Decomposable Reward Modeling and Realistic Environment Design for Reinforcement Learning-Based Forex Trading

Nabeel Ahmad Saidd *

Posted Date: 23 March 2026

doi: 10.20944/preprints202603.1701.v1

Keywords: reinforcement learning; foreign exchange trading; reward decomposition; action masking; pyramiding; martingale scaling

 doi:10.20944/preprints202603.1701.v1

Article

Decomposable Reward Modeling and Realistic Environment Design for Reinforcement Learning-Based Forex Trading

Nabeel Ahmad Saidd

Dr APJ Abdul Kalam Technical University, India; nabeelahmadsaidd@gmail.com

Abstract

Applying reinforcement learning (RL) to foreign exchange (Forex) trading remains challenging because realistic environments, well-defined reward functions, and expressive action spaces are all required simultaneously. Many existing studies simplify these elements through basic simulators, single scalar rewards, and limited action representations, making learned policies difficult to diagnose and limiting practical relevance. This paper introduces a modular RL framework for foreign exchange trading that addresses these limitations. The framework comprises three components. First, a *friction-aware execution engine* enforces strict anti-lookahead semantics—observations are taken at close$_t$, orders are executed at open$_{t+1}$, and positions are marked-to-market at close$_{t+1}$—while incorporating realistic transaction costs, including spread, commission, slippage, rollover financing, and margin-triggered liquidation. Second, a *decomposable 11-component reward architecture* uses fixed, pre-specified weights with per-step diagnostic logging, facilitating systematic ablation and component-wise attribution analysis. Third, a *10-action discrete interface with legal-action masking* defines explicit trading primitives (scaling, reduction, closure, and reversal) while enforcing margin-aware feasibility constraints during both training and evaluation. Three controlled experiment families are evaluated on EURUSD to analyze learning dynamics rather than generalization. Within this controlled setting, reward component interactions exhibit strongly non-monotonic effects—adding penalty terms does not consistently improve outcomes—with the full-reward configuration achieving the highest terminal training Sharpe (0.765) and cumulative return (57.09%). In the action-space comparison, the extended 10-action interface improves cumulative return while increasing turnover and reducing Sharpe relative to a conservative 3-action adapter, indicating a return–activity trade-off under a fixed training budget. In scaling experiments, all scaling-enabled variants reduce drawdown relative to no scaling, and the combined configuration achieves the strongest endpoint return.

Keywords: reinforcement learning; foreign exchange trading; reward decomposition; action masking; pyramiding; martingale scaling

1. Introduction

Reinforcement learning (RL) has emerged as a prominent paradigm for sequential decision-making in complex, uncertain environments, achieving strong performance across domains ranging from games to continuous control in robotics [17,25]. Financial trading is a particularly compelling yet inherently difficult RL application: decisions are sequential, delayed consequences are material, and market microstructure imposes strict feasibility constraints [8,18,29]. Within global financial markets, foreign exchange (Forex) is especially demanding. Unlike many equity settings, Forex trading is continuous (24/5), highly leveraged, and sensitive to financing and transaction frictions, while return distributions remain heavy-tailed and regime-dependent [5].

Despite substantial recent activity and promising results in applied RL-for-trading research—such as predictive signal extraction [6], multi-asset portfolio allocation [12], and raw return optimization

 doi:10.20944/preprints202603.1701.v1

algorithms [16,26]—three structural gaps continue to limit real-world applicability. First, *environment fidelity* is often insufficient. Real-world financial environments are heavily driven by execution mechanics that are challenging to simulate, yet crucial for performance [29]. Many studies rely on simplified simulators that under-model or omit key factors such as spread-embedded execution friction, dynamic financing costs, margin-based liquidation, and realistic decision-execution timing [4,20]. These simplifications introduce a significant sim-to-real gap, often yielding strategies that are economically brittle outside controlled simulations.

Second, *reward design opacity* obscures core learning dynamics. Conventional approaches typically optimize a single scalar reward (e.g., Profit and Loss (PnL), equity delta, or risk-adjusted metrics like proxies for the Sharpe ratio) [18,23,26]. While effective for driving learning, such monolithic formulations complicate credit assignment and make it difficult to disentangle which signals improve trade timing versus which penalties suppress excessive trading activity. Without systematic decomposition and ablation, reward shaping remains difficult to analyze, tune, and reproduce [19].

Third, *action-space granularity* is frequently misspecified. Existing formulations often rely on either coarse discrete spaces (e.g., simplistic buy/sell/hold commands) [26] or unconstrained continuous position sizing derived from deterministic policy gradients [9,12,22]. Coarse spaces fail to capture practical trading operations such as scaling into positions (pyramiding), reducing exposure, or executing explicit reversals. Conversely, continuous formulations can bypass dynamically enforceable constraints such as margin limits and position feasibility, which are often more naturally handled using action masking in structured discrete domains [11].

To address these challenges, this paper presents a unified, open-source framework designed around these three dimensions. The proposed system integrates: (i) a high-fidelity, Gymnasium-compatible Forex environment with strict anti-lookahead execution, realistic friction and rollover modeling, and margin-aware risk controls; (ii) an 11-component decomposable reward architecture with fixed, pre-specified aggregation, state-dependent clipping, and transparent per-step component logging; and (iii) a legal-action-masked, 10-action discrete space with explicit primitives for pyramiding, martingale-style scaling, reduction, closure, and reversal. All experiments are driven through configuration-based protocols, enabling controlled generation of experimental variants without ad-hoc code modifications.

Contributions. Our primary contributions are methodological and infrastructural:

- **Decomposable reward architecture:** We introduce an 11-component reward system for Forex RL, where distinct economic drivers and penalties are independently enabled, weighted, and logged, supporting systematic ablation and component-wise attribution analysis.
- **Structured action space with legality masking:** We formulate a 10-action discrete interface governed by margin-aware legality constraints, applying valid-action masking during both environment interaction and replay-based learning.
- **High-fidelity execution environment:** We design an environment with explicit anti-lookahead timing (observe close_t, execute open_{t+1}, mark close_{t+1}), comprehensive transaction cost modeling, and realistic rollover financing mechanisms, including triple-swap Wednesdays.
- **Controlled experimental protocol:** We demonstrate the framework through three experiment families (reward ablation, action-space comparison, and scaling analysis) on the EURUSD training split, with all variants generated via configuration rather than structural code changes.
- **Reproducible research infrastructure:** We provide a complete pipeline enabling deterministic reproducibility under controlled conditions through resolved configuration snapshots, fixed seeding, isolated artifact management, and leakage-free feature processing.

The empirical emphasis of this work is methodological robustness rather than state-of-the-art performance benchmarking. Reproducibility and experimental scope (including seeding and evaluation partitioning) are detailed in Section 4; interpretations focus on system behavior and learning dynamics rather than stochastic performance variation, consistent with cautionary guidance in trading-RL evaluation [29].

 doi:10.20944/preprints202603.1701.v1

Research questions. The empirical program is organized around three focused research questions:

RQ1. Which reward components contribute useful training signal, and does progressive penalty accumulation produce non-monotonic effects on training-period risk-adjusted return?

RQ2. Does increased action-space granularity improve training efficiency and risk-adjusted learning relative to a coarse three-action discrete formulation?

RQ3. How do asymmetric position-scaling strategies (pyramiding vs. martingale-style scaling) influence training-period risk profiles, drawdown dynamics, and average scaling behavior?

See Section 4 for the canonical experimental setup, including evaluation partitioning and reproducibility controls.

The remainder of the paper is organized as follows. Section 2 situates the framework within existing RL trading literature and identifies the research gap. Section 3 details the environment, reward, and agent design. Section 4 describes the evaluation protocol and reproducibility controls. Section 5 presents the empirical findings, and Section 6 discusses implications. Section 7 then outlines limitations and forward directions before Section 8 concludes.

2. Related Work

This section critically situates the manuscript in reinforcement learning (RL) for financial trading, with emphasis on where prior work succeeds and where methodological gaps remain for leveraged foreign exchange (Forex) environments. We organize the review by research theme to separate algorithmic advances from simulation assumptions and evaluation practice.

2.1. Reinforcement Learning for Financial Trading: What Has Been Solved and What Has Not

Early financial RL studies established the feasibility of direct objective optimization. Moody and Saffell [18] demonstrated recurrent RL for trading with Sharpe-oriented objectives, and Bertoluzzo and Corazza [2] compared practical RL configurations in market settings. These studies established conceptual viability but used comparatively simple execution abstractions, limiting transfer to leveraged, friction-dominated environments.

The deep-learning phase expanded representational capacity. For example, Deng et al. [6] introduced deep RL networks utilizing fuzzy logic for financial signal representation, and Jiang et al. [12] proposed portfolio-oriented deep RL in multi-asset settings using deterministic policy gradients. More recently, Theate and Ernst [26] provided an end-to-end deep RL trading workflow focusing on raw return optimization, while Liu et al. [13] improved reproducibility through their library-driven benchmark ecosystem (FinRL). However, across this line of work, execution engines frequently abstract away financing and liquidation details, making it difficult to determine whether gains arise from genuine learning quality or overly permissive simulators that ignore limit order book mechanics and friction [8,29].

To make this distinction explicit, Table 1 contrasts representative studies on market scope, reward formulation, and action-space design.

Table 1. Representative RL trading studies compared by market, reward formulation, and action-space design.

Study	Market	Reward	Actions
Moody and Saffell [18]	General	Scalar (log return / Sharpe proxy)	Continuous (portfolio weights)
Deng et al. [6]	Equities	Scalar (profit / return)	3 discrete (buy/hold/sell)
Jiang et al. [12]	Crypto	Scalar (portfolio return)	Continuous (portfolio allocation)
Carapuco et al. [4]	Forex	Scalar (profit-based)	3 discrete (buy/sell/hold)
Rundo [20]	Forex (HF)	Scalar (PnL / return)	Discrete
Theate and Ernst [26]	Equities	Scalar (net return)	3 discrete (buy/hold/sell)
Liu et al. [13]	Equities	Scalar (portfolio return / reward shaping)	Discrete (multi-asset actions)
This work	**Forex**	**Decomposable (11 components)**	**10 discrete actions**

2.2. Environment Fidelity and Temporal Causality

Simulator realism remains one of the strongest bottlenecks for credible trading RL [29]. Many implementations model spread only partially, omit rollover financing, or approximate leverage effects

 doi:10.20944/preprints202603.1701.v1

with static penalties. In leveraged Forex, these omissions are first-order, not second-order: carrying costs, maintenance margin constraints, and liquidation rules directly shape feasible policies.

Temporal causality is similarly under-specified in many studies. When observation and execution timing are not strictly separated, subtle lookahead leakage can inflate backtest quality [29]. In contrast, auditable anti-lookahead contracts explicitly bind decision, fill, and mark-to-market timestamps and therefore reduce hidden leakage pathways. This distinction is central for deployment-oriented inference.

2.3. Reward Design and Risk-Aware Objectives

Reward construction in trading RL is historically dominated by scalar objectives (raw return, equity delta, or risk-adjusted proxies) [16,26]. These objectives are easy to optimize but offer weak interpretability for diagnosing whether an agent learns profitable timing, excessive turnover, or leverage overuse.

Risk-aware formulations improve objective alignment. Sharpe-ratio optimization [18,23], Sortino-style downside sensitivity [24], and broader tail-risk considerations motivated by heavy-tailed return evidence [5] each target different risk dimensions. Yet these designs are often still collapsed into a single scalar at training time, obscuring component-level attribution.

Reward shaping theory [19] provides formal guarantees only under potential-based transformations; finance-specific penalties such as drawdown, overtrading, and margin-utilization terms are generally not policy-invariant. Consequently, component-wise ablation is methodologically necessary, not optional, when claiming that auxiliary terms improve learning.

2.4. Action-Space Modeling and Legality Constraints

Action-space design spans minimal discrete interfaces and continuous allocation policies. Minimal interfaces improve optimization stability but under-model execution operations such as scaling, partial reduction, and explicit reversal. Continuous actor-critic approaches, including proximal policy optimization (PPO) [22] and soft actor-critic (SAC) [9], improve expressive control but can make hard feasibility constraints less transparent unless constraint handling is explicitly implemented.

In constrained market simulators, legality-aware discrete control is a practical middle ground: actions remain interpretable while invalid operations can be excluded directly via masking. Prior evidence on invalid-action masking in policy-gradient settings [11] supports this direction, but adoption in trading environments remains inconsistent and often limited to interaction-time checks rather than both interaction and target computation.

2.5. Algorithmic Stability and Replay Under Financial Non-Stationarity

Value-based methods are widely used because of their implementation simplicity and compatibility with discrete control. Deep Q-Network (DQN) and Double Deep Q-Network (DDQN) methods [17,27] remain strong baselines, while dueling and distributional variants [1,28] improve representation and uncertainty modeling in heavy-tailed return settings.

Financial environments introduce additional instability: regime shifts, sparse high-quality transitions, and temporal dependence in replay. Prioritized experience replay [21] and Rainbow-style combinations [10] address part of this challenge, while sequence models such as long short-term memory (LSTM) architectures [7] capture temporal structure more explicitly. Nonetheless, these algorithmic advances do not eliminate simulator-induced bias when execution fidelity is weak.

2.6. Forex-Specific Evidence and Evaluation Standards

Forex-focused RL studies [4,14,20] show that predictive edge can be learned in currency markets, but most emphasize profitability outcomes over enforceable execution semantics and decomposable incentives. This leaves a persistent comparability problem: methods differ simultaneously in agent architecture, market assumptions, and evaluation protocol.

More generally, trading RL evaluation still suffers from heterogeneous reporting standards [29]. Modern practice requires joint reporting of return, volatility, drawdown, and turnover, plus calibration against rule-based baselines; these requirements are consistent with classical risk-return foundations in

 doi:10.20944/preprints202603.1701.v1

portfolio theory [15]. Multi-seed uncertainty quantification and stress testing remain under-reported across Forex RL literature.

2.7. Novelty and Contributions

The above review reveals a specific gap: prior studies usually improve *one* axis (algorithm, reward objective, or benchmark tooling) while under-specifying the interaction among execution realism, legality-aware control, and decomposable incentives. This manuscript targets that joint gap.

Unlike Deng et al. [6] and Theate and Ernst [26], this work enforces a strict anti-lookahead execution contract with explicit fill and mark timing in a leveraged Forex setting. In contrast to Carapuco et al. [4] and Rundo [20], this work models rollover financing, maintenance margin constraints, and forced liquidation as explicit simulator mechanics rather than implicit risk penalties. Unlike typical scalar-reward designs [16,26], this work introduces an 11-component reward architecture with deterministic per-step logging to support auditable ablation. In contrast to environments that treat invalid actions only at interaction time, this work applies legality masks consistently in both data collection and value-target computation [11].

Therefore, the contribution is not a claim of a universally superior RL algorithm; it is a systems-level methodological contribution that couples realistic execution, transparent incentives, and reproducible experimentation into a single auditable framework.

3. Methodology

This section formalizes the end-to-end research pipeline, from causal data preparation to environment transition mechanics and mask-aware value learning. To improve reproducibility, we provide explicit algorithmic specifications for (i) execution and portfolio updates, (ii) compositional reward assembly, and (iii) training-loop optimization; these correspond to Algorithms 1, 2, and 3, respectively.

Algorithm 1 Execution and portfolio update with frictions.

Input: portfolio state P_t, action proposal a_t, bars
  $(\text{close}_t, \text{open}_{t+1}, \text{close}_{t+1})$, configuration Ω, legal mask m_t
Output: next portfolio state P_{t+1}, realized/unrealized PnL,
  cost trace, violation flag, and liquidation flag
Recompute m_t from current direction, free margin, and scaling-depth limits
if $m_t[a_t] = 0$ **then**
  Set executed action $\tilde{a}_t \leftarrow$ `HOLD` and raise constraint-violation flag
else
  Set $\tilde{a}_t \leftarrow a_t$
end if
Perform pre-trade feasibility checks for $\tilde{a}_t$ (margin and leverage)
if pre-trade feasibility fails **then**
  Set $\tilde{a}_t \leftarrow$ `HOLD` and raise constraint-violation flag
end if
Set base execution anchor $p_{\text{base}} \leftarrow \text{open}_{t+1}$
Apply spread adjustment to obtain side-aware quoted price
Apply deterministic slippage to obtain final fill price p_{fill}
Execute action semantics for $\tilde{a}_t$ (open, scale, reduce, close, reverse, or hold)
Deduct commissions from cash according to traded volume
if timestamp equals rollover time (22:00 UTC) **then**
  Apply rollover financing (triple rollover on Wednesday)
end if
Mark open positions to close_{t+1} and update unrealized and realized PnL
Update cash, equity, used/free margin, utilization ratio, and drawdown statistics
if equity below liquidation threshold **or** maintenance-margin rule violated **then**
  Force-liquidate all open positions using configured liquidation convention
  Set liquidation event flag and apply liquidation accounting effects
end if
Construct and return P_{t+1} with per-step diagnostic traces

 doi:10.20944/preprints202603.1701.v1

Algorithm 2 Per-step compositional reward computation.

Input: transition trace τ_t, component enable switches $g_i \in \{0,1\}$,
  weights w_i, clipping bounds $[r_{\min}, r_{\max}]$
Output: clipped reward r_t and component log dictionary $\mathcal{L}_t$
Define fixed component order:
  profit, holding, volatility, drawdown, transaction, overtrading,
  pyramiding, martingale, margin-utilization, liquidation, constraint-violation
for each component i in the fixed order **do**
  if $g_i = 1$ **then**
    Compute component value $c_i(\tau_t)$
  else
    Set $c_i(\tau_t) \leftarrow 0$
  end if
  Compute weighted term $u_i \leftarrow w_i\, c_i(\tau_t)$
  Append (c_i, w_i, u_i, g_i) to $\mathcal{L}_t$
end for
Compute raw reward $r_t^{\text{raw}} \leftarrow \sum_i u_i$
Compute clipped reward $r_t \leftarrow \text{clip}(r_t^{\text{raw}}, r_{\min}, r_{\max})$
Append $(r_t^{\text{raw}}, r_t, \mathbb{1}[r_t^{\text{raw}} \neq r_t])$ to $\mathcal{L}_t$
Return $(r_t, \mathcal{L}_t)$

Algorithm 3 Mask-aware DQN/DDQN training with legal-action masking.

Input:
  Environment & state: environment $\mathcal{E}$; state s_t; legal-action mask $m_t \in \{0,1\}^{|\mathcal{A}|}$, where $m_t[a] = 1$ iff a is legal in s_t
  Networks: online network Q_θ; target network Q_{θ^-}
  Replay buffer: $\mathcal{D}$ with capacity N
  Training hyperparameters: batch size B; discount γ
  Scheduling: learn-start t_{start}; learn frequency f_{learn}; target-sync interval f_{target}; evaluation interval f_{eval}; total steps T; evaluation budget K_{eval}
  Algorithm type: `DQN` or `DDQN`
Initialize θ randomly; set $\theta^- \leftarrow \theta$; initialize the ϵ schedule
Reset $\mathcal{E}$ and observe (s_0, m_0)
for $t = 0$ to $T-1$ **do**
  Select a_t by mask-aware ϵ-greedy on m_t: with probability ϵ, sample $a_t \sim \text{Unif}(\{a : m_t[a] = 1\})$; otherwise $a_t \leftarrow \arg\max_{a:\, m_t[a]=1} Q_\theta(s_t, a)$
  Execute a_t in $\mathcal{E}$ and observe $(s'_t, r_t, d_t, \text{info}_t, m'_t)$
  Store $(s_t, a_t, r_t, s'_t, d_t, m_t, m'_t)$ in $\mathcal{D}$
  Update $(s_t, m_t) \leftarrow (s'_t, m'_t)$
  if $t \geq t_{\text{start}}$ **and** $t \bmod f_{\text{learn}} = 0$ **and** $|\mathcal{D}| \geq B$ **then**
    Sample minibatch $\{(s_i, a_i, r_i, s'_i, d_i, m_i, m'_i)\}_{i=1}^{B}$ from $\mathcal{D}$
    if algorithm type is `DDQN` **then**
      $a_i^\star \leftarrow \arg\max_{a:\, m'_i[a]=1} Q_\theta(s'_i, a),\ \forall i$
      $y_i \leftarrow r_i + \gamma(1 - d_i) Q_{\theta^-}(s'_i, a_i^\star),\ \forall i$
    else
      $y_i \leftarrow r_i + \gamma(1 - d_i) \max_{a:\, m'_i[a]=1} Q_{\theta^-}(s'_i, a),\ \forall i$
    end if
    $\mathcal{L} \leftarrow \frac{1}{B} \sum_{i=1}^{B} \text{Huber}\big(Q_\theta(s_i, a_i) - y_i\big)$
    Update θ by gradient descent with gradient clipping
  end if
  if $t \bmod f_{\text{target}} = 0$ **then**
    Hard-sync target network: $\theta^- \leftarrow \theta$
  end if
  if $t \bmod f_{\text{eval}} = 0$ **then**
    Evaluate the deterministic policy ($\epsilon = 0$) for K_{eval} episodes and log all outputs
  end if
  Update ϵ
end for
Output: trained parameters θ, replay buffer state, and diagnostic artifacts

 doi:10.20944/preprints202603.1701.v1

3.1. Design Principles and System Overview

The methodology is organized around three design principles: *economic fidelity*, *temporal causality*, and *auditability*. Economic fidelity is enforced through explicit transaction-cost and margin accounting; temporal causality is enforced through strict chronological processing and anti-lookahead execution timing; and auditability is enforced through deterministic component ordering, configuration snapshots, and structured logging artifacts. The resulting end-to-end pipeline (data ingestion → preprocessing → environment simulation → value-function learning → standardized evaluation export) is summarized in Figure 1.

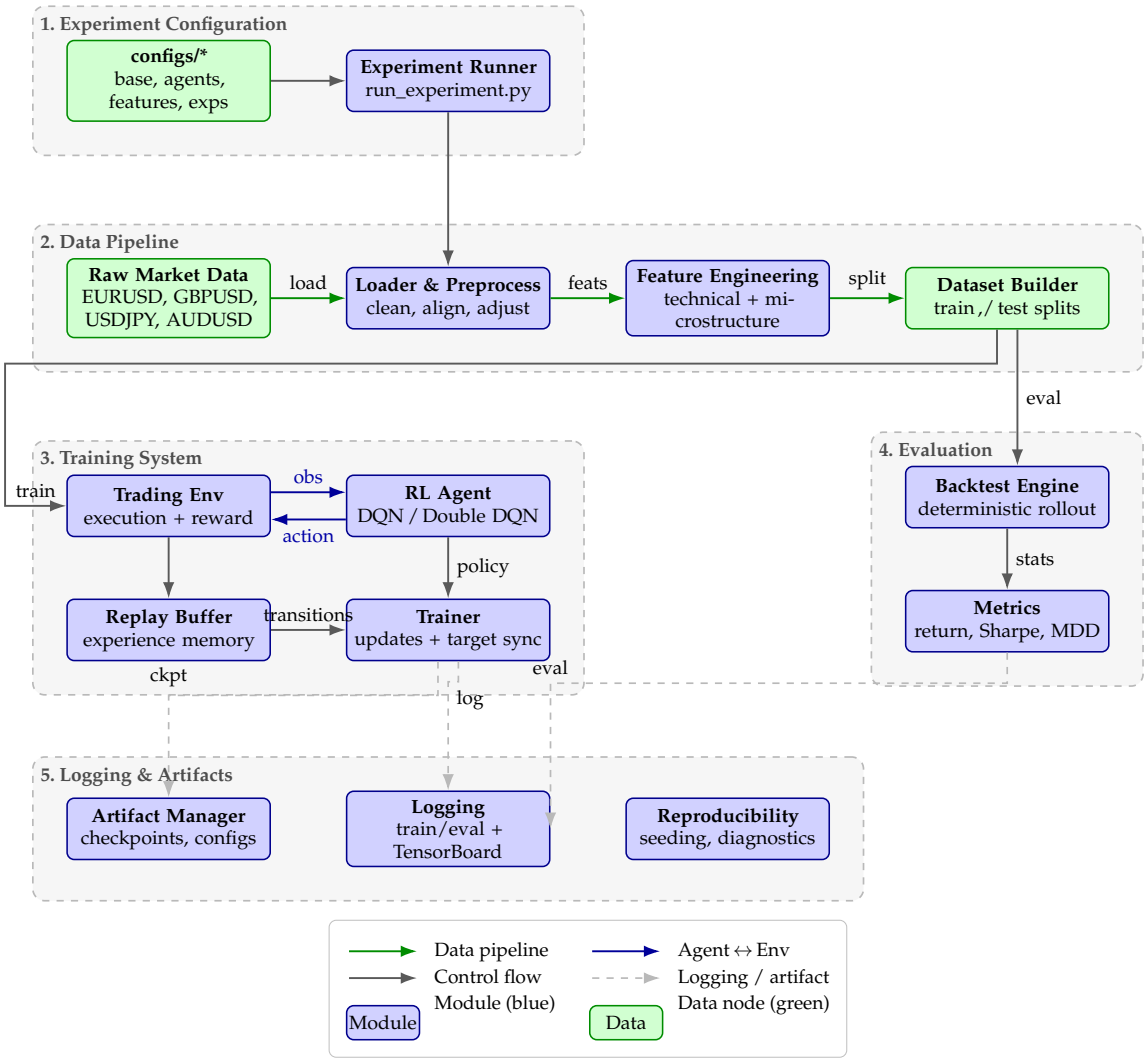

Figure 1. System architecture of the RL trading framework, linking configuration, data processing, training, deterministic backtesting, and reproducibility logging in one end-to-end pipeline.

3.2. Data Pipeline, Temporal Partitioning, and Leakage Control

The dataset consists of hourly open-high-low-close-volume (OHLCV) bars for EURUSD, GBPUSD, USDJPY, and AUDUSD (summary in Table 5). Data are normalized to Coordinated Universal Time (UTC), validated for schema integrity, deduplicated using a last-occurrence retention policy, and processed with deterministic missing-value handling.

Chronological processing policy. To preserve temporal causality, all experiments process data in strict chronological order with no shuffling. All evaluation metrics are computed exclusively on training trajectories and logged artifacts. No held-out or test split is used at any stage. This ensures that all reported results reflect only the training distribution, with no claims regarding generalization or out-of-sample performance.

 doi:10.20944/preprints202603.1701.v1

Train-only transformations and warm-up handling. Feature scaling parameters are estimated on the training data only and reused throughout. To initialize rolling indicators at the start of training, a warm-up lookback segment is prepended from the beginning of the training data, and warm-up rows are discarded from final training artifacts. No test or held-out data is used for any transformation or initialization.

3.3. Feature Engineering and Observation Construction

Features are organized into deterministic namespaces so that all runs share a stable, auditable column ordering:

- **Technical indicators**: simple and exponential moving averages (SMA/EMA; 10, 20, 50), relative strength index (RSI; 14), moving average convergence divergence (MACD; 12, 26, 9), Bollinger components (20, 2σ), one-bar log return, and rolling volatility.
- **Microstructure proxies**: spread proxy $(H - L)/C$, short-horizon price-change proxy, realized volatility proxy, and UTC session label.

The canonical ordering improves reproducibility and prevents accidental feature-index drift across experiment variants.

Windowing and input dimensions. Let L denote the observation lookback window ($L = 24$ in release runs), d_{feat} the number of engineered market features, $d_{\text{port}} = 10$ the portfolio-state dimension, and n_a the active action count ($n_a \in \{3, 10\}$). The flattened MLP input dimension is

$$d_{\text{flat}} = L\, d_{\text{feat}} + d_{\text{port}} + n_a. \tag{1}$$

Equation 1 is used consistently across training and evaluation to guarantee that architecture instantiation matches the declared observation schema. The observation schema is summarized in Table 2.

Table 2. Observation schema and dimensions used by the Gymnasium environment.

Field	Shape	Role
`market`	$[L, d_{\text{feat}}]$	Rolling market-feature window ($L = 24$)
`portfolio`	$[d_{\text{port}}]$	Normalized account state ($d_{\text{port}} = 10$)
`mask`	$[n_a]$	Legal-action mask ($n_a \in \{3, 10\}$)
`flat`	$[Ld_{\text{feat}} + d_{\text{port}} + n_a]$	Flattened vector for MLP agents

3.4. Trading Environment

The environment follows the Gymnasium application programming interface (API), consistent with the OpenAI Gym design lineage [3], and exposes dictionary observations together with deterministic transition diagnostics. This explicit state contract enables direct reproducibility checks across training and evaluation workflows.

3.4.1. Action Space and Legal Masking

The extended action interface contains 10 operations:

- `HOLD`
- `OPEN_LONG`, `OPEN_SHORT`
- `PYRAMID_LONG`, `PYRAMID_SHORT`
- `MARTINGALE_LONG`, `MARTINGALE_SHORT`
- `REDUCE`
- `CLOSE`
- `REVERSE`

For controlled comparisons, a simplified 3-action adapter is used:

- `HOLD`
- `TARGET_LONG`
- `TARGET_SHORT`

 doi:10.20944/preprints202603.1701.v1

These are mapped to valid extended operations conditioned on the current position state.

Action legality is recomputed at every step using direction, margin availability, and scaling-depth constraints. The legal mask is stored with transitions in replay and reused for both exploratory sampling and masked argmax target computation. Although prior empirical evidence is reported primarily for policy-gradient settings [11], the same legality principle is applied here to value-based updates. Table 3 summarizes extended-action semantics.

Table 3. Extended action semantics and legality conditions (executed at $open_{t+1}$).

ID	Action	Primary legality precondition	Execution effect
0	HOLD	Always legal	Maintain current position; no new order submitted
1	OPEN_LONG	Flat position and sufficient free margin	Open long exposure at base lot size
2	OPEN_SHORT	Flat position and sufficient free margin	Open short exposure at base lot size
3	PYRAMID_LONG	Existing long position, scaling depth below cap, margin available	Add to long position using configured pyramid increment
4	PYRAMID_SHORT	Existing short position, scaling depth below cap, margin available	Add to short position using configured pyramid increment
5	MARTINGALE_LONG	Existing long position, martingale depth below cap, margin available	Add long exposure after adverse movement using martingale scaling rule
6	MARTINGALE_SHORT	Existing short position, martingale depth below cap, margin available	Add short exposure after adverse movement using martingale scaling rule
7	REDUCE	Non-flat position	Partially reduce current exposure (configured reduction fraction)
8	CLOSE	Non-flat position	Fully close current position
9	REVERSE	Non-flat position and sufficient margin for opposite side	Close current direction, then open opposite direction within one action

3.4.2. Execution Timing and Market Frictions

To prevent lookahead leakage, step timing is fixed as:

$$\begin{aligned} \text{observe close}_t \;\rightarrow\; \text{decide } a_t \;\rightarrow\; \text{fill open}_{t+1} \\ \rightarrow\; \text{mark close}_{t+1}. \end{aligned} \tag{2}$$

The fill model applies spread, commission, and slippage in deterministic order; rollover financing is charged at 22:00 UTC with triple rollover on Wednesdays. Equation 2 is the causal timing contract summarized in Figure 2.

 doi:10.20944/preprints202603.1701.v1

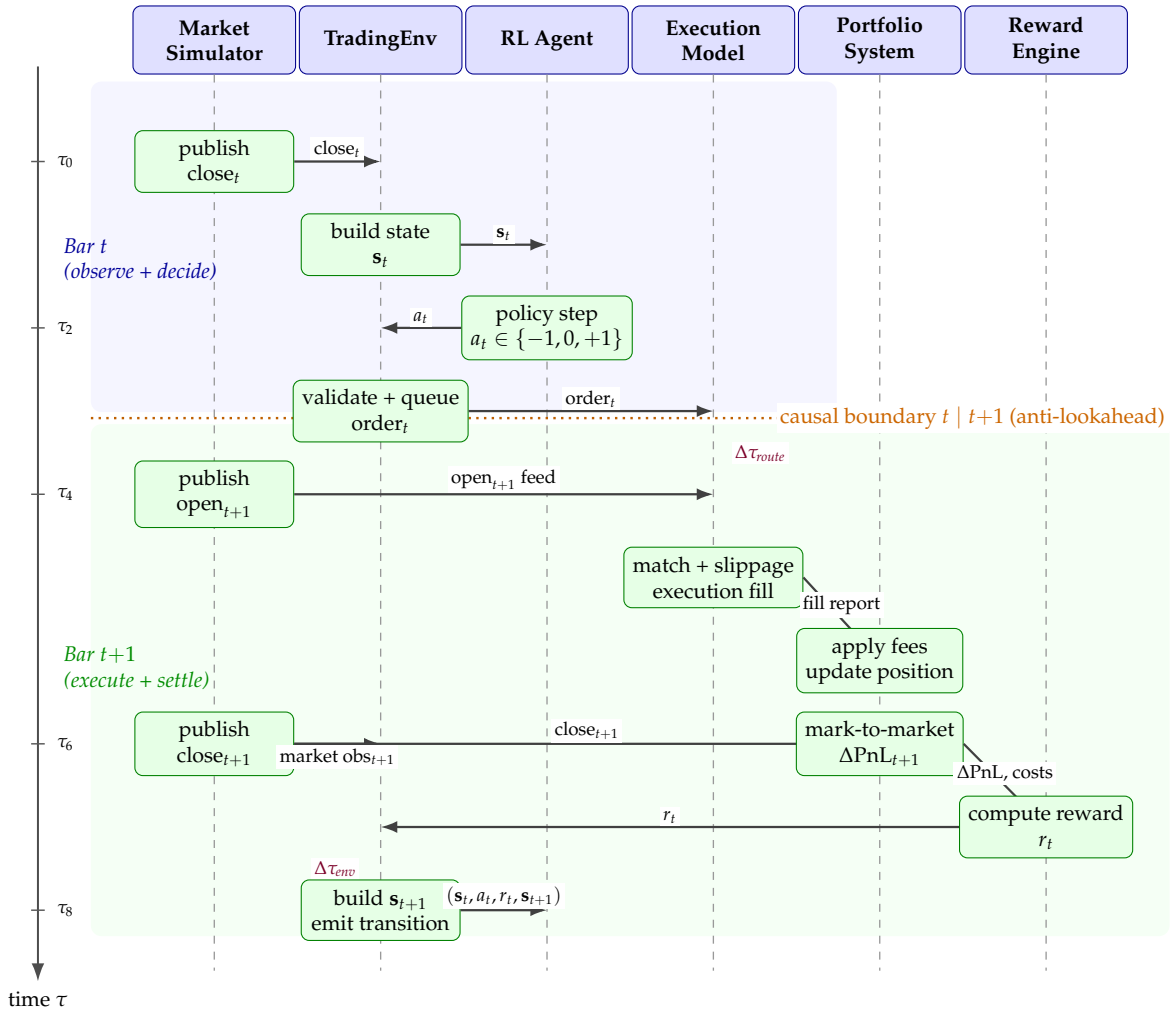

Figure 2. Causality-correct environment step timing: the environment emits $\mathbf{s}_t$, receives a_t, execution is driven by market open_{t+1}, portfolio state is marked at close_{t+1}, reward engine returns only r_t to the environment, and the environment emits the full transition $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ without any Reward→Agent bypass.

3.4.3. Portfolio and Risk Accounting

The account model tracks cash, equity, realized and unrealized PnL, used/free margin, directional exposure, scaling depth, and drawdown. Core constraints include maximum leverage (30×), maintenance margin (50% of initial margin requirement), bounded scaling depth, and forced liquidation at a fixed equity threshold. These controls are necessary to prevent value-function optimization from exploiting infeasible leverage trajectories.

Algorithm 1 formalizes the exact execution and accounting order, including legality checks, friction application, rollover timing, margin updates, and forced liquidation handling.

3.5. Compositional Reward Modeling

We model the reward as a weighted composition of components. Let $c_i(s_t, a_t, s_{t+1})$ denote component i with weight w_i. The pre-normalized reward is:

$$R_t^{\text{raw}} = \sum_{i=1}^{11} w_i \, c_i(s_t, a_t, s_{t+1}). \tag{3}$$

The final reward applies clipping:

$$R_t = \text{clip}(R_t^{\text{raw}}, -1, 1). \tag{4}$$

Equation 3 defines deterministic component aggregation, and Equation 4 bounds temporal-difference target magnitude.

 doi:10.20944/preprints202603.1701.v1

Clipping vs. running normalization. Clipping bounds temporal-difference targets and stabilizes early training by limiting extreme updates. It preserves sign information and relative ordering within the unclipped region, but compresses large-magnitude rewards. In contrast, running normalization mitigates scale drift but introduces non-stationarity into the learning objective. We therefore adopt clipping as the default and use adaptive normalization only in sensitivity analyses.

Component structure and weighting. The 11 components span four functional groups: (i) return (post-cost profit), (ii) risk control (volatility and drawdown), (iii) trading frictions (transaction cost and overtrading), and (iv) constraints (margin usage, liquidation, and rule violations), with asymmetric penalties for scaling behaviors (martingale penalized more strongly than pyramiding). Baseline weights and definitions are provided in Table 4.

Table 4. Reward component definitions, baseline weights, and intended behavioral effect.

Component	Weight	Primary purpose	Typical trigger / penalty form
Profit (post-cost)	1.00	Core economic learning signal	Step-wise post-cost equity return
Holding incentive	0.03	Encourage controlled profitable holding	Small bonus when position remains profitable under low drawdown
Volatility penalty	0.01	Discourage unstable equity trajectories	Negative term proportional to short-horizon equity-return volatility
Drawdown penalty	0.05	Penalize increasing downside risk	Incremental drawdown penalty with stronger regime beyond severe drawdown threshold
Transaction burden	0.10	Reduce excessive cost-churning behavior	Cost-normalized penalty for spread, slippage, commission, and rollover
Overtrading penalty	0.02	Suppress unnecessary action frequency	Bounded penalty tied to elevated local trade frequency
Pyramiding penalty	0.05	Constrain winner-adding aggressiveness	Depth-aware penalty for additional pyramid levels
Martingale penalty	0.12	Strongly discourage loss-amplifying averaging down	Depth-aware penalty with larger coefficient than pyramiding
Margin-utilization penalty	0.05	Avoid extreme leverage usage before liquidation	Nonlinear penalty when utilization exceeds conservative threshold
Liquidation penalty	2.00	Impose catastrophic-event aversion	Large penalty on forced liquidation event
Constraint-violation penalty	0.10	Deter illegal action proposals	Penalty when policy attempts an invalid action

Algorithm 2 specifies the deterministic per-step reward assembly pipeline, including component enable switches for ablations, weighting, clipping, and structured logging.

Logging and ablation protocol. Component evaluation follows a fixed order, and each term is logged per step via the environment `info` dictionary and persisted reward traces. This enables direct attribution in ablation experiments (`r1`–`r7`), where individual components or weights are selectively modified. Because non-potential reward shaping can alter optimal policies [19], profit remains the primary signal, with auxiliary penalties introduced incrementally.

 doi:10.20944/preprints202603.1701.v1

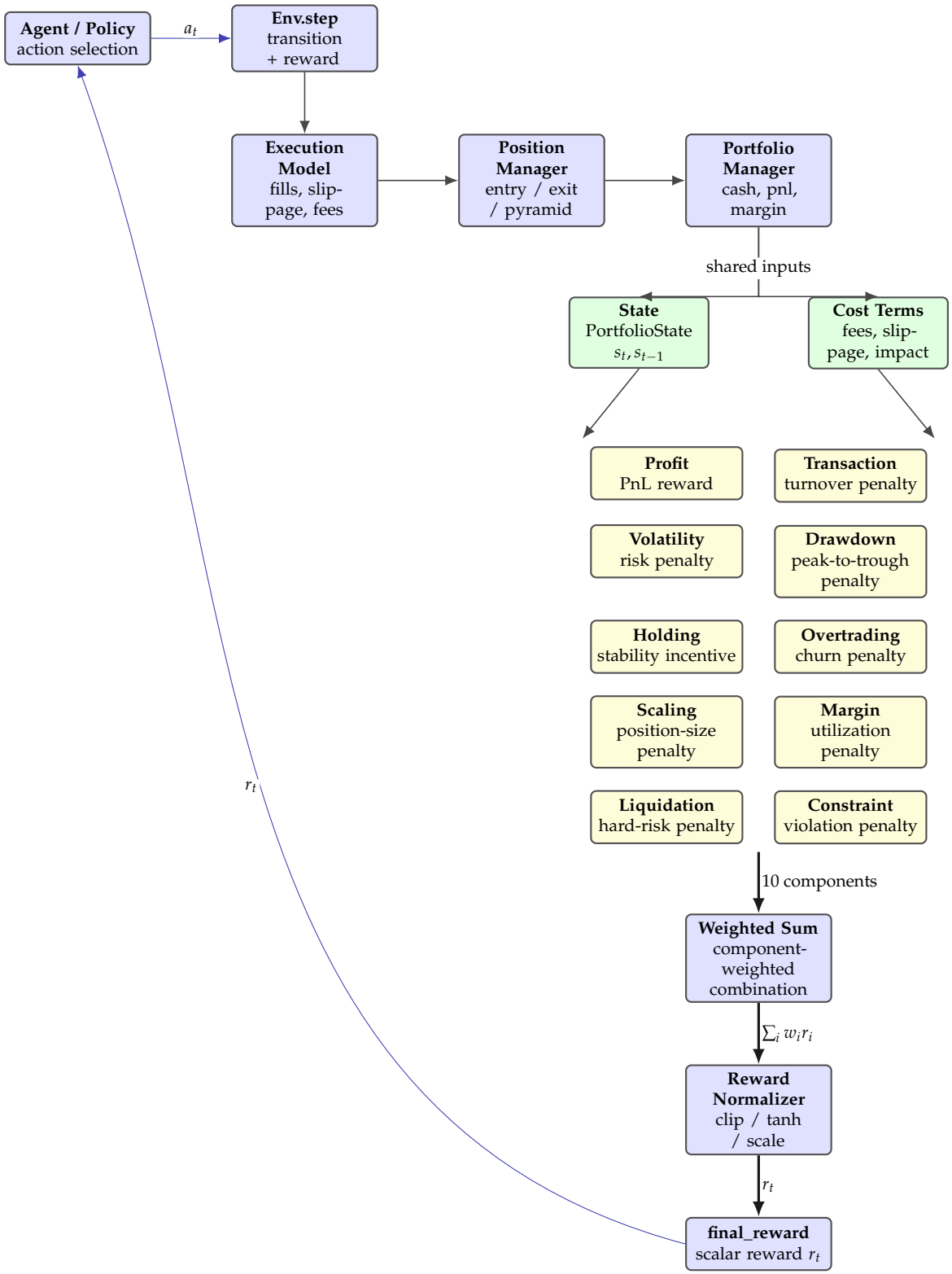

Figure 3. Reward taxonomy and computation flow for the RL trading environment, from agent–environment interaction through component aggregation and normalization to the final scalar reward.

3.6. Agent Architectures

Both agents are value-based and share the same backbone: a feed-forward multilayer perceptron (MLP) with hidden dimensions $[512, 512, 256]$, ReLU activations, and a linear Q-value output head. Output dimensionality equals the active action-space size (3 in simplified mode, 10 in extended mode). The agents are Deep Q-Network (DQN) and Double Deep Q-Network (DDQN) variants, selected because they support explicit legality masking in discrete control.

 doi:10.20944/preprints202603.1701.v1

3.6.1. DQN

DQN uses temporal-difference targets with periodic hard target-network synchronization [17]:

$$y_t^{\mathrm{DQN}} = r_t + \gamma(1 - d_t) \max_{a' \in \mathcal{A}_{\mathrm{legal}}(s_{t+1})} Q_{\theta^-}(s_{t+1}, a'). \tag{5}$$

Equation 5 masks invalid actions directly in target construction.

3.6.2. Double DQN

Double DQN decouples action selection and evaluation:

$$a^* = \arg \max_{a' \in \mathcal{A}_{\mathrm{legal}}(s_{t+1})} Q_\theta(s_{t+1}, a'). \tag{6}$$

$$y_t^{\mathrm{DDQN}} = r_t + \gamma(1 - d_t) Q_{\theta^-}(s_{t+1}, a^*). \tag{7}$$

Equations 6 and 7 separate online action selection from target-network evaluation, reducing value overestimation in volatile training regimes [27].

Mask-aware exploration policy. Exploration follows an ϵ-greedy policy defined on legal actions only: with probability ϵ, the agent samples uniformly from legal actions; otherwise, it selects the legal argmax action. The same legality constraints are applied during replay-based target computation, ensuring consistency between data collection and value updates.

Algorithmic variants outside release scope. The architecture is compatible with dueling heads and distributional value estimation, which are relevant in heavy-tailed return settings [1,28]. These variants are intentionally excluded from the main release protocol to isolate the effect of environment and reward design under DQN-family baselines.

3.7. Training Protocol

Core settings are fixed across experiment families unless explicitly overridden. For global reproducibility and side-by-side comparability, the exact training and environment values are consolidated in Table 6 within Section 4.

Algorithm 3 summarizes the full mask-aware DQN/DDQN training loop, including replay storage of legality masks, learn/start frequency gates, target-network synchronization, and periodic evaluation cadence.

Rationale for optimization settings. The 10^6-step horizon improves coverage of diverse market regimes for off-policy replay. A 40K replay buffer balances decorrelation and recency under non-stationary financial dynamics. Batch size 128 and Huber loss provide stable gradient estimates, while gradient clipping (norm 10) limits rare but destabilizing updates. The learning rate (2.5×10^{-4}) and target-update interval (2,000 steps) were selected to control estimator variance without slowing convergence excessively.

Evaluation cadence and diagnostic logging. Training logs include learning curves, loss, reward components, action distributions, turnover, and liquidation events. Periodic deterministic evaluation is performed during training (default interval: every 10,000 steps) using only training episodes and stored trajectories. At the end of each run, a deterministic full-horizon evaluation is conducted on the training data. Reported metrics include cumulative and annualized returns, volatility, Sharpe [23], Sortino [24], maximum drawdown, win rate, turnover, liquidation statistics, and scaling utilization—all computed strictly from training artifacts. No held-out or test data is used at any point.

Determinism scope and methodological limitation. Reproducibility and seeding details are centralized in the experimental-design protocol (see Section 4). The manuscript reports deterministic, artifact-level runs; see Section 4 for the canonical seed value and evaluation scope.

 doi:10.20944/preprints202603.1701.v1

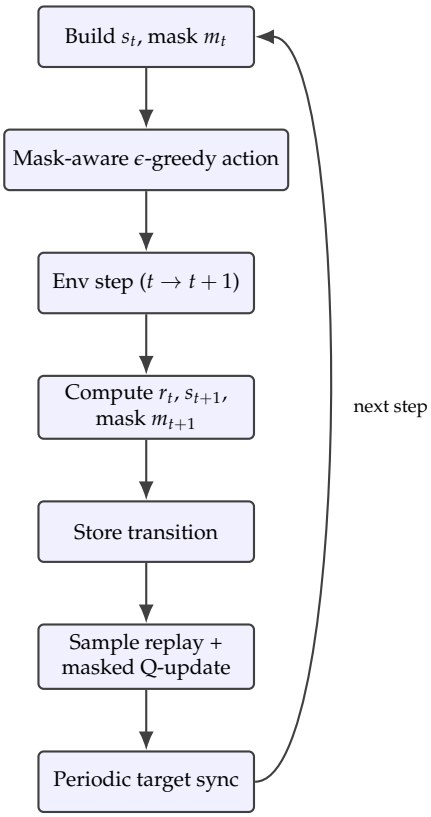

Figure 4. Training-loop schematic with mask-aware interaction, complete transition construction, replay learning, and periodic target synchronization.

4. Experimental Design

This section specifies the controlled evaluation protocol used to answer **RQ1**–**RQ3**. We first define scope and invariants, then map each experiment family to a research question, and finally describe metrics, statistical procedures, and reproducibility controls.

4.1. Dataset Scope and Split Policy

The primary controlled families (EXP-01–EXP-03) are executed on hourly EURUSD training-split data with Double DQN [27] to isolate component effects under a fixed optimization budget. We additionally report contextual robustness analyses (benchmark and cross-pair comparisons) from the same artifact generation pipeline to calibrate the practical relevance of the primary findings.

Table 5. Dataset statistics (raw hourly bars; chronological 80/20 split shown for completeness; all reported experiments use the training split only).

Pair	N_{total}	N_{train}	$N_{\text{held-out}}$	Missing(%)
AUDUSD	25000	20000	5000	–
EURUSD	25000	20000	5000	–
GBPUSD	25000	20000	5000	–
USDJPY	25000	20000	5000	–

Pair	Train period	Held-out period (unused here)
AUDUSD	2022-01-01 – 2025-01-01	2025-01-01 – 2026-01-01
EURUSD	2022-01-01 – 2025-01-01	2025-01-01 – 2026-01-01
GBPUSD	2022-01-01 – 2025-01-01	2025-01-01 – 2026-01-01
USDJPY	2022-01-01 – 2025-01-01	2025-01-01 – 2026-01-01

The dataset table clarifies that all pairs share an aligned chronological split policy, allowing pair-level comparisons without confounding differences in temporal coverage.

 doi:10.20944/preprints202603.1701.v1

4.2. Experiment Families and Contextual Extensions

All studies are executed via configuration-only overrides using YAML (YAML Ain't Markup Language) files, with no structural code changes:

- **EXP-01 Reward ablation**: cumulative reward component schedule `r1` to `r7`.
- **EXP-02 Action space**: simplified (3-action adapter) vs extended (10-action).
- **EXP-03 Scaling analysis**: no scaling, pyramid-only, martingale-only, both.
- **Context A (benchmark calibration)**: RL agents compared against random, buy-and-hold, momentum, and mean-reversion baselines under identical simulator semantics.
- **Context B (cross-pair transfer diagnostics)**: DQN vs. Double DQN comparisons on GBPUSD, USDJPY, and AUDUSD using the full reward profile.

All families inherit the same anti-lookahead timing rule, transaction-friction model, and margin constraints, ensuring that observed differences arise from controlled design changes rather than simulator drift.

4.3. Protocol Invariants and Hyperparameter Budget

To preserve comparability, core optimization and environment settings are held constant across families unless a specific family intentionally perturbs that axis. These shared values are summarized in Table 6. Fixing this budget is especially important when interpreting action-space and scaling effects, because it prevents confounding from unequal replay exposure or optimizer aggressiveness.

Table 6. Core hyperparameters, interface dimensions, and environment settings used in release experiments.

Parameter	Value	Scope
Observation schema	Dict: market/portfolio/mask/flat	Environment
Observation window	24 bars	Environment
Portfolio vector dimension	10	Environment
Action outputs (n_a)	10 (extended) or 3 (simplified)	Agent/Environment
Flat input dimension	$24\,d_{\text{feat}} + 10 + n_a$	Agent
Q-network architecture	MLP [512, 512, 256] + ReLU + linear head	Agent
Total timesteps	1,000,000	Training
Replay buffer size	40,000	Training
Batch size	128	Training
Learn start	10,000 steps	Training
Learn frequency	every 4 steps	Training
γ	0.99	Training
Optimizer	Adam (2.5e-4)	Agent
Epsilon schedule	1.0→0.01 (30K)	Agent
Target update interval	2,000 steps	Agent
Gradient clip	10.0	Agent
Periodic diagnostic evaluation	every 10,000 steps (default)	Training
Automatic mixed precision (AMP) dtype	fp16	Agent
Initial capital	USD 100,000	Environment
Max leverage	30x	Environment
Liquidation threshold	25% equity	Environment
Commission	USD 3.5 / lot (round trip)	Environment
Base slippage	0.5 pips	Environment
Seed	See Section 4	Reproducibility
Temporal split	80/20 chronological	Reproducibility

4.4. Research Questions

Each experiment family is explicitly tied to a research question that guides design choices and governs interpretation of outcomes:

 doi:10.20944/preprints202603.1701.v1

RQ1 (Reward Interactions):
Do reward components interact non-monotonically, and does progressive penalty activation improve or degrade training-period risk-adjusted return compared to a bare profit signal? (EXP-01)

RQ2 (Action Granularity):
How does the extended 10-action interface change return, risk-adjusted performance, drawdown, and turnover relative to the 3-action coarse adapter? (EXP-02)

RQ3 (Scaling Asymmetry):
Is the 2.4× martingale-to-pyramid penalty ratio sufficient to produce meaningfully asymmetric position-management behavior, with pyramid-only variants outperforming martingale-only on drawdown? (EXP-03)

Context A and Context B are interpreted as calibration analyses rather than additional primary RQs; they are used to situate the main EURUSD findings against stronger baselines and pair-level variation.

4.5. Metrics and Statistical Testing

Evaluation captures profitability, risk, and behavior: cumulative return, annualized return/volatility, Sharpe [23], Sortino [24], maximum drawdown, win rate, turnover, liquidation metrics, and scaling utilization. For paired model comparisons, paired outcomes are reported descriptively; no inferential hypothesis testing is performed.

Because all reported values are endpoint artifacts, we emphasize effect patterns and relative trade-offs over formal claims of distributional superiority; see Section 4 for canonical seeding and evaluation scope, consistent with current cautionary guidance in trading-RL evaluation practice [29].

4.6. Reproducibility Protocol

Reproducibility controls include deterministic seeding across Python/NumPy/PyTorch/environment, resolved-config snapshots stored per run, canonical artifact paths, and strict chronological splitting. Numerical findings in this manuscript are therefore directly traceable to saved outputs. This protocol is aligned with reproducibility priorities emphasized in prior trading-RL benchmark literature [13,29].
Important scope note. This release reports *single-seed* outcomes (seed 42). Accordingly, we interpret differences as informative system-behavior evidence rather than definitive estimates of expected out-of-sample superiority.

5. Results

All values in this section are derived from *training artifacts only* (see Section 4). We report primary outcomes for EXP-01–EXP-03, then place them in context using benchmark and cross-pair diagnostics generated under the same simulator semantics.

5.1. Global Performance Dashboard

Before unpacking each experiment family, Figure 5 provides a compact multi-metric snapshot of profitability, risk, and behavior indicators used throughout this section.

The dashboard confirms that no single configuration dominates all dimensions simultaneously: variants with stronger endpoint return often incur higher activity or weaker conservative risk summaries, reinforcing the need for objective-specific model selection.

 doi:10.20944/preprints202603.1701.v1

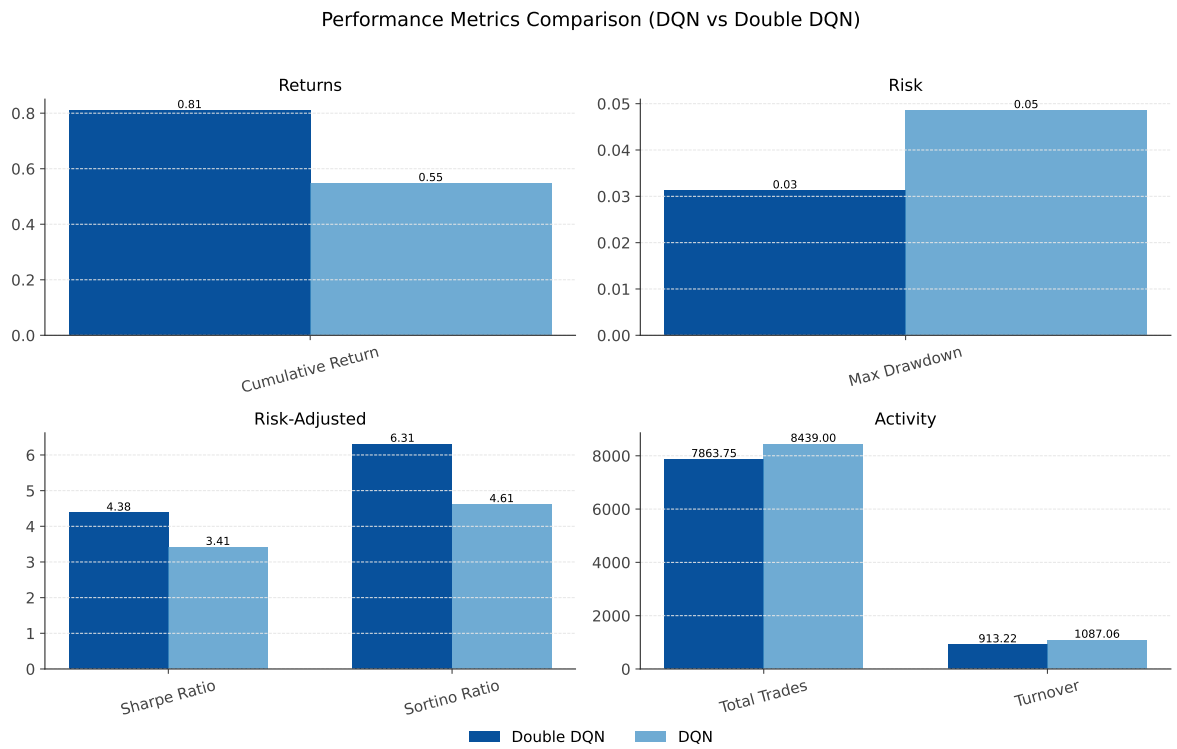

Figure 5. Multi-panel dashboard of endpoint metrics across retained experiment families, summarizing return, drawdown, risk-adjusted performance, and activity.

5.2. Reward Component Ablation (EXP-01)

Objective. EXP-01 progressively enables reward components to identify constructive versus destructive interactions.

We first report endpoint statistics in Table 7, then use distributional and trajectory views to diagnose interaction nonlinearity.

Table 7. Reward ablation on EURUSD train split (Double DQN; see Section 4).

Variant	Sharpe	Sortino	Cum.Ret(%)	MaxDD(%)	WinRate(%)	Turnover	AvgPyr	AvgMart
`r1`	0.412	0.573	27.79	10.48	35.18	1592.83	0.113	0.121
`r2`	0.237	0.353	12.20	8.60	31.85	1741.58	0.187	0.127
`r3`	0.638	0.979	41.20	5.44	31.53	1488.67	0.165	0.138
`r4`	0.687	1.029	43.21	5.68	30.99	1344.47	0.197	0.144
`r5`	0.589	0.822	41.74	4.12	31.07	1350.00	0.172	0.156
`r6`	0.410	0.539	24.58	7.70	27.38	1200.32	0.132	0.089
`r7`	0.765	1.117	57.09	2.31	33.15	1156.51	0.173	0.169

Endpoint values already indicate interaction effects: Sharpe and cumulative return do not improve monotonically with added penalties, and several intermediate variants underperform simpler compositions.

Figure 6 makes the non-monotonicity explicit: the sequence deteriorates from `r1` to `r2`, recovers across `r3`–`r5`, softens at `r6`, and peaks at `r7`.

The trajectory view in Figure 7 corroborates the endpoint summary: the strongest terminal profile is not produced by the earliest penalty additions, but by the fully calibrated composition. This supports the claim that reward engineering is an empirical calibration task rather than a strictly additive design process.

 doi:10.20944/preprints202603.1701.v1

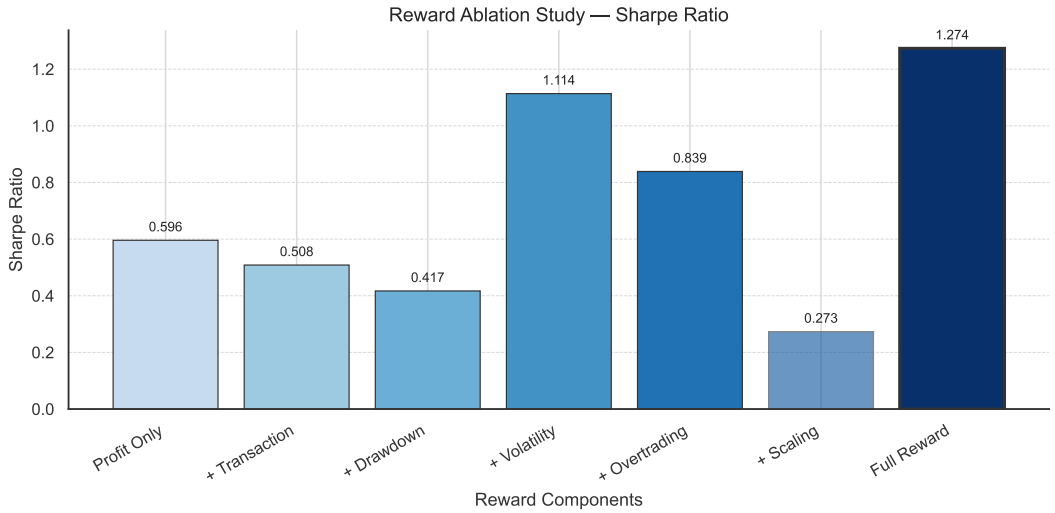

Figure 6. Reward-ablation comparison for variants `r1`–`r7` on EURUSD, showing a non-monotone profile with `r7` as the strongest endpoint.

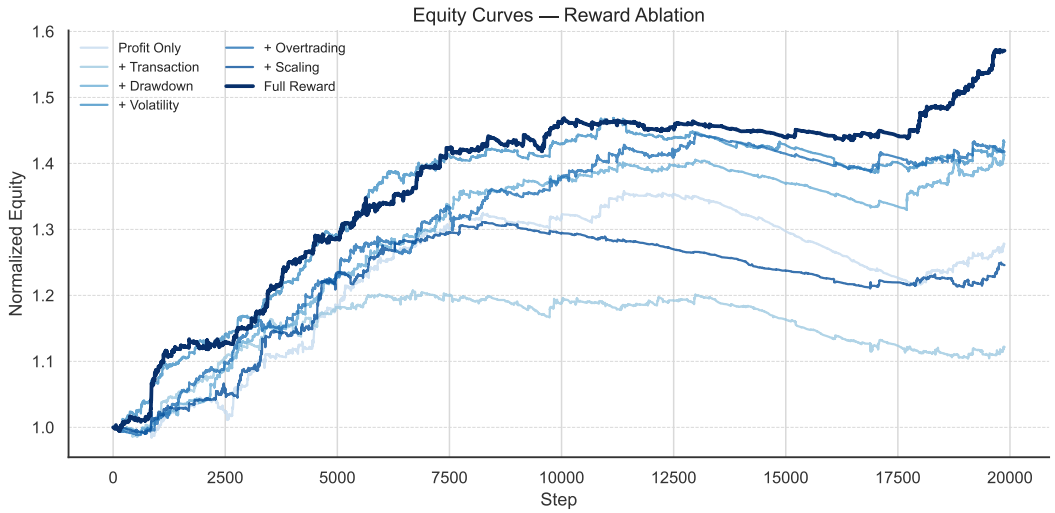

Figure 7. Training equity trajectories for `r1`–`r7`, highlighting non-monotone intermediate behavior and the strongest endpoint for `r7`.

5.3. Action-Space Comparison (EXP-02)

Objective. EXP-02 compares the extended 10-action interface with the simplified 3-action adapter under matched training settings.

As shown in Table 8, the extended interface improves cumulative return while increasing turnover and reducing Sharpe relative to the conservative adapter, indicating a return–activity trade-off under a fixed budget.

Table 8. Action-space comparison on EURUSD (Double DQN).

Action Space	Sharpe	Cum.Ret(%)	MaxDD(%)	WinRate(%)	Turnover
Extended (10)	0.765	57.09	2.31	33.15	1156.51
Simplified (3)	2.433	33.21	0.29	68.13	528.65

To visualize how this trade-off accumulates over the episode rather than only at endpoint, Figure 8 contrasts the corresponding equity paths.

The temporal separation in Figure 8 clarifies that action granularity changes policy behavior throughout training, not only final-point metrics: richer primitives increase opportunity capture but amplify path variability.

 doi:10.20944/preprints202603.1701.v1

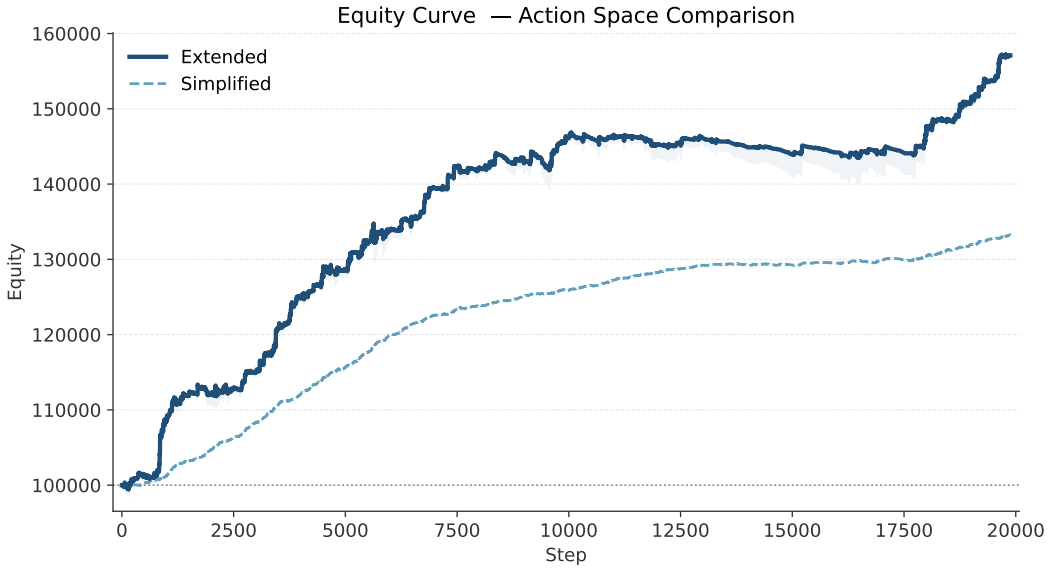

Figure 8. Equity-curve comparison of simplified (3-action) and extended (10-action) interfaces on EURUSD, showing a return–smoothness trade-off.

5.4. Scaling Strategy Analysis (EXP-03)

Objective. EXP-03 evaluates four scaling configurations: no scaling (`s1`), pyramid-only (`s2`), martingale-only (`s3`), and combined (`s4`).

Table 9 shows that all scaling-enabled variants improve over no scaling, with distinct objective-dependent preferences across return and drawdown.

Table 9. Scaling-strategy analysis on EURUSD (Double DQN).

Variant	Sharpe	Cum.Ret(%)	MaxDD(%)	AvgPyr	AvgMart	Turnover
s1_no_scaling	-1.568	-16.54	16.74	0.000	0.000	1440.09
s2_pyramid	-0.136	-2.48	5.73	0.083	0.000	1394.52
s3_martingale	0.319	8.94	4.63	0.000	0.122	1273.71
s4_both	0.355	13.08	6.79	0.086	0.096	1483.55

The path-level evidence in Figure 9 confirms that the gap is structural rather than noise at a single endpoint: enabling scaling alters drawdown dynamics over the full training horizon.

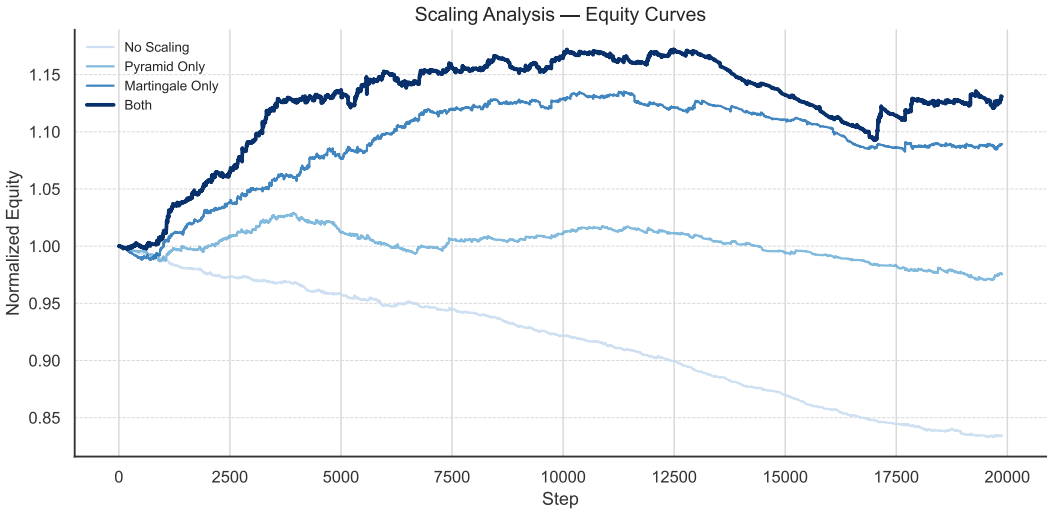

Figure 9. Training equity trajectories for scaling variants `s1`–`s4` on EURUSD, where scaling-enabled variants reduce maximum drawdown versus no scaling.

 doi:10.20944/preprints202603.1701.v1

5.5. Benchmark Calibration on EURUSD

Primary RQ findings are further calibrated against non-RL baselines and a DQN comparator. We first report headline endpoint metrics in Table 10, then inspect complementary risk diagnostics in Table 11.

Table 10. Benchmark comparison on EURUSD. RL agents use full reward variant `r7`.

Method	Sharpe	Cum.Ret(%)	MaxDD(%)	WinRate(%)	Turnover
Random Policy	-13.380	-34.58	34.70	25.99	2199.98
Buy-and-Hold	-0.358	-1.05	2.08	0.00	0.11
Momentum	-1.355	-3.82	4.05	23.62	186.28
Mean-Reversion	-0.807	-1.67	2.34	24.64	157.67
DQN	0.369	24.56	8.72	34.12	1316.51
Double DQN	0.765	57.09	2.31	33.15	1156.51

Table 11. Benchmark risk diagnostics on EURUSD (see Section 4).

Method	Ann.Ret(%)	Ann.Vol(%)	Sortino	Trades
Random Policy	-12.11	0.96	-16.523	13728
Buy-and-Hold	-0.32	0.89	-0.466	1
Momentum	-1.18	0.87	-1.789	1715
Mean-Reversion	-0.51	0.63	-0.772	1469
DQN	6.36	3.37	2.643	11240
Double DQN	14.82	4.11	4.771	8415

Taken together, the two tables show that RL improvements are not merely profit inflation: Double DQN remains favorable under risk-aware summaries and avoids liquidation events in this artifact view, while rule-based baselines underperform on cumulative return.

Figure 10 reinforces that the learning-based agents maintain stronger growth profiles than benchmark heuristics under identical execution assumptions.

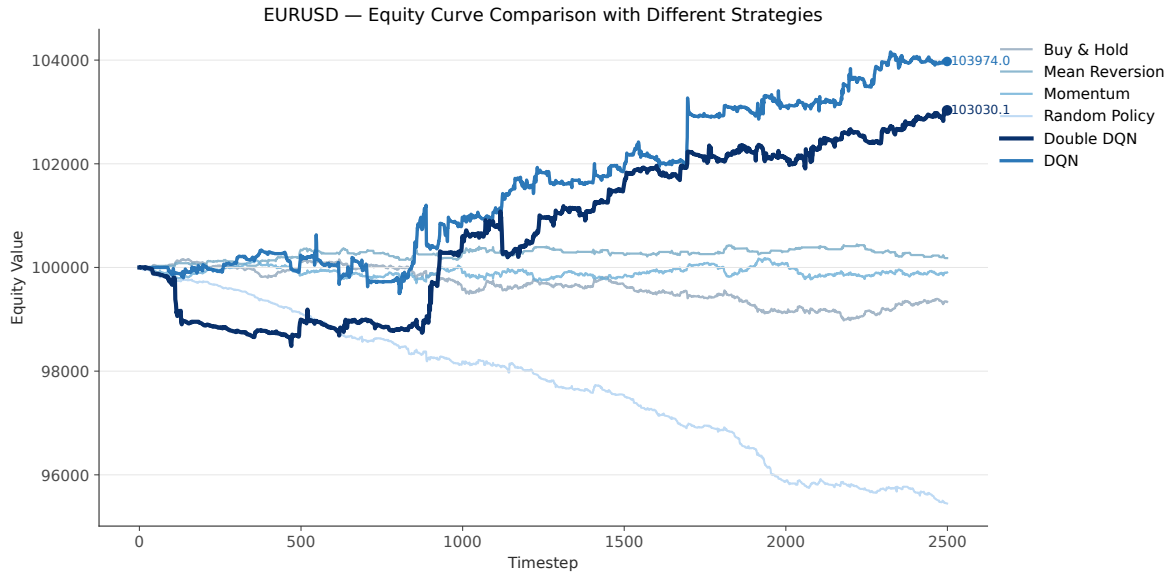

Figure 10. Equity trajectories for RL agents and benchmark strategies on EURUSD, with rule-based baselines as calibration anchors.

5.6. Cross-Pair Generality Diagnostics

To assess whether the EURUSD pattern is idiosyncratic, Table 12 reports DQN-vs-DDQN outcomes on three additional pairs using the same full-reward configuration.

 doi:10.20944/preprints202603.1701.v1

Table 12. Cross-pair DQN vs. Double DQN on train split (full reward variant `r7`).

Pair	Agent	Sharpe	Sortino	Cum.Ret(%)	MaxDD(%)	WinRate(%)
EURUSD	DQN	0.369	0.501	24.53	8.72	34.15
EURUSD	DDQN	0.765	1.117	57.09	2.31	33.15
GBPUSD	DQN	0.164	0.195	11.98	9.59	31.96
GBPUSD	DDQN	0.759	0.969	83.19	3.14	31.90
USDJPY	DQN	0.958	1.419	94.08	2.16	35.63
USDJPY	DDQN	1.115	1.615	115.01	4.44	33.69
AUDUSD	DQN	0.542	0.754	33.93	7.63	30.91
AUDUSD	DDQN	0.811	1.222	52.98	4.91	32.16

These results are restricted to the experimental scope defined in Section 4; the directional trend across pairs is descriptive: Double DQN is generally stronger on return-oriented metrics, with pair-specific differences in drawdown behavior.

Figure 11 illustrates that cross-pair behavior is heterogeneous in amplitude but consistent in qualitative ordering, supporting the use of pair-diverse diagnostics alongside the primary EURUSD-focused research questions.

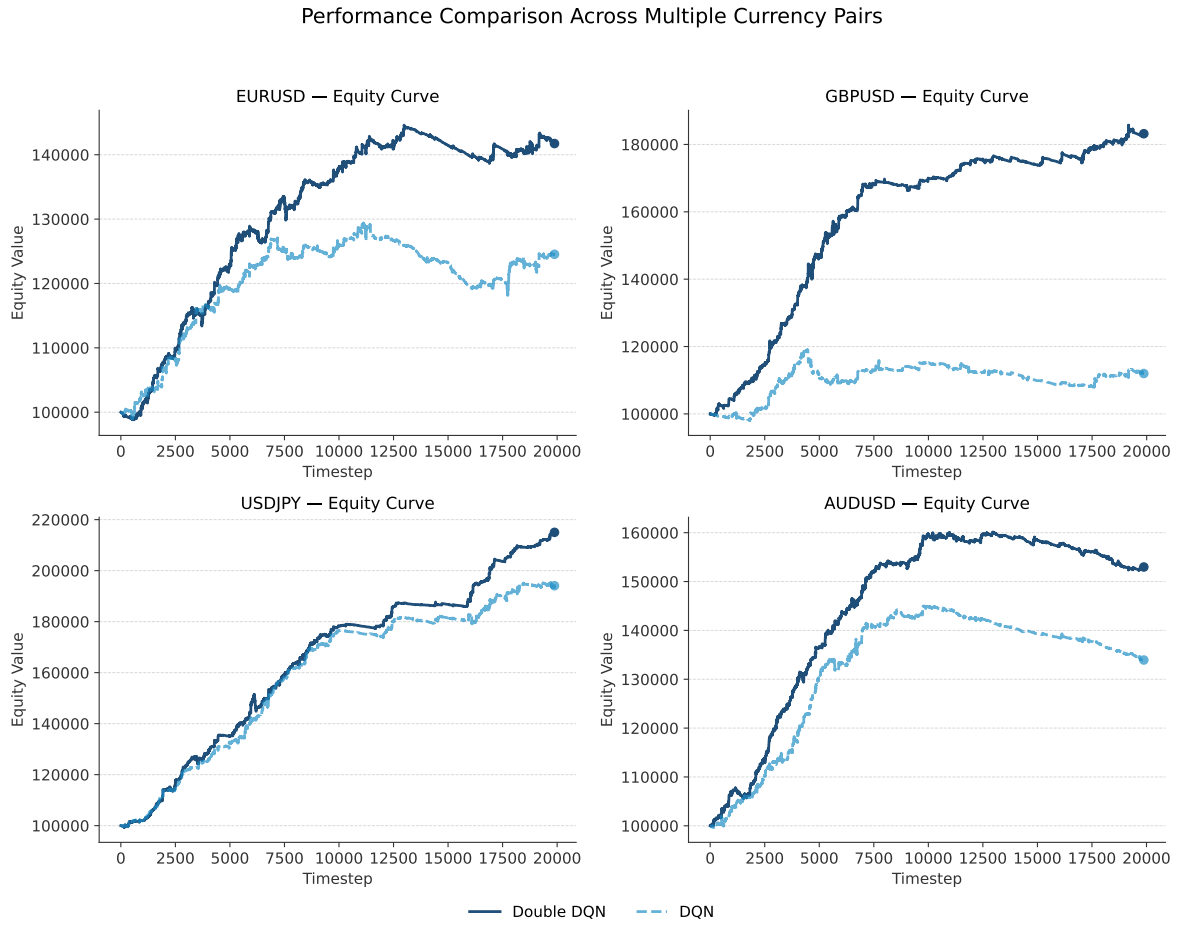

Figure 11. Multi-pair equity trajectories (EURUSD, GBPUSD, USDJPY, AUDUSD) for value-based agents under aligned simulator and reward settings.

6. Discussion

This section interprets the empirical findings through the lens of design objectives introduced earlier: transparent reward construction, legality-aware action control, and causal simulation fidelity. We focus on what can be inferred from controlled experimental evidence (see Section 4) and explicitly

 doi:10.20944/preprints202603.1701.v1

separate methodological insights from broader deployment claims, aligning with the evaluation rigor advocated by Zhang et al. [29].

6.1. Reward Decomposition as an Experimental Instrument

The principal methodological insight of this manuscript is that reward engineering should be treated as controlled experimentation rather than iterative tuning. The 11-component architecture, together with deterministic per-step logging, enables direct attribution of behavioral changes to specific reward terms. In EXP-01, performance does not improve monotonically as components are added; instead, quality degrades and recovers across the schedule before peaking at `r7`. This pattern demonstrates interaction effects that are difficult to diagnose under monolithic scalar objectives, which are commonly used in trading RL to optimize raw return or equity delta [16,26]. Unlike these conventional approaches, which obscure component-level attribution and complicate reward shaping analysis [19], our decomposable framework isolates the specific penalties that suppress excessive trading and the signals that improve entry timing.

6.2. Behavioral Consequences of Action Granularity

Endpoint metrics in Table 8 suggest a return–activity trade-off, but the mechanism is clearer when inspecting action usage directly. Figure 12 reports aggregate action frequencies for the compared interfaces.

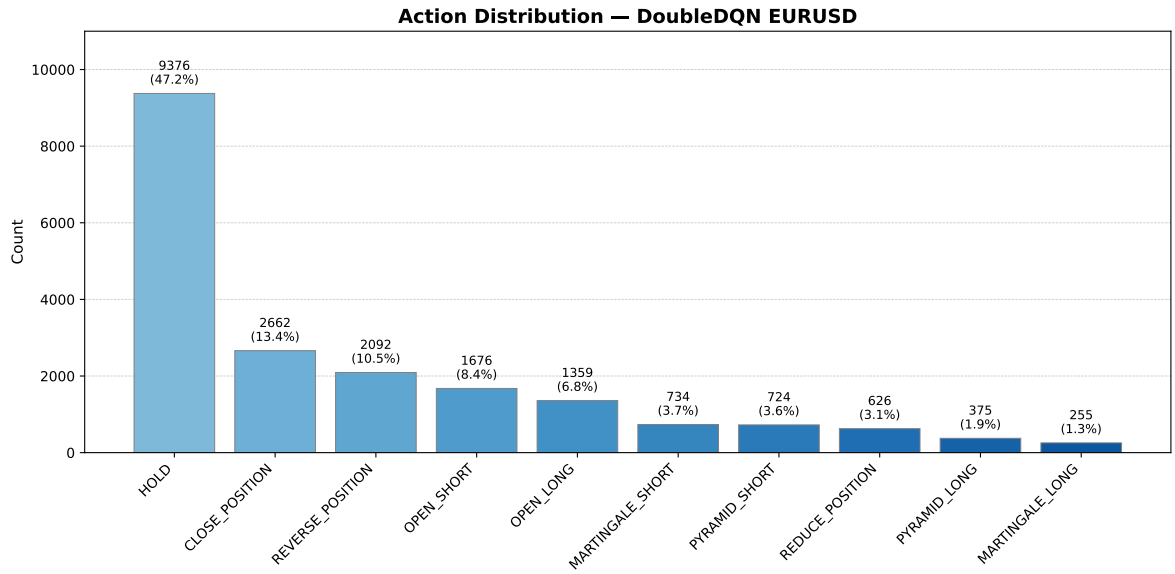

Figure 12. Aggregate action distributions across training runs, showing broader action usage and higher activity in the extended interface.

The distribution shows that richer action semantics are not merely available but actively exploited: scaling, reduction, and reversal primitives contribute substantial policy mass. This explains why the extended interface captures more return opportunities while simultaneously increasing turnover and path volatility.

6.3. Scaling Mechanics Under Asymmetric Penalties

The scaling family (EXP-03) is best interpreted through both endpoint values (Table 9) and operational depth statistics. Figure 13 visualizes average pyramid and martingale utilization by variant.

Two implications follow. First, the penalty asymmetry does shape behavior in the intended direction: enabling both mechanisms does not collapse into pure martingale usage. Second, utilization remains bounded, indicating that legality masks and margin controls are actively constraining excessive depth accumulation.

 doi:10.20944/preprints202603.1701.v1

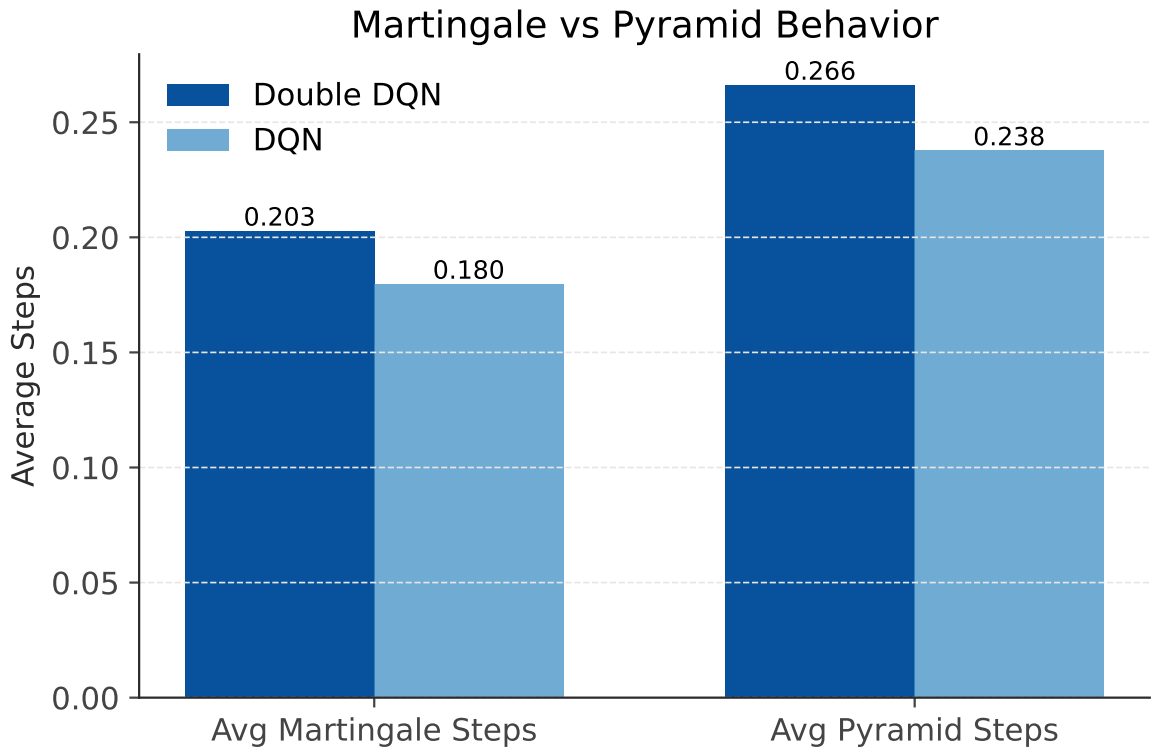

Figure 13. Episode-level average pyramid depth (AvgPyr) and martingale depth (AvgMart) by scaling variant.

6.4. Interpreting Endpoint Metrics

The retained results are intentionally restricted to the experimental scope defined in Section 4. Under this scope, endpoint metrics are informative about learning behavior but do not establish statistical dominance or deployment readiness. This distinction is central: the manuscript provides evidence about system dynamics, not definitive estimates of out-of-sample expected return. Consequently, directional claims should be interpreted as controlled observations that motivate multi-seed follow-up, in line with reproducibility concerns noted in prior surveys [29].

6.5. Layered Safety and Causality Guarantees

Two cross-cutting properties remain robust across experiments. First, strict anti-lookahead timing is enforced by environment mechanics rather than documentation convention, reducing leakage risk at the API level [29]. Second, legality masks are applied consistently during both interaction and target computation, following prior invalid-action masking practice [11]. Together with margin and liquidation controls, this yields a defense-in-depth design appropriate for high-risk financial RL experimentation.

6.6. Practical Interpretation

For practitioners, the key message is to align architecture choices with deployment objectives: decomposable rewards for auditability, action-space size for desired aggressiveness, and scaling configuration for explicit return–risk preferences, consistent with classical risk-return framing in portfolio analysis [15]. The present study provides an internally consistent, reproducible baseline for that design process; broader claims require multi-seed and out-of-sample validation.

7. Future Work

The present manuscript establishes a controlled, reproducible baseline for Forex reinforcement learning, but several limitations naturally define the next research stage. This section outlines those limitations as concrete future-work priorities.

 doi:10.20944/preprints202603.1701.v1

7.1. Limitations of the Current Study

All primary findings are training-split based (see Section 4 for canonical seeding and setup). As a result, the reported differences should be interpreted as directional evidence about system behavior rather than fully characterized performance distributions. Future studies should report multi-seed confidence intervals and wider temporal stress windows to quantify stability under regime variation.

The current experiments are pair-local: each environment instance models one currency pair at a time. This excludes cross-pair capital allocation, shared margin coupling, and correlation-aware portfolio control. Extending the framework to joint portfolio settings is required for full institutional realism.

7.2. Scalability and Algorithmic Extensions

Although the infrastructure is modular, scaling to larger instrument universes and longer histories will require more efficient replay management, distributed rollouts, and memory-aware training pipelines. From an algorithmic perspective, the release focuses on value-based baselines (DQN and DDQN) to isolate environment and reward effects. Future work should evaluate whether proximal policy optimization (PPO), soft actor-critic (SAC), and Rainbow-style extensions alter the observed reward and action-space trade-offs under identical simulator assumptions [9,10,22].

7.3. Potential Methodological Improvements

Position management is currently discretized with fixed lot and rule-bounded scaling depth. A natural extension is hybrid control that preserves legality constraints while permitting continuous sizing and explicit risk-budget targets. Reward calibration can also be expanded by testing adaptive weighting schedules and regime-conditioned penalties, while retaining the same auditable per-component logging introduced in this paper.

7.4. Real-World Deployment Challenges

The current evaluation is simulation-only. It does not yet incorporate broker-specific routing behavior, execution latency, partial fills, quote outages, or production risk controls (for example, kill-switch policies and operational monitoring). Addressing these constraints requires a staged transition from historical simulation to paper trading and, eventually, limited-capital live pilots under strict governance.

In summary, future progress should combine broader uncertainty quantification, portfolio-level scaling, richer algorithmic baselines, and deployment-grade execution modeling before making strong claims about operational readiness.

8. Conclusion

This paper presents a modular and reproducible framework for reinforcement learning-based Forex trading, centered on three design priorities: economic realism, temporal causality, and auditability. The framework combines anti-lookahead execution semantics, friction-aware portfolio accounting, decomposable reward modeling, and legal-action masking within a unified DQN-family training pipeline.

Summary of principal findings. Controlled experiments on EURUSD training artifacts yield three main outcomes (see Section 4 for canonical seeding and setup):

1. **Reward interactions are non-monotone.** Progressive reward composition (`r1`–`r7`) does not produce linear improvement. The full specification (`r7`) attains the strongest endpoint profile (Sharpe 0.765, cumulative return 57.09%), while intermediate variants exhibit both gains and regressions.
2. **Action granularity induces a measurable trade-off.** The extended 10-action interface increases cumulative return and turnover, whereas the simplified 3-action adapter yields stronger conservative risk summaries (higher Sharpe and lower drawdown) in the same endpoint view.

 doi:10.20944/preprints202603.1701.v1

3. **Scaling materially improves over no scaling.** All scaling-enabled settings outperform the no-scaling baseline on drawdown. The combined configuration (`s4`) achieves the best endpoint return in this artifact view, while martingale-only (`s3`) attains the lowest drawdown among scaling-enabled variants.

Contributions revisited. The core contribution is infrastructural: a research workflow in which reward decomposition, legality constraints, and anti-lookahead execution are encoded as enforceable system properties rather than informal conventions. This design improves reproducibility, supports component-level ablation, and enables more transparent diagnosis of trading-policy behavior.

Outlook. As detailed in Section 7, the next stage is to combine multi-seed uncertainty quantification, broader algorithmic baselines, and deployment-grade execution modeling. These extensions are necessary to convert controlled training evidence into robust out-of-sample and operational claims.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Full Hyperparameter Configuration

This appendix records the core configuration used across the release experiments. Main-paper values are summarized in Table 6; this appendix clarifies which settings were held constant and which were varied by experiment-family YAML overrides.

Appendix A.1. Shared Training Settings

- Total timesteps: 1,000,000
- Replay buffer size: 40,000
- Batch size: 128
- Learn start: 10,000 steps
- Learn frequency: every 4 environment steps
- Discount factor: $\gamma = 0.99$
- Optimizer: Adam (2.5×10^{-4})
- Gradient clip: 10.0
- Epsilon schedule: $1.0 \rightarrow 0.01$ over 30,000 steps
- Target update interval: 2,000 learning steps
- Mixed precision: fp16

Appendix A.2. Environment and Risk Settings

- Observation window: 24 bars
- Initial capital: USD 100,000
- Maximum leverage: 30x
- Maintenance margin ratio: 50% of initial margin
- Liquidation threshold: 25% of initial capital
- Commission: USD 3.5 per lot (round trip equivalent)
- Base deterministic slippage: 0.5 pips
- Rollover timestamp: 22:00 UTC (triple rollover Wednesday)

Appendix A.3. Experiment-Family Overrides

- EXP-01: reward-component enable/disable schedule (`r1`–`r7`)
- EXP-02: action-mode switch (simplified vs extended)
- EXP-03: scaling action availability (none, pyramid, martingale, both)

 doi:10.20944/preprints202603.1701.v1

Appendix A.4. Determinism Scope

All reported runs in this manuscript used a fixed seed; see Section 4 for the canonical seed value and related reproducibility details. This setting ensures deterministic reproducibility of the presented artifact snapshots but does not capture between-seed uncertainty.

Appendix B. Resolved YAML Configuration Snapshots

This appendix provides representative resolved configurations for reproducibility. Full resolved files are stored in run artifacts under experiment output directories.

Appendix B.1. Ablation Variant `r1` (Profit-Only Core)

```
experiment:
family: 01_reward_ablation
variant: r1_profit_only
agent:
name: doubledqn
model:
hidden_dims: [512, 512, 256]
training:
total_timesteps: 1000000
learn_start_steps: 10000
learn_frequency: 4
reward:
components:
profit: {enabled: true, weight: 1.0}
transaction: {enabled: false}
drawdown: {enabled: false}
volatility: {enabled: false}
overtrading: {enabled: false}
pyramid_penalty: {enabled: false}
martingale_penalty: {enabled: false}
holding: {enabled: false}
margin: {enabled: false}
liquidation: {enabled: false}
constraint: {enabled: false}
training:
random_seed: 42
```

Appendix B.2. Ablation Variant `r7` (Full Reward)

```
experiment:
family: 01_reward_ablation
variant: r7_full
agent:
name: doubledqn
reward:
components:
profit: {enabled: true, weight: 1.0}
holding: {enabled: true, weight: 0.03}
volatility: {enabled: true, weight: 0.01}
drawdown: {enabled: true, weight: 0.05}
transaction: {enabled: true, weight: 0.10}
overtrading: {enabled: true, weight: 0.02}
pyramid_penalty: {enabled: true, weight: 0.05}
martingale_penalty: {enabled: true, weight: 0.12}
margin: {enabled: true, weight: 0.05}
liquidation: {enabled: true, weight: 2.0}
constraint: {enabled: true, weight: 0.10}
reward_normalization:
mode: clip_only
clip_min: -1.0
clip_max: 1.0
training:
random_seed: 42
```

Appendix B.3. Action-Space Variants

```
# simplified
environment:
actions:
mode: simplified
```

 doi:10.20944/preprints202603.1701.v1


```
# extended
environment:
actions:
mode: extended
```

Appendix C. Anti-Lookahead Test Specification

This appendix formalizes the anti-lookahead guarantee enforced by the environment and summarizes corresponding validation tests.

Appendix C.1. Formal Timing Rule

Definition A1 (Decision–Execution Separation)**.** *At step t, the agent receives observations built from bars up to and including close_t. The chosen action a_t is executed at open_{t+1} under configured spread/slippage/commission rules. Reward and equity marking for that step are computed using information available at close_{t+1}, without access to bar $t+2$ or later.*

Appendix C.2. Validation Test Cases

Test 1: Feature staleness check. Inject a sentinel value into a future bar and verify that the step-t observation tensor does not expose this sentinel.
Test 2: Fill-price rule check. For deterministic synthetic bars, verify execution at open_{t+1} (not close_t and not close_{t+1}).
Test 3: Reward-timing check. Verify that reward at step t depends on costs and marks from bar $t+1$ only.
Test 4: Scaler leakage check. Verify that feature scaling parameters are fitted on the train split only and, if a held-out split is instantiated, reused unchanged during held-out transformation.
Test 5: Mask-timing check. Verify that legal-action masks are computed from current state/margin before action dispatch, and that illegal-action coercion/penalty is logged without future leakage.

Appendix C.3. Interpretation

Passing these tests does not guarantee real-market validity, but it materially reduces a major source of optimistic bias in trading RL backtests: unintended access to future information through timing or preprocessing pathways.

Appendix D. Data and Code Availability

The implementation code and relevant configuration files used to reproduce the experiments in this manuscript are available in a public GitHub repository: https://github.com/NabeelAhmad9/frl_trading_framework.

Processed data artifacts used in this study are available from the corresponding author upon reasonable request.

References

1. M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 449–458, 2017.
2. F. Bertoluzzo and M. Corazza. Testing different reinforcement learning configurations for financial trading: Introduction and applications. *Procedia Economics and Finance*, 3:68–77, 2012.
3. G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.
4. J. a. Carapuco, R. Neves, and N. Horta. Reinforcement learning applied to forex trading. *Applied Soft Computing*, 73:783–794, 2018.
5. R. Cont. Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance*, 1(2): 223–236, 2001.

 doi:10.20944/preprints202603.1701.v1

6. Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664, 2017.
7. T. Fischer and C. Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.
8. M. D. Gould, M. A. Porter, S. Williams, M. McDonald, D. J. Fenn, and S. D. Howison. Limit order books. *Quantitative Finance*, 13(11):1709–1742, 2013.
9. T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870, 2018.
10. M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.
11. S. Huang and S. Ontañón. A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*, 2020.
12. Z. Jiang, D. Xu, and J. Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.
13. X.-Y. Liu et al. FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*, 2021.
14. G. Lucarelli and M. Borrotti. Deep reinforcement learning for automated stock trading: An ensemble strategy. *Expert Systems with Applications*, 117:126–139, 2019.
15. H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.
16. T. L. Meng and M. Khushi. Reinforcement learning in financial markets. *Data*, 4(3):110, 2019.
17. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
18. J. Moody and M. Saffell. Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12 (4):875–889, 2001.
19. A. Y. Ng, D. Harada, and S. J. Russell. Policy invariance under reward transformations: Theory and application to shaped reward reinforcement learning. In *Proceedings of the 16th International Conference on Machine Learning*, pages 278–287, 1999.
20. F. Rundo. Deep LSTM with reinforcement learning layer for financial trend prediction in FX high frequency trading systems. *Applied Sciences*, 9(20):4460, 2019.
21. T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.
22. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.
23. W. F. Sharpe. The Sharpe ratio. *Journal of Portfolio Management*, 21(1):49–58, 1994.
24. F. A. Sortino and L. N. Price. Performance measurement in a downside risk framework. *Journal of Investing*, 3(3):59–64, 1994.
25. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.
26. T. Theate and D. Ernst. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632, 2021.
27. H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.
28. Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1995–2003, 2016.
29. Z. Zhang, S. Zohren, and S. Roberts. Deep reinforcement learning for trading. *Journal of Financial Data Science*, 2(2):25–40, 2020.