown, Profit Factor, etc. This suggests that risk-adjusted model is more stable and resilient to randomness. It is also more active with it on average making more trades and receiving bigger average rewards per episode. So in conclusion it is generally better to use profit-based reward function unless the user really wants to avoid randomness.

5.2 Effectiveness of Multi-objective Weighting (SQ2)

The multi-objective formulation balances four competing signals (profit, risk, cost, drawdown). Fine-tuned weights (1:10:1:5) produce a model that delivers respectable returns *and* the best max drawdown and also it is the most active among all reward functions with -395.83 Max Drawdown and 959 total trades (Table 3). Nevertheless, Figure 7 shows pronounced deterioration after episode 30, suggesting that the fixed weight vector either cannot accommodate non-stationary regimes or starts overfitting on the training data and is not able to generalize anymore. Hence, while the weighted mixture outperforms single-objective baselines in early training, adaptive or state-dependent weighting may be required for long-run robustness. In conclusion it is an effective reward function which could definitely be considered but the

fine-tuning of hyperparamaters should be considered when choosing.

5.3 Self-rewarding Mechanisms (SQ3)

Among the self-reward variants, the *return-expert* signal emerges as best (Table 4) but still exhibits high-amplitude swings (Figure 9). The internal reward network evidently provides richer feedback than a scalar PnL, yet remains prone to divergence if the expert proxy itself is noisy. The large standard deviations in trade counts (± 590 trades) indicate inconsistent exploration across seeds. These findings partially address SQ3: self-rewarding can match traditional objectives *if* expert proxies are well-behaved, but they do not inherently grant superior stability. In conclusion this reward strategy could have a lot potential but due to its complexity more data might be needed, in depth hyperparameter tuning is required and possibly longer training times could help.

5.4 Imitation-learning Reward (SQ4)

The imitation-based agent converges to a narrow action space and delivers modest, near-zero returns (Figure 11), mirroring Table 5: a negative Sharpe (-0.34) and equity decline (-3.07%). It is also not as active with only an average of 53 total trades of which of, short trades completely dominate which means the model does not provide variety and could lose money because of that. Furthermore the steep early learning curve followed by a flat plateau indicates that the agent quickly saturates its capacity to replicate expert behaviour but cannot devise novel, profitable deviations. Consequently, SQ4 is *not* supported— pure imitation does not outperform direct optimization in this setting.

Pure profit signals still dominate on headline metrics—posting the highest Sharpe, smallest drawdown and largest cumulative return—yet they do so at the cost of pronounced episodic noise. Risk-adjusted rewards shrink that variance but leave money on the table, while multi-objective weights moderate drawdowns only until over-fitting sets in. Self-rewarding shows promise but remains unstable, and imitation-based rewards under-perform outright. These results suggest that the most effective trading reward should keep a strong profit core but temper it with adaptive, context-aware risk penalties rather than relying on any single metric in isolation.

6 Responsible Research

This study adheres to responsible research principles in terms of transparency, reproducibility, and ethical integrity.

- **Reproducibility:** All code, hyperparameter configurations, environment wrappers, and experiment scripts are publicly released in a dedicated repository. This includes version-locked dependencies and a README with step-by-step instructions.
- **Data Integrity:** Historical Forex data were sourced from publicly available, reliable APIs (e.g., Dukascopy), and all transformations are logged to preserve traceability. No synthetic or manipulated data were used in model training or testing.
- **Model Fairness:** The agent was evaluated across multiple seeds and episodes to account for variance. Results were averaged and evaluated with used performance metrics 3.5.
- **Ethical Considerations:** This research does not involve human subjects, personal data, or sensitive financial records. However, we acknowledge that algorithmic trading can influence real-world market dynamics. This study is purely academic and does not endorse using untested RL models in live markets without thorough safeguards.
- **Limitations Disclosure:** As outlined in the discussion, the study excludes transaction costs, latency, slippage, and multi-asset settings. These were omitted to isolate the effect of reward functions but should be addressed in follow-up research before practical deployment.

7 Conclusions and Future Work

This research explored the impact of different reward function formulations on the performance and learning dynamics of reinforcement learning agents in the context of Forex trading. Using a controlled experimental setup, we compared profit-based, risk-adjusted, multi-objective, imitation-based, and self-rewarding mechanisms, isolating their influence on both financial outcomes and training behavior.

Key Takeaways

- **SQ1 (Profit vs. Risk)** – Pure profit rewards produced the highest Sharpe ratio, profit factor, and cumulative return, albeit with greater variance. Risk-adjusted rewards lowered drawdowns and metric dispersion but failed to deliver competitive risk-adjusted returns. *Recommendation*: favour profit-centred objectives unless strict risk ceilings are paramount.
- **SQ2 (Multi-objective)** – A fixed weighted blend of profit, risk, cost, and drawdown outperformed single objectives early on, yet deteriorated after $\sim$30 episodes, signalling weight over-fitting. *Recommendation*: explore adaptive or state-dependent weighting schemes to retain long-run robustness.
- **SQ3 (Self-reward)** – The return-expert network matched baseline profitability but remained volatile and seed-sensitive. Its promise hinges on high-quality expert signals and extensive hyper-parameter tuning. *Recommendation*: combine self-reward with larger data regimes and regularisation to stabilise learning.
- **SQ4 (Imitation Learning)** – Imitation learning based reward converged quickly but plateaued at sub-par, negative-Sharpe performance, indicating limited capacity to innovate beyond the expert. *Recommendation*: use imitation only as a warm-start, then switch to direct optimisation.

Future Work

- **Broader environments**: Add multiple assets, regime shifts, slippage, and fees.
- **Reward–algorithm fit**: Test how each reward pairs with alternative agents, features, and exploration schemes.

- **Hyper-parameter tuning**: Search learning rates, buffer sizes, and reward weights for faster convergence.
- **Advanced architectures**: Compare policy-gradient and actor–critic models under identical rewards.
- **Adaptive rewards**: Let multi-objective weights adjust online to market conditions.
- **Live evaluation**: Deploy agents in paper-trading or realistic simulators to test out-of-sample robustness.

These steps will deepen our understanding of reward engineering and support more reliable RL trading systems.

References

[1] Y. Bai, Y. Gao, R. Wan, S. Zhang, and R. Song. A review of reinforcement learning in financial applications. *Annual Review of Statistics and Its Application*, 12:209–232, 2025.

[2] Bank for International Settlements. Triennial central bank survey: *Foreign exchange turnover in April 2022*. Bank for International Settlements, October 2022.

[3] João Carapuço, Rui Neves, and Nuno Horta. Reinforcement learning applied to forex trading. *Applied Soft Computing*, 2018.

[4] Federico Cornalba, Constantin Disselkamp, Davide Scassola, and Christopher Helf. Multi-objective reward generalization: Improving performance of deep reinforcement learning for applications in single-asset trading. *arXiv preprint*, 2023.

[5] Sven Goluža, Tomislav Kovačević, Stjepan Begušić, and Zvonko Kostanjčar. Robot see, robot do: Imitation reward for noisy financial environments. In *2024 IEEE International Conference on Big Data (BigData)*, pages 4884–4891, 2024.

[6] Ben Hambly, Renyuan Xu, and Huining Yang. Recent advances in reinforcement learning in finance. *arXiv preprint*, 2023.

[7] Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint*, 2018. Cornell University.

[8] Yuling Huang, Chujin Zhou, Lin Zhang, and Xiaoping Lu. A self-rewarding mechanism in deep reinforcement learning for trading strategy optimization. *Mathematics*, 12(24):4020, 2024.

[9] Zhiye Jiang, Shuyue Xu, and Jianjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint*, 2017.

[10] Ganesh Marimuthu. Algorithmic trading in forex markets: The impact of ai-driven strategies on liquidity and market efficiency. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, 2025.

[11] Terry Lingze Meng and Matloob Khushi. Reinforcement learning in financial markets. *Data*, 2019.

[12] John Moody and Mike Saffell. Reinforcement learning for trading. *Advances in Neural Information Processing Systems*, 11:917–923, 1998.

[13] Sachin Napate, Mukul Thakur, and D B-School. Algorithmic trading and strategies. November 2020.

[14] Martijn Otterlo and Marco Wiering. Reinforcement learning and markov decision processes. *Reinforcement Learning: State of the Art*, pages 3–42, 01 2012.

[15] Georgios Rodinos, Paraskevi Nousi, Nikolaos Passalis, and Anastasios Tefas. A Sharpe ratio based reward scheme in deep reinforcement learning for financial trading. In *Artificial Intelligence Applications and Innovations (AIAI 2023)*, volume 675 of *IFIP Advances in Information and Communication Technology*, pages 15–23. Springer, Cham, 2023.

[16] Dukascopy Bank SA. Forex historical data feed, 2025.

[17] R Silva and Camila Eleutério Rodrigues. Optimizing automated trading systems portfolios with reinforcement learning for risk control. *Journal of Bioengineering Technologies and Health*, 2025.

[18] Avraam Tsantekidis, Nikolaos Passalis, Anastasia-Sotiria Toufa, Konstantinos Saitas-Zarkias, Stergios Chairistanidis, and Anastasios Tefas. Price trailing for financial trading using deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7):2837–2846, 2021.

A Reward functions mathematical definitions

Five reward types were explored. Variables are defined as follows: E_t—portfolio equity at step t; $R_t = \frac{E_t - E_{t-1}}{E_{t-1}}$—period return; μ_t and σ_t—mean and standard deviation of the last w returns; CVaR_α—conditional value at risk at confidence α; Cost_t—transaction cost; DD_t—instantaneous drawdown; $\lambda, \{w_i\}$—tunable coefficients; π_θ—agent policy; π_E—expert policy.

R1: **Profit-based**. Reward is tied directly to gains.

Net equity change

$$r_t^{\mathrm{EQ}} = E_t - E_{t-1}.$$

Percentage return

$$r_t^{\%} = \frac{E_t - E_{t-1}}{E_{t-1}}.$$

Log-equity change (additive and numerically stable)

$$r_t^{\log} \;=\; \ln\big(E_t\big) - \ln\big(E_{t-1}\big) = \ln\Big(\tfrac{E_t}{E_{t-1}}\Big).$$

R2: **Risk-adjusted**. Add an explicit risk penalty to the profit signal.

Sharpe-adjusted

$$r_t^{\mathrm{Sharpe}} = \frac{\mu_t}{\sigma_t + \varepsilon}, \qquad \varepsilon \ll 1.$$

Mean–variance

$$r_t^{\mathrm{MV}} = \mu_t - \lambda \sigma_t^2.$$

CVaR-adjusted

$$r_t^{\mathrm{CVaR}} = \mu_t - \lambda\, \mathrm{CVaR}_\alpha(R_{t-w+1:t}).$$

R3: **Multi-objective**. Weighted blend of profit, risk, cost, and drawdown:

$$r_t^{\text{MO}} \;=\; w_1\, r_t^{\text{EQ}} - w_2\, \text{Cost}_t - w_3\, \text{DD}_t - w_4\, \sigma_t,$$

R4: **Self-rewarding mechanism** [8]. A small MLP reward network f_{ϕ} is first fitted to an expert label $g(s, a)$ via MSE:

$$\mathcal{L}(\phi) = \mathbb{E}\big[(f_{\phi} - g)^2\big].$$

During training the agent receives

$$r_t^{\text{SR}} = \max\big(f_{\phi}(s_t, a_t),\; g(s_t, a_t)\big),$$

letting it exploit either the learned or expert reward—whichever is larger—while the policy is updated by standard DQN.

R5: **Imitation-learning reward** [5]. Let r_t^{RF} be the agent's own profit and r_t^{IF} the profit obtained by following an oracle trend label $y_t \in \{0, 1\}$. The reward is their difference:

$$r_t^{\text{IL}} \;=\; r_t^{\text{RF}} \;-\; r_t^{\text{IF}}.$$

Thus the agent receives zero reward when it mimics the oracle, is penalised for worse trades, and is encouraged only when it outperforms the expert.

B The Use of Large Language Models (LLMs)

Large Language Models (LLMs) assistance was used sparingly and responsibly. Its purposes were to help paraphrasing sentences in a more academic and professional way as well as styling the text. In addition they were used to help make plots and tables in LaTex format. Lastly it helped to write mathematical equations and definitions in LaTex.

This output was reviewed, fixed if needed and edited before inclusion in the final version.

No content was generated autonomously without author review and/or edit. All use of LLMs was limited to surface-level editing and phrasing fixes. The scientific content, structure, analysis, and conclusions of the report reflect the independent work of the authors.