OpenFinGym: A Verifiable, Multi-Task Gym Environment for Evaluating Quant Agents

Anonymous ACL submission

Abstract

Although large language model agents are increasingly applied to quantitative-finance workflows, their evaluation remains fragmented across isolated tasks, while the financial relevance of benchmark tasks is often overlooked. Yet financial workflows are inherently multi-stage, spanning interdependent tasks such as forecasting, strategy construction, risk management, and trading. Existing platforms typically focus on a single task, and can therefore overstate agent competence and fail to reveal weaknesses in generalization, real-market interaction, and financially meaningful decision-making. We introduce OPENFINGYM, a unified gym environment for quantitative-finance agent development that covers forecasting, market generation, real-time trading, and fraud detection under a single execution and verification interface. OPENFINGYM additionally provides an automated task-construction pipeline that turns quantitative finance publications into executable task packages; a containerised runtime with a host-side verifier service that supports scalable agent rollouts and prevents runtime train–test leakage; a paper trading engine with a low-latency data-stream design; deferred-resolution support for long-horizon and event-market forecasts; and integration for SFT and RL post-training.

1 Introduction

Quantitative finance has, over the past two years, emerged as a testing ground for autonomous large language model agents (Zhang et al., 2024; Yu et al., 2025b, 2024). Where earlier work investigated whether deep models could outperform classical baselines on a forecasting or trading task (Gu et al., 2020a; Sun et al., 2023; Liu et al., 2022), recent systems treat the LLM itself as a researcher: an agent that selects data, engineers features, trains models, generates predictions or executes trades. This shift changes what evaluation and training infrastructure must support. The task surface should span future price forecasting (jun Gu et al., 2024; Wang et al., 2024), generating synthetic market scenarios for stress testing (Hounwanou and Gaba, 2025), executing trades against historical and live data (Zong et al., 2024; Kashif and Ślepaczuk, 2025), and detecting anomalies in transaction graphs (Li et al., 2024). It also should span interaction modes, from one-shot batch submission on historical data to step-by-step decision-making on streaming market feeds.

Although some existing platforms such as FinRL-Meta (Liu et al., 2022), TradeMaster (Sun et al., 2023) supply gym-style trading environments, and domain-general gym environments such as DSGym (Nie et al., 2026) and TimeSeriesGym (Cai et al., 2025) can handle static tasks such as batch forecasting, the former are restricted to trading and the latter are not specialised for finance. Yet financial tasks are interconnected in practice: forecasts can guide trades, generated scenarios can stress-test strategies, and anomaly signals can inform risk management. No existing platform provides a unified environment for evaluating and training agents across forecasting, generation, trading, and anomaly detection under a common execution and verification interface, which is needed to assess whether agents can perform and generalize across financial workflows. Moreover, in a high-stakes domain such as finance, scalable task construction must be paired with rigorous verification: tasks should reflect meaningful financial workflows while ensuring reproducibility, leakage control, and evaluation against hidden ground truth.

We introduce OPENFINGYM, a unified gym environment for quantitative-finance agent development that closes this gap. OPENFINGYM contains a curated suite of 78 tasks across the four task families with an automated, literature-to-task pipeline. Agents run inside containers that exclude held-out

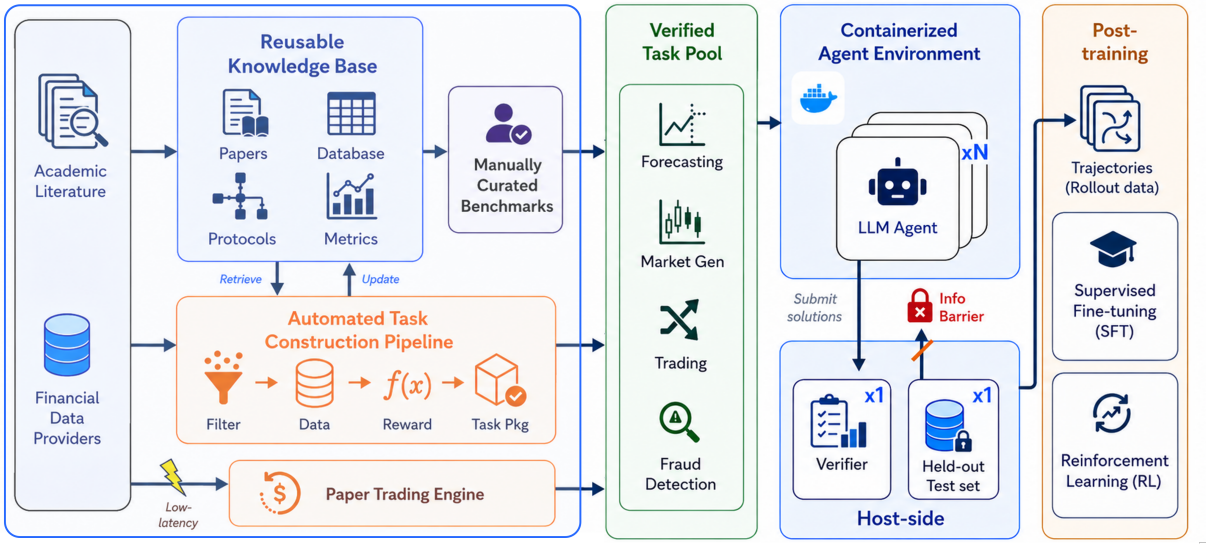


Figure 1: A high-level illustration of OpenFinGym architecture

test set ground truth, and submit predictions or actions to a host-side verifier service for reward computation. A low-latency data streaming service, a paper trading engine, and a persistent SQLite ledger extend the same interface to real-time trading and long-horizon price and prediction market forecasting. We benchmark four frontier LLMs on OpenFinGym and find task-family-specific leaders rather than a single dominant model. We also demonstrate OpenFinGym's utility as a post-training environment, where SFT and GRPO post training improves the executable generation success rates of Qwen3 base models from 0% to 100% on held-out forecasting tasks, and improves the reward by an order of magnitude on Treasury.

Our contributions are as follows:

- A curated task suite of 78 verified quantitative-finance tasks spanning four different task families, constructed from prior literature and extended to real-time financial market environment.
- An automated task-construction pipeline that converts academic quantitative-finance papers into executable and verifiable task packages.
- A containerised runtime that separates agent-visible information from verifier-only ground truth, enabling leakage-resistant evaluation across both batch and sequential tasks.
- A gym environment for evaluating and training quantitative-finance agents, which enables supervised fine-tuning and reinforcement learning.

2 Related work

Financial agent environments include reinforcement learning trading platforms such as FinRL-Meta (Liu et al., 2022), TradeMaster (Sun et al., 2023), quantitative research frameworks such as Qlib (Yang et al., 2020), and market microstructure simulators such as ABIDES-Gym (Amrouni et al., 2021). These systems primarily support quantitative research, trading, and simulation rather than executable agentic benchmarking grounded in financial literature. Financial LLM benchmarks, including PIXIU (Xie et al., 2023), FinBen (Xie et al., 2024), FinanceBench (Islam et al., 2023) evaluate financial reasoning and analysis abilities on largely static datasets and evaluation protocols. More recent work has focused on executable agent evaluation. SWE-Bench (Jimenez et al., 2024), AgentBench (Liu et al., 2024), and WebArena (Zhou et al., 2024) evaluate agents in autonomous environment, while, DSGym (Nie et al., 2026), TimeSeriesGym (Cai et al., 2025), and Harbor (Harbor Framework Team, 2026) extend this paradigm to data-science, time-series, and containerised execution settings. OpenFinGym builds on this line of work but targets finance-specific challeges such as temporal leakage and economically grounded evaluation. Concurrent studies further investigate LLM-based financial agents in live or semi-live markets, such as StockBench (Chen et al., 2025), LiveTradeBench (Yu et al., 2025a), and When Agents Trade (Qian et al., 2025). Other works highlight the look-ahead risks, temporal contamination,

System	Task coverage					Runtime		Agent lifecycle	
	Fore-casting	Trading (offline)	Trading (real-time)	Market generation	Fraud detection	Leakage control	Low-latency	Auto task construction	SFT / RL integration
Quantitative finance agent systems									
FinRL-Meta (Liu et al., 2022)	–	✓	∼	–	–	∼	–	–	∼
FinBen (Xie et al., 2024)	∼	✓	–	–	✓	–	–	–	∼
INVESTORBENCH (Li et al., 2025a)	–	✓	–	–	–	∼	–	–	–
When Agents Trade (Qian et al., 2025)	–	∼	✓	–	–	∼	–	–	–
General containerised agent gyms									
SWE-Bench (Jimenez et al., 2024)	–	–	–	–	–	✓	–	∼	–
TimeSeriesGym (Cai et al., 2025)	✓	–	–	–	–	✓	–	∼	–
Harbor (Harbor Framework Team, 2026)	–	–	–	–	–	✓	–	–	✓
OPENFINGYM (ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of OPENFINGYM against competitive systems across task coverage, runtime properties, and agent-lifecycle support. ✓ denotes full support, ∼ partial support, and – no support.

and benchmark leakage in financial agent evaluation (Benhenda, 2026; Li et al., 2025b; Xu et al., 2024). OPENFINGYM is positioned at the intersection of executable agent evaluation and quantitative-finance benchmarking through literature-grounded task construction and verifier-isolated evaluation.

3 Open Financial Gym

OPENFINGYM is designed to support rigorous, scalable, and leakage-resistant evaluation of quantitative-finance agents. OPENFINGYM covers a broader family of financial workflows, including forecasting, trading, market generation, and fraud detection. The benchmark is organised around three components: a reusable quantitative-finance knowledge base, a collection of representative and verified tasks, and a gym-style execution environment for evaluating and training agents. The overall architecture is shown in Figure 1.

3.1 Task Specifications

Unified task abstraction. The quantitative-finance tasks in OPENFINGYM are formulated as machine-learning problems. Formally, each task is represented by a triplet $(\mathcal{D}, \mathcal{T}, \mathcal{E})$, where

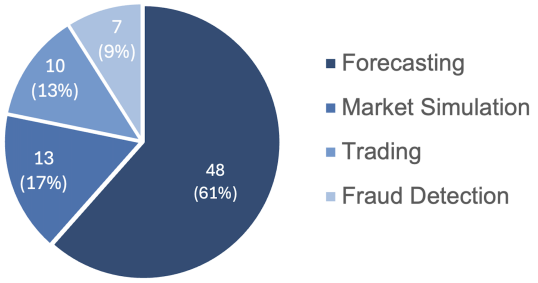
- $\mathcal{D}$ denotes the dataset, including the training and test splits;
- $\mathcal{T}$ specifies the task description, including the data source, input-output interface, prediction or decision objective, required output format, and overall goal of the task;
- $\mathcal{E}$ denotes the evaluation protocol used to assess agent performance. The evaluation protocol may include multiple metrics, together with an aggregation rule that defines the final task score.

This formulation provides a unified interface for representing heterogeneous quantitative-finance problems.

Each task is packaged in a standardised format extended from by the Harbor framework (Harbor Framework Team, 2026). A task package contains the agent-facing instructions, task interface module, data payload, configuration file used for execution and orchestration, and the evaluation components required by the runtime. A key design principle is the separation between information available to the agent and information used only for verification. During evaluation, agents may access the training split and the input features of the test split, but the corresponding labels, realised outcomes, or hidden evaluation state are withheld and made available only to the verifier. This separation reduces leakage risk and allows the same agent to be evaluated across heterogeneous task sources without task-specific glue code.

Task taxonomy. OPENFINGYM covers four task families.

1. **Forecasting.** Forecasting tasks require an agent to predict future financial quantities, such as returns, prices, volatility, yield-curve movements, or market states, from historical observations and other predictive signals. These tasks are evaluated using statistical metrics such as R^2, RMSE, and directional accuracy, as well as finance-specific metrics that measure the downstream economic value or risk profile of the forecasts.
2. **Trading.** Trading tasks require an agent to take sequential trading actions on the observed market snapshots. Offline trading tasks evalu-



Task type	Asset Classes	# Tasks
Forecasting	Equities, Commodities, FX, Crypto, LOB, Yield Curve	48
Market Simulation	Equities, FX, Crypto, LOB, Synthetic Processes	13
Trading	Equities, Crypto	10
Fraud Detection	Transactional, Behavioral, and Graph-Structured Signals	7

Table 2: Task type distribution (top) and asset class breakdown (bottom) of the curated financial tasks in the gym.

ate policies on historical data, while real-time trading tasks require sequential interaction with live market observations. Evaluation metrics include profit and loss, Sharpe ratio, drawdown, turnover, and returns with configurable transaction cost and price slippage.

3. **Market generation.** Market generation tasks evaluate whether an agent can generate realistic financial time series, market states, or scenario distributions. Generated samples are compared with real data using distributional, temporal, and financial diagnostics, including distributional scores, statistic-based scores (e.g., autocorrelation score, marginal distribution score), volatility, tail-risk, and return-dynamics statistics.

4. **Fraud detection.** Fraud detection tasks are formulated as classification, ranking, or anomaly-detection problems. The objective is to identify fraudulent, suspicious, or anomalous financial activity. Evaluation metrics include AUROC, recall, and threshold-dependent operating metrics.

The task catalogue spans multiple asset classes, including equities, indices, foreign exchange (FX), commodities, crypto-assets, rates, limit-order-book (LOB) microstructure, and prediction markets, as well as different frequencies ranging from intraday to monthly observations and graph-structured data. It includes offline tasks derived from the academic literature and real-time tasks in which agents must act before the relevant market or event outcome is observed. The resulting task universe is summarised in Table 2.

Knowledge base. Task construction produces a reusable quantitative-finance knowledge base. The knowledge base stores structured information about papers, datasets, task settings, experimental protocols, evaluation metrics, and reusable reward functions. This enables datasets and metrics to be reused across related tasks, supports systematic construction of new benchmark instances, and provides the substrate for the automated task construction pipeline described in Section 4.

3.2 Agents

An agent receives the task instructions, the observable data, and the required output contract. For batch tasks, the agent typically writes executable code that trains a model on the provided training data and submits predictions for the held-out test inputs. For sequential tasks, the agent interacts with the environment through a step-based interface, observes market or portfolio state, and submits actions such as forecasts, trades, or portfolio weights.

The agent is not required to know how the verifier stores hidden labels or realised outcomes. Instead, it interacts with the task through a fixed submission interface. This design allows agents with different internal architectures, such as prompted language-model agents, tool-using agents, coding agents, or trained policies, to be evaluated under the same task interface.

4 Methodology

OpenFinGym has two methodological components. The first is a task-construction pipeline that turns quantitative-finance publications and curated benchmark specifications into executable tasks. The second is a task-execution framework that runs agents in controlled environments and evaluates their submissions through isolated verifier processes or deferred-resolution services.

4.1 Task construction

OpenFinGym constructs tasks through three complementary pathways: automated task generation, curated benchmark implementation, and trading task routing. Forecasting and market-generation

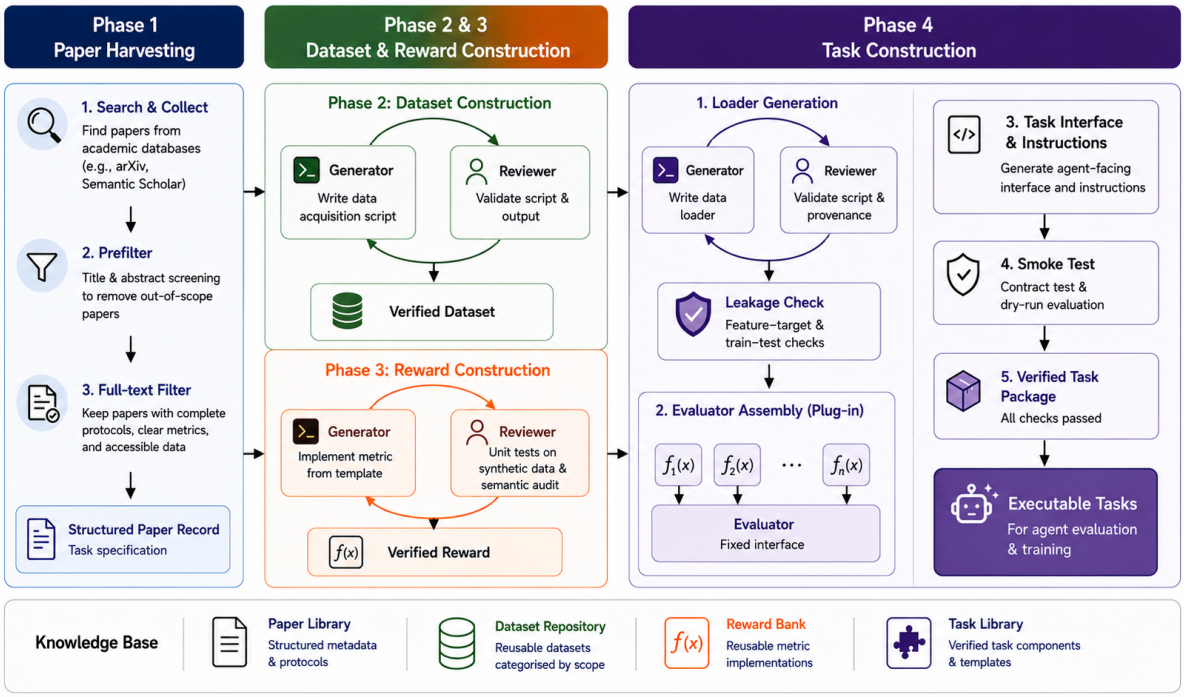


Figure 2: Schematics of the four phases of the task construction pipeline and the knowledge base

tasks are well suited to automated construction because they typically follow a common pattern: materialise a dataset, define an input-output split, generate predictions, and evaluate them against held-out targets. Trading and real-time tasks require additional infrastructure, including execution engines, market-data providers, portfolio-state tracking, and deferred evaluation; these tasks are therefore implemented through curated harnesses that can be parameterised by paper-specific configurations.

4.1.1 Automated Task Construction pipeline

OpenFinGym constructs its knowledge base and executable benchmark tasks through a multi-stage pipeline, comprising four main phases: paper harvesting, dataset construction, reward construction, and task generation, as shown in Figure 2.

Paper harvesting. The pipeline first collects candidate papers academic finance within a user-specified task scope, then filters them through a two-stage procedure. A prefiltering stage screens titles and abstracts to discard out-of-scope papers; A full-text filtering stage admits a candidate only if it presents a complete, in-scope experimental protocol together with well-documented evaluation metrics and at least one publicly accessible datasets. Accepted papers are summarised into structured records detailing its task setting, datasets, evaluation metrics, and experimental protocol. These records form the initial substrate of the knowledge base and provide the task specifications for the downstream construction stages.

Dataset and Reward construction. For each accepted task specification, the pipeline first check for potential reuses of existing datasets and rewards in the knowledge base. Otherwise, the required artifact is materialised via a generator-reviewer loop, in which an LLM writes the artifact and a reviewer judges it with the aid of unit tests. For dataset construction, the loop runs an LLM-authored download script in a safety sandbox, with the reviewer judging both the script and the materialised payload against the task's metadata. Evaluation metrics undergo similar reuse checks against a reusable reward bank. A metric absent from the bank is generated by the same generator-reviewer loop from a standardised template. It is admitted only after interface checks, unit tests on synthetic tensors and the reviewer's audit. Together these stages populate a reusable quantitative-finance knowledge base of papers, datasets and reward functions organised by task scope.

Task assembly. The pipeline assembles the datasets, rewards, and task specifications into a self-contained, OpenFinGym-structured task package. Where applicable, the dataset is partitioned into train and test sets by a task-specific loader gen-

erated through the same generator-reviewer loop. The generator step also emits a ground-truth provenance record (e.g. source column, prediction horizon) that the reviewer audits for source-mechanism consistency. A separate feature-leakage test then checks over the loader's output tensors before the loader is accepted. An evaluator is then assembled deterministically from a template by plugging each resolved reward, as specified by the task's metadata. A final generation step writes the agent-facing task interface module, declaring the task's metadata and observation / action spaces, and writes an instruction markdown file detailing task goal, dataset description, and evaluation protocol. The task is installed only after it passes both contract and smoke tests that exercise the full agent–evaluator path. Each package bundles these artifacts with environment configuration files and dataset payload, forming executable benchmark tasks consumable by the OPENFINGYM runtime for agent evaluation and downstream training.

4.1.2 Generated and curated task packages

The automated pipeline of Section 4.1.1 emits a task package from each accepted paper. For forecasting, market-generation and fraud-detection papers, this pipeline materialises the full executable package end to end: dataset, evaluator, task module and instructions. Additionally, we construct offline benchmarks within each scope by manually selecting papers that are widely cited, methodologically reproducible, and accompanied by either public code or a sufficiently detailed protocol. Each task is then implemented and manually verified to reproduce the original literature benchmark as faithfully as possible. The full summary of the curated offline tasks is given in Appendix C.

For trading tasks, the required execution and market-data infrastructure cannot be reliably built by the automated pipeline alone. The pipeline instead routes the task specification to the curated trading harness and re-exports a task parameterised by the paper-derived configuration (symbols, dates, frequency, prediction horizon, etc.). The harness reused here is the same one used by the manually curated trading tasks, which bundles a configurable execution engine, in-process portfolio-state accounting, and adapters to stock, crypto, and prediction-market data providers.

Table 3: Latency and peak refresh rate measurements of the WebSocket (WS) and REST paths of data providers for real-time trading tasks. Both measures are done in a 15 s window. The WS path incurs an additional $\approx 0.5\,\mu$s buffer lookup time on the agent's side. Results may vary depend on users' locations and market conditions.

Provider	Channel	Latency (ms)	Speed-up (Latency)	Peak refresh (Hz)	Improvement (Freshness)
Alpaca	WS	60.5	1.85×	48	24×
	REST	112.0		2	
Binance	WS	125.8	1.91×	67	22×
	REST	240.3		3	

4.1.3 Real-time tasks

Real-time tasks are another case where OPENFINGYM relies on a curated harness rather than the automated pipeline. The agent takes trading actions or makes forecasts on streaming market data, and the runtime requirements cannot be reliably built by an automated pipeline alone such as low-latency data streaming, in-process order execution, and long-horizon deferred resolution.

Real-time trading tasks. The agent observes live market and portfolio state at each gym step and submits orders through an executor interface. The default is an in-process paper trading simulator that holds positions, cash, and pending orders in memory and fills orders against the latest market snapshot. It supports the common order types and applies configurable price slippage and per-trade transaction costs. The alternative executor routes the same orders to a broker's paper trading sandbox and mirrors the resulting account state locally, keeping account and portfolio reads constant-time on the agent side without a broker round-trip per observation. Further details of the trading implementation are given in Appendix A.

Latency-aware data path. Each gym step in a real-time trading task reads the current price and order book before the agent acts, and the freshness of those reads to the quality of the agent's actions. A naive REST-per-step approach incurs a network round-trip on every observation and reaches rate limits well below the price refresh rate. OPENFINGYM instead serves these data from an in-process buffer that is continuously refreshed by a background WebSocket (WS) subscription. The WS path roughly halves the per-read latency and achieves a peak refresh rate more than an order of magnitude higher than REST, as shown by Table 3.

Model	Forecasting		Generation		Trading		Fraud Detection	
	Reward ↓	Success (%) ↑	Reward ↓	Success (%) ↑	Reward ↑	Success (%) ↑	Reward ↑	Success (%) ↑
GPT-4o	0.052	100	7.377	70	**19.912**	100	1.290	100
GPT-5.1-codex-mini	**0.036**	100	11.212	80	18.090	100	1.205	100
Haiku 4.5	0.058	100	14.772	90	11.490	100	1.147	100
Sonnet 4.6	0.053	100	**7.114**	90	12.044	100	**1.630**	100

Table 4: Agent performance across task families and LLM models. For each task family, we report both the average reward score and the task success rate, aggregated over all tasks within that category. The reward score is computed as a weighted combination of multiple evaluation metrics, providing a comprehensive assessment of overall agent performance. ↓ represents lower is better while ↑ represents higher is better.

Deferred-resolution forecasting tasks. OPENFINGYM also accommodates price and event market forecasting on real-time data. This raises a different problem: the realised outcome of a long-horizon price forecast or an event market prediction is not available at the moment of submission. The verifier therefore records each submission together with its reference state and the resolution time in a persistent SQLite ledger. Once the resolution time has elapsed, a resolver service fetches the realised market price or event outcome from the relevant data provider and computes the configured reward.

4.2 Task execution

OPENFINGYM executes tasks through containerised agent environments and a verifier-based scoring service. Each agent rollout runs inside a container that holds the task instructions, the training data (both inputs and labels), test inputs, task interface, and the Python libraries required for modelling and submission. The held-out test labels are explicitly excluded from the container's bind mounts and are therefore inaccessible to the agent, removing test-set leakage during runtime by construction.

The verifier is a host-side process that loads the held-out test set ground truth into memory at startup and exposes a submission API. The API has a rate limit that prevents the reverse engineering of the held-out ground truth. The verifier process can be shared across every agent container running the same task concurrently, making the setup suitable for large-scale batch evaluation, batch rollouts for RL, and trajectory collection for SFT. On each call, the verifier validates the submission format, computes the reward host-side, and returns the per-metric scores together with a weighted total.

OPENFINGYM supports two interaction surfaces: Batch tasks, such as forecasting, use a one-shot submission interface, where the agent submits all predictions in a single call, which are scored immediately or after deferred resolution; Gym-loop tasks, such as trading, use a sequential interface, where the agent repeatedly observes state, takes actions, accumulates a trajectory, and is evaluated at the end of the episode.

5 Numerical Results

5.1 Agent Performance

We evaluate a representative subset of OpenFinGym comprising 24 tasks across four families: 5 forecasting tasks, 10 market-generation tasks, 2 trading tasks, and 7 fraud-detection tasks. Results are summarised in Table 4. For each task family, we report both the average reward score and the task success rate. The success rate measures the percentage of tasks for which the agent successfully generates executable submissions that satisfy the task interface and can be evaluated by the OpenFinGym verifier harness.

Several observations emerge from the results. First, all evaluated agents achieve 100% success rates on forecasting, trading, and fraud-detection tasks, indicating that frontier models can reliably interact with the OpenFinGym interfaces and produce executable quantitative-finance solutions. In forecasting, GPT-5.1-codex-mini performs best with a reward of 0.036, clearly ahead of the other agents whose rewards range from 0.052 to 0.058. In market generation, Sonnet 4.6 gives the strongest result with the lowest reward of 7.114, slightly better than GPT-4o at 7.377 and substantially better than the remaining agents. For trading, GPT-4o achieves the highest reward of 19.912, followed by GPT-5.1-codex-mini at 18.090, indicating stronger trading-strategy quality among the evaluated agents. In fraud detection, Sonnet 4.6 obtains the best reward of 1.630, outperforming GPT-4o by 26.4%, GPT-5.1-codex-mini by 35.3%, and Haiku

Model	Commodity		Crypto		Equity		Treasury		FX	
	Reward ↓	Success ↑	Reward ↓	Success ↑	Reward ↓	Success ↑	Reward ↓	Success ↑	Reward ↓	Success ↑
Qwen3 1.7b	–	0	–	0	–	0	–	0	–	0
SFT	0.023	60	0.539	30	0.016	60	1.152	60	**0.006**	40
RL	**0.020**	100	**0.007**	100	**0.015**	100	**0.099**	100	**0.006**	100
Qwen3 4b	–	0	–	0	–	0	–	0	–	0
SFT	0.019	70	0.007	50	0.015	100	1.055	60	**0.006**	90
RL	**0.018**	100	**0.006**	100	**0.014**	100	**0.090**	100	**0.006**	100
Qwen3 8b	–	0	–	0	–	0	–	0	–	0
SFT	0.019	90	0.007	90	0.015	70	0.803	70	0.006	70
RL	**0.018**	100	**0.006**	100	**0.014**	100	**0.090**	100	**0.005**	100

Table 5: The performance of base model, SFT and RL results on the five held-out forecasting test tasks, averaged over 10 random seeds per task. The first row of each model is the base checkpoint. We report the mean reward (↓ lower is better) and the task success rate (%, ↑ higher is better). Base success rates are uniformly 0%.

4.5 by 42.1%.

Overall, Table 4 shows that OpenFinGym reveals task-specific agent strengths rather than a single dominant model: GPT-5.1-codex-mini leads on forecasting, Sonnet 4.6 leads on market generation and fraud detection, and GPT-4o leads on trading. These results demonstrate that OpenFinGym can differentiate agent capabilities across heterogeneous quantitative-finance settings while supporting reliable large-scale automated evaluation.

5.2 SFT & RL Results

OpenFinGym is integrated with both SFT and RL post-training. To demonstrate its utility as a post-training environment for quant agents, we focus on forecasting tasks under a single-shot generation setting, in which the agent emits its full submission in one pass without iterative tool use.

Our SFT corpus consists of 189 single-shot trajectories collected from Claude Opus 4.7 at low reasoning effort over 21 forecasting training tasks. Agents are encouraged to take classical machine learning approaches over generating neural network architectures and training loops from scratch, as the latter are more difficult for small base models to tackle within a single shot.

We fine-tune three Qwen3 backbones (Yang et al., 2025) of sizes 1.7B, 4B, and 8B with LoRA (Hu et al., 2022) for two to four epochs. The merged LoRA weights then serve as the initialization for RL, where each SFT checkpoint is post-trained with GRPO (Shao et al., 2024) under SkyRL (Cao et al., 2025). With a batch size of 4 prompts per gradient step and 8 rollouts per prompt, training runs for 50–100 epochs (∼250–500 gradient steps), with reward defined as the negative weighted averages of per-task metrics. Full hyperparameter settings are listed in Appendix B.

Table 5 reports mean reward and success rate on the five held-out forecasting tasks. Base Qwen3 of every size fails to produce an executable submission on any task. SFT closes most of this gap, lifting mean success to roughly 50–80%. RL then boosts every model to a 100% success rate on every task, with mean reward at or below the corresponding SFT value across the board, including order-of-magnitude reductions on Treasury and on Crypto. We view these results as a demonstration of OpenFinGym's SFT and RL integration.

6 Conclusion

We introduced OpenFinGym, a comprehensive benchmark and training environment for quant agents. By combining executable task packages, verification mechanisms, and an automated task-construction pipeline, this gym supports rigorous benchmarking and downstream agent training across diverse financial workflows.

The future works on OpenFinGym includes the extension to a broader range of financial task types, including additional execution settings, risk-management workflows, and decision-making problems beyond the current task families. The existing task-generation pipeline already provides a scalable path for this expansion. Another future direction is to expose OpenFinGym's authenticated verifier interface as a public evaluation service, allowing external users to evaluate their agents without downloading the full benchmark or accessing test labels. This would enable fair comparison, supports for leaderboard-style evaluation, and help establish OpenFinGym as a community platform for quant agent developments.

7 Limitations

While OpenFinGym provides a unified environment for evaluating and training quant agents, our current SFT and RL results should be viewed as an initial demonstration rather than a fully optimized training recipe. Due to time and computational constraints, the trajectories used for post-training are limited in both scale and quality, and the backbone agent used in our experiments is relatively lightweight. These choices constrain the attainable gains from supervised fine-tuning and reinforcement learning. Future work will scale trajectory collection, improve trajectory filtering and reward-based data selection, incorporate stronger backbone models, and study larger-scale RL post-training across more diverse financial tasks.

References

Selim Amrouni, Aymeric Moulin, Jared Vann, Svitlana Vyetrenko, Tucker Balch, and Manuela Veloso. 2021. Abides-gym: gym environments for multi-agent discrete event simulation and application to financial markets. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9.

Yihao Ang, Yifan Bao, Lei Jiang, Jiajie Tao, Anthony KH Tung, Lukasz Szpruch, and Hao Ni. 2025. Structured agentic workflows for financial time-series modelling with llms and reflective feedback. In *Proceedings of the 6th ACM International Conference on AI in Finance*, pages 924–932.

Yifan Bao, Xinyu Xi, Liu Xinyu, Wen Ge, Lei Jiang, Kaicheng Zhang, Raad Khraishi, Yihao Ang, Anthony Kum Hoe Tung, Lukasz Szpruch, and Hao Ni. 2026. MOSAIC: Modular orchestration for structured agentic intelligence and composition.

Mostapha Benhenda. 2026. Look-ahead-bench: A standardized benchmark of look-ahead bias in point-in-time llms for finance. *Preprint*, arXiv:2601.13770.

Yifu Cai, Xinyu Li, Mononito Goswami, Michał Wiliński, Gus Welter, and Artur Dubrawski. 2025. Timeseriesgym: A scalable benchmark for (time series) machine learning engineering agents. *Preprint*, arXiv:2505.13291.

Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning.

Yanxu Chen, Zijun Yao, Yantao Liu, Jin Ye, Jianing Yu, Lei Hou, and Juanzi Li. 2025. Stockbench: Can llm agents trade stocks profitably in real-world markets? *Preprint*, arXiv:2510.02209.

Ilya Chevyrev and Harald Oberhauser. 2022. Signature moments to characterize laws of stochastic processes. *Journal of Machine Learning Research*, 23(176):1–42.

Rama Cont, Mihai Cucuringu, Renyuan Xu, and Chao Zhang. 2026. Tail-gan: Learning to simulate tail risk scenarios. *Management Science*, 72(4):2917–2936.

Francis X. Diebold and Robert S. Mariano. 1995. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263.

Francis X Diebold and Kamil Yilmaz. 2012. Better to give than to receive: Predictive directional measurement of volatility spillovers. *International Journal of forecasting*, 28(1):57–66.

Bivas Dinda. 2024. Gated recurrent neural network with tpe bayesian optimization for enhancing stock index prediction accuracy. *arXiv preprint arXiv:2406.02604*.

Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. 2017. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*.

European Insurance and Occupational Pensions Authority (EIOPA). 2026. Risk-free interest rate term structures. Official monthly Solvency II risk-free interest rate term structures. Accessed: 2026-05-26.

Shihao Gu, Bryan Kelly, and Dacheng Xiu. 2020a. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Shihao Gu, Bryan Kelly, and Dacheng Xiu. 2020b. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Harbor Framework Team. 2026. Harbor: A framework for evaluating and optimizing agents and models in container environments.

Christophe D Hounwanou and Yae Ulrich Gaba. 2025. Deep generative models for synthetic financial data: Applications to portfolio and risk modeling: Applications of synthetic financial data in portfolio and risk modeling. *arXiv preprint arXiv:2512.21798*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. Financebench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pages 54107–54157.

Wen jun Gu, Yi hao Zhong, Shi zun Li, Chang song Wei, Li ting Dong, Zhuo yue Wang, and Chao Yan. 2024. Predicting stock prices with finbert-lstm: Integrating news sentiment analysis. In *Proceedings of the 2024 8th International Conference on Cloud and Big Data Computing*, ICCBDC '24, page 67–72, New York, NY, USA. Association for Computing Machinery.

Kamil Kashif and Robert Ślepaczuk. 2025. Lstm-arima as a hybrid approach in algorithmic investment strategies. *Knowledge-Based Systems*, 320:113563.

Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. 2010. Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE transactions on neural networks*, 22(3):337–346.

Pulikandala Nithish Kumar, Nneka Umeorah, and Alex Alochukwu. 2024. Dynamic graph neural networks for enhanced volatility prediction in financial markets. *arXiv preprint arXiv:2410.16858*.

Paul H Kupiec and 1 others. 1995. Techniques for verifying the accuracy of risk measurement models.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Kp Subbalakshmi, Jimin Huang, and 1 others. 2025a. Investorbench: A benchmark for financial decision-making tasks with llm-based agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2509–2525.

Kaidi Li, Tianmeng Yang, Min Zhou, Jiahao Meng, Shendi Wang, Yihui Wu, Boshuai Tan, Hu Song, Lujia Pan, Fan Yu, and 1 others. 2024. Sefraud: Graph-based self-explainable fraud detection via interpretative mask learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5329–5338.

Xiangyu Li, Yawen Zeng, Xiaofen Xing, Jin Xu, and Xiangmin Xu. 2025b. Profit mirage: Revisiting information leakage in llm-based financial agents. *Preprint*, arXiv:2510.07920.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2024. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pages 52989–53046.

Xiao-Yang Liu, Ziyi Xia, Jingyang Rui, Jiechao Gao, Hongyang Yang, Ming Zhu, Christina Wang, Zhaoran Wang, and Jian Guo. 2022. Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1835–1849.

Hang Lou, Siran Li, and Hao Ni. 2023. Pcf-gan: generating sequential data via the characteristic function of measures on the path space. *Advances in Neural Information Processing Systems*, 36:39755–39781.

Whitney K Newey and Kenneth D West. 1986. A simple, positive semi-definite, heteroskedasticity and autocorrelationconsistent covariance matrix.

Fan Nie, Junlin Wang, Harper Hua, Federico Bianchi, Yongchan Kwon, Zhenting Qi, Owen Queen, Shang Zhu, and James Zou. 2026. Dsgym: A holistic framework for evaluating and training data science agents. *arXiv preprint arXiv:2601.16344*.

Lingfei Qian, Xueqing Peng, Yan Wang, Vincent Jim Zhang, Huan He, Hanley Smith, Yi Han, Yueru He, Haohang Li, Yupeng Cao, Yangyang Yu, Alejandro Lopez-Lira, Peng Lu, Jian-Yun Nie, Guojun Xiong, Jimin Huang, and Sophia Ananiadou. 2025. When agents trade: Live multi-market trading benchmark for llm agents. *Preprint*, arXiv:2510.11695.

Eghbal Rahimikia, Hao Ni, and Weiguan Wang. 2025. Re (visiting) time series foundation models in finance. *arXiv preprint arXiv:2511.18578*.

Ronald Richman and Salvatore Scognamiglio. 2024. Multiple yield curve modeling and forecasting using deep learning. *ASTIN Bulletin: The Journal of the IAA*, 54(3):463–494.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Shuo Sun, Molei Qin, Wentao Zhang, Haochong Xia, Chuqiao Zong, Jie Ying, Yonggang Xie, Lingxuan Zhao, Xinrun Wang, and Bo An. 2023. Trademaster: a holistic quantitative trading platform empowered by reinforcement learning. *Advances in Neural Information Processing Systems*, 36:59047–59061.

Jiajie Tao, Hao Ni, and Chong Liu. 2024. High rank path development: an approach to learning the filtration of stochastic processes. *Advances in Neural Information Processing Systems*, 37:115309–115350.

Shengkun Wang, Taoran Ji, Jianfeng He, Mariam Almutairi, Dan Wang, Linhan Wang, Min Zhang, and Chang-Tien Lu. 2024. Ama-lstm: Pioneering robust and fair financial audio analysis for stock volatility prediction. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 379–386.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, Yijing Xu, Haoqiang Kang, Ziyan Kuang, Chenhan Yuan, Kailai Yang, Zheheng Luo, Tianlin Zhang, Zhiwei Liu, Guojun Xiong, and 15 others. 2024. Finben: A holistic financial benchmark for large language models. *Advances in Neural Information Processing Systems*, 37:95716–95743.

Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023. Pixiu: A large language model, instruction data and evaluation benchmark for finance. *arXiv preprint arXiv:2306.05443*.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024. Benchmarking benchmark leakage in large language models. *arXiv preprint arXiv:2404.18824*.

Yahoo Finance. 2026. Historical market data. Accessed: 2026-05-20.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. 2020. Qlib: An ai-oriented quantitative investment platform. *arXiv preprint arXiv:2009.11189*.

Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. 2019. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32.

Haofei Yu, Fenghai Li, and Jiaxuan You. 2025a. Livetradebench: Seeking real-world alpha with large language models. *Preprint*, arXiv:2511.03628.

Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Jordan W Suchow, Denghui Zhang, and Khaldoun Khashanah. 2025b. Finmem: A performance-enhanced llm trading agent with layered memory and character design. *IEEE Transactions on Big Data*.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan W Suchow, Zhenyu Cui, Rong Liu, and 1 others. 2024. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *Advances in Neural Information Processing Systems*, 37:137010–137045.

Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, and 1 others. 2024. A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist. In *Proceedings of the 30th acm sigkdd conference on knowledge discovery and data mining*, pages 4314–4325.

Lecheng Zheng, John Birge, Haiyue Wu, Yifang Zhang, and Jingrui He. 2025. Cluster aware graph anomaly detection. In *Proceedings of the ACM on Web Conference 2025*, pages 1771–1782.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

Chuqiao Zong, Chaojie Wang, Molei Qin, Lei Feng, Xinrun Wang, and Bo An. 2024. Macrohft: Memory augmented context-aware reinforcement learning on high frequency trading. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4712–4721.

A Paper Trade Engine

Trading tasks require additional execution and market-data infrastructure that the automated pipeline cannot synthesise from a paper alone. We therefore build a paper trade engine so that the construction of the trading tasks in the task construction pipeline reduces to a configuration change, e.g. asset type, time window, data resolution, etc. The engine is organised around a single executor interface that plugs into the existing financial data providers. The default is an in-process simulator that mirrors portfolio state directly against the latest asset price, as well as applying configurable slippage on market orders and per-trade transaction costs.

Offline trading tasks step over a retrieved historical bar dataset. The engine accepts market, limit, stop, and stop-limit orders under immediate-or-cancel or good-till-cancelled time-in-force, supports batch multi-symbol submissions and explicit order cancellation, and enforces account-level constraints such as cash availability for long entries and a $1\times$ equity cap on short leverage. Orders rejected for constraint violations are surfaced through the next observation step rather than raised as exceptions, letting the agent learn from rejections without crashing the rollout.

Realtime trading tasks act on fresh market data and require a latency-sensitive design. Rather than the naive approach of pulling each observation from a REST endpoint, we subscribe to the data provider's per-trade websocket stream and aggregate the trades into OHLCV bars locally. This decision is motivated by two factors. First, the per-update latency is roughly halved relative to a REST round-trip, and the achievable refresh rate is over an order of magnitude higher in our measurements (Table 3). Second, the websocket path avoids the tight rate limits of the REST API for frequent data fetch. Each symbol's prices and order book are served from a thread-safe market-data buffer that a background daemon thread continuously refreshes from the streaming subscription, so an ordinary gym observation step is a constant-time read from memory. A REST refresh is dispatched only as a fallback when the buffer's staleness exceeds a small per-quantity budget; when one is needed, per-symbol calls are fanned out in parallel through a persistent worker pool so that the wall-clock cost is one round-trip rather than one per symbol.

As an alternative to the in-process simulator, and selectable by configuration alone, the engine can route the same orders to a real broker's paper-trading sandbox and mirror the resulting account state locally. The local mirror keeps portfolio queries, e.g. positions, cash, pending orders, and PnL as constant-time lookups on the agent side, and is kept honest by treating the broker's order-event stream as the primary update path with a periodic REST reconciler as gap-fill; the reconciler takes care not to overwrite confirmed local fills from the lagging account aggregate.

B Training Integrations

This appendix lists the hyperparameters and implementation details of the SFT and RL results in Section 5.2.

Trajectory collection. The 189 SFT trajectories are produced over 21 training tasks by a one-shot agent with a Claude Opus 4.7 backbone at low reasoning effort. The agent emits a single executable per task, which is executed by the verifier executes and returns an aggregated reward. The system prompt forbids training neural networks from scratch and steers the agent toward ridge, lasso, HistGradientBoosting, LightGBM, ARIMA/VAR, exponential smoothing, and GARCH. Trajectories from the five held-out test tasks are dropped.

Supervised fine-tuning. Each Qwen3 backbone is adapted with a LoRA module targeting every linear projection: rank 32 with $\alpha = 64$ for the 1.7B model, rank 16 with $\alpha = 32$ for the 4B and 8B models; dropout 0.05 for 1.7B and 4B, 0.10 for 8B. Training uses a global batch size of 4, maximum sequence length 8192 and AdamW with a warmup-then-cosine schedule. Per-size learning rates are 2×10^{-4}, 1.5×10^{-4}, and 1×10^{-4}; the three models are trained for 184, 138, and 92 optimization steps, corresponding to roughly four, three, and two epochs over the trajectory corpus. This takes < 0.5 GPUh on a single GH200. As a demonstrative result, we did not tune the training hyperparameter for SFT. The trained LoRA weights are then merged into the full-parameter checkpoint that initializes RL.

Reinforcement learning. GRPO runs under SkyRL and vllm with full-parameter updates. Rollouts use temperature 1.0 and top-p 1.0, with sequence length capped at 8192 tokens. We apply a KL penalty of 0.01 against a frozen reference policy together with token-level truncated importance

sampling clipped at 2.0. Training takes roughly $1-4$ GPUh depending on the model size. The RL system prompt is identical to the one used at trajectory collection and SFT.

Reward transform. The verifier's per-task scalar is a lower-is-better weighted aggregate of task-specific metrics. We remap it to a maximization-friendly RL reward by negating and clipping: scoreable trials receive $r = -\min(\text{loss}, 5)$, while failed or unparseable submissions receive a fixed $r = -6$ so that any scoreable trial is strictly preferred over any failure.

C Curated Tasks Catalogue

This appendix catalogues the manually curated and verified forecasting, generation, and fraud detection tasks shipped with OpenFinFym. Each task fixes deterministic train/validation/test splits, a supervised or sample contract that pins the agent's output shape, and a verifier-enforced evaluator whose metric definitions are ported from the source paper.

Dataset provenance. Every curated task is either reproduced from a publicly released source-paper benchmark (Bao et al., 2026; Ang et al., 2025; Tao et al., 2024; Yoon et al., 2019) following their Apache licenses, or rebuilt from open public data feeds—Yahoo Finance (Yahoo Finance, 2026), EIOPA (European Insurance and Occupational Pensions Authority (EIOPA), 2026), and U.S. Treasury par yields (Rahimikia et al., 2025; Gu et al., 2020b; Dinda, 2024; Kumar et al., 2024; Lou et al., 2023; Richman and Scognamiglio, 2024) in a way Consistent with their intended uses; no proprietary or paywalled feed is required at any stage. Synthetic tasks (Ornstein–Uhlenbeck (OU) and rough-volatility data (Lou et al., 2023), Tail-GAN (Cont et al., 2026), and the multivariate fractional Brownian Motion (fBM) dataset (Tao et al., 2024)) regenerate their data based on the model parameters specified in the corresponding source papers. Fraud-detection tasks reuse established node-anomaly benchmarks (Zheng et al., 2025) together with a synthetic graph benchmark (Li et al., 2024). Per-task source citations appear in the Source column of Tables 7, 8, and 9.

C.1 Metric definitions and references

Table 6 defines the metrics abbreviations used throughout this appendix. Unless otherwise stated, scoring metrics are losses for which smaller values are better.

C.2 Curated forecasting tasks

The forecasting catalogue is shown in Table 7. For each task we report the source paper, the data and frequency, the supervised contract (lookback L to horizon H), and the headline metric family. Three entries deviate from the standard time-series-tensor and are flagged in the table: Equity excess return uses a tabular per-row input (42 features per asset-month); Global index volatility is graph-conditioned (correlation + Diebold-Yilmaz spillover graphs); and Yield curve is probabilistic (lower / central / upper interval curves).

Following the method of task generation proposed in (Bao et al., 2026) and its accompanying code, we construct 38 curated forecasting tasks, which cover U.S. equities (organised by sector), U.S. Treasury curve points (3m–30y), G7 / emerging-market / Asia-focused FX pairs, crypto majors and altcoins, and metal / energy / soft commodities, and instantiate 7-8 variants per asset family with different panel selections and lookback-to-horison contracts. All variants are scored with the same metric set as in the source paper (RMSE, MAE, and the Δrisk triplet (ΔSharpe, ΔVaR, ΔES) between predicted and realised paths).

C.3 Curated generative tasks

The generative catalogue is shown in Table 8. The *Sample contract* column pins the shape of the generated tensor: unconditional tasks list a single window shape, while conditional tasks list the context-to-future shape together with the number K of futures generated per context.

C.4 Curated Fraud detection tasks

The fraud detection tasks are shown in Table 9. It contains seven graph-structured tasks drawn from established anomaly-detection and fraud-detection benchmarks. Six tasks are node-level anomaly scoring problems on attributed graphs; one task is a graph-level binary classification on a synthetic motif benchmark.

Symbol	Definition
Point-forecast errors	
MSE	mean squared error.
RMSE	root mean squared error.
MAE	mean absolute error.
MAFE	mean absolute forecast error (per-horizon MAE used in the volatility-graph task).
MAPE	mean absolute percentage error.
SMAPE	symmetric MAPE.
R^2	coefficient of determination.
R^2_{OOS}, R^2_{zero}	out-of-sample R^2 / R^2 vs. a zero forecast.
Statistical tests	
DM	Diebold-Mariano forecast-accuracy test (Diebold and Mariano, 1995; Gu et al., 2020b).
NW t-stat	Newey-West HAC t-statistic (Newey and West, 1986; Rahimikia et al., 2025).
Financial-risk diagnostics	
Sharpe	annualised Sharpe ratio.
VaR_α, ES_α	Value-at-Risk / Expected Shortfall at level α.
ΔSharpe	asset-mean $\lvert\mathrm{Sharpe}_{\mathrm{pred}} - \mathrm{Sharpe}_{\mathrm{true}}\rvert$.
ΔVaR	asset-mean $\lvert\mathrm{VaR}_{\mathrm{pred}} - \mathrm{VaR}_{\mathrm{true}}\rvert$.
ΔES	asset-mean $\lvert\mathrm{ES}_{\mathrm{pred}} - \mathrm{ES}_{\mathrm{true}}\rvert$.
Kupiec test	unconditional coverage test for VaR (Kupiec et al., 1995; Cont et al., 2026).
Interval forecasting	
PICP	prediction-interval coverage probability (target ≈ 0.95) (Khosravi et al., 2010; Richman and Scognamiglio, 2024).
MPIW	mean prediction-interval width (tighter is better, conditional on PICP) (Khosravi et al., 2010; Richman and Scognamiglio, 2024).
Generative-model evaluators	
Disc. score	post-hoc discriminative score: train / test AUC of a classifier separating real from generated paths (Yoon et al., 2019; Lou et al., 2023).
Pred. score	post-hoc predictive score: next-step prediction error of a model trained on generated paths and tested on real ones (Yoon et al., 2019; Esteban et al., 2017; Lou et al., 2023).
Sig-MMD	signature-kernel MMD between path distributions (Chevyrev and Oberhauser, 2022; Lou et al., 2023).
Sig-W_1	signature-based Wasserstein-1 path distance (Tao et al., 2024).
ACF	autocorrelation-function distance, per asset and lag.
ONND	one-nearest-neighbour distance for distributional fidelity (Tao et al., 2024).
Marginal	marginal-distribution histogram distance.
Correlation	L_1 error of cross-asset correlation matrix.
Covariance	Frobenius distance of covariance matrices.
Volatility-graph specific	
DY spillover	Diebold-Yilmaz generalised-FEVD volatility-spillover graph (Diebold and Yilmaz, 2012).

Table 6: Metric abbreviations used throughout the task catalogue.

Task	Source	Data / frequency	Contract ($L \to H$)	Metrics
LOB	(Bao et al., 2026)	14-channel LOB; high-frequency bars	$72 \to 3$ (5 of 14 price channels scored)	RMSE, MAE, Δrisk
Stock-10	(Bao et al., 2026)	10 U.S. equities; daily	$60 \to 3$	RMSE, MAE, Δrisk
Crypto pairs	(Bao et al., 2026)	20 crypto pairs; hourly	$72 \to 3$	RMSE, MAE, Δrisk
Exchange rate	(Ang et al., 2025)	8 FX series; daily	$60 \to 20$	RMSE, MAE, MAPE, SMAPE
Stock OHLCV	(Ang et al., 2025)	5-channel OHLCV; daily	$60 \to 10$ (Close only scored)	RMSE, MAE, MAPE, SMAPE
TSFM	(Rahimikia et al., 2025)	60-name U.S. excess returns; daily	$252 \to 1$, yearly walk-forward	R^2_{OOS}, Sharpe, NW t-stat
NIFTY-50	(Dinda, 2024)	8-factor panel; daily	$30 \to 1$	R^2, RMSE, MAPE
Equity excess return*	(Gu et al., 2020b)	U.S. stock-month panel, 42 features	per-row, yearly walk-forward	R^2_{zero}, DM, long-short Sharpe
Global index volatility†	(Kumar et al., 2024)	8 indices with corr. / DY graphs; daily	$21 \to \{1, 5, 15, 21\}$	MSE, MAFE, MAPE, R^2 per horizon
Yield curve‡	(Richman and Scognamiglio, 2024)	Multi-country EIOPA curves; monthly	$10 \times 150 \to 150$	MSE, MAE, PICP, MPIW
MOSAIC pipeline-generated tasks (7–8 variants per asset family)				
Equity panels	(Bao et al., 2026)	U.S. equity panels by sector (5–9 names); daily	$20 \to 1 - 90 \to 20$	RMSE, MAE, Δrisk
Commodity panels	(Bao et al., 2026)	metals / energy / softs panels (6–17 names); daily	$20 \to 2 - 90 \to 20$	RMSE, MAE, Δrisk
Crypto panels	(Bao et al., 2026)	majors / DeFi / L1 / altcoin panels (6–12 names); hourly	$12 \to 1 - 168 \to 24$	RMSE, MAE, Δrisk
FX panels	(Bao et al., 2026)	G7 / EM / Asia / European FX panels (6–10 pairs); daily	$10 \to 1 - 90 \to 20$	RMSE, MAE, Δrisk
Treasury curve	(Bao et al., 2026)	U.S. Treasury curve segments (3m–30y, 6–10 points); daily	$5 \to 1 - 60 \to 10$	RMSE, MAE, Δrisk

Table 7: Forecasting tasks in OpenFinGym. The top block lists the individually curated tasks; the bottom block summarises additional tasks produced by the MOSAIC pipeline, with each row collapsing several variants that share a data family and metric set but differ in panel composition and lookback-to-horizon contract. Input and output channel counts are equal unless noted. Δrisk denotes the triplet $(\Delta\text{Sharpe}, \Delta\text{VaR}, \Delta\text{ES})$ between predicted and realised paths. Markers: * tabular per-row input (no lookback tensor); † graph-conditioned input (correlation and Diebold-Yilmaz spillover graphs alongside the lookback tensor); ‡ probabilistic output (lower / central / upper interval curves) rather than a point forecast. All metric abbreviations are defined in Table 6.

Task	Source	Data / frequency	Sample contract	Metrics
Crypto	(Bao et al., 2026)	20 pairs; hourly	uncond. 60×20	Marginal, Correlation, ACF, Covariance, Δrisk
LOB	(Bao et al., 2026)	14-channel LOB; HF bars	uncond. 60×14	Marginal, Correlation, ACF, Covariance
Stock	(Bao et al., 2026)	10 equities; daily	uncond. 60×10	Marginal, Correlation, ACF, Covariance, Δrisk
Crypto log-return	(Ang et al., 2025)	10 returns; hourly	uncond. 24×10	VaR, ES
Exchange	(Ang et al., 2025)	8 FX series; daily	uncond. 7×8	Marginal, Correlation, ACF, Covariance
Stock log-return	(Ang et al., 2025)	8 stocks; daily	uncond. 5×8	Marginal, Correlation, ACF, Covariance
OU	(Lou et al., 2023)	Ornstein–Uhlenbeck	uncond. 64×1	Disc., Pred., Sig-MMD
Rough vol	(Lou et al., 2023)	log price + log vol	uncond. 200×2	Disc., Pred., Sig-MMD, ACF
Stock OHLCV	(Lou et al., 2023)	10 stocks OHLCV; daily	uncond. 20×5	Disc., Pred., Sig-MMD
US-5 (cond.)	(Tao et al., 2024)	5 U.S. stocks; daily ret.	cond. $5 \to 5$, $K = 128$	Sig-W_1, ACF, Correlation, Marginal, Am.-put err., ONND
fBM (cond.)	(Tao et al., 2024)	3D fBM, $H = 0.25$	cond. $6 \to 5$, $K = 128$	Sig-W_1, ACF, Correlation, Marginal, ONND
Tail-GAN	(Cont et al., 2026)	5-asset synthetic	uncond. 100×5	rel. VaR / ES, Kupiec, Correlation, ACF
TimeGAN stock	(Yoon et al., 2019)	6-column stock; daily	uncond. 24×6	Disc., Pred.

Table 8: Curated generative tasks. The *Sample contract* column reports the window shape as time-steps $\times$ channels for unconditional tasks; conditional bundles list context length $\to$ future length together with the number K of generated futures emitted per context. Metric symbols are defined in Table 6; Δrisk denotes the triplet $(\Delta\text{Sharpe}, \Delta\text{VaR}, \Delta\text{ES})$.

Dataset	Original task	Description	Evaluation mode	Anomaly rate	Metric
Amazon	(Zheng et al., 2025)	10,224-node e-commerce product graph; 25 feat.; COO adjacency	Node anomaly scoring	6.8%	AUCROC, Recall
BlogCatalog	(Zheng et al., 2025)	5,196-node social network (bloggers); 8,189 feat.; COO adjacency	Node anomaly scoring	5.8%	AUCROC, Recall
YelpChi	(Zheng et al., 2025)	23,831-node review graph (reviews); 32 feat.; COO adjacency	Node anomaly scoring	5.1%	AUCROC, Recall
DBLP	(Zheng et al., 2025)	4,057-node citation/co-author graph; 334 feat.; 2 relation types merged into dense adjacency	Node anomaly scoring	7.0%	AUCROC, Recall
IMDB 5K	(Zheng et al., 2025)	4,780-node movie co-occurrence graph; 1,232 feat.; 2 relation types merged into dense adjacency	Node anomaly scoring	7.0%	AUCROC, Recall
CERT	(Zheng et al., 2025)	1,000-node user behaviour graph; 6 feat. (2 per view: email, logon, file); COO email-activity adjacency	Node anomaly scoring	7.0%	AUCROC, Recall
BA-2motifs	(Li et al., 2024)	1,000 synthetic graphs; 25 nodes $\times$ 10 feat. per graph (constant 0.1); 25 $\times$ 25 adjacency per graph; balanced binary classes (house vs. 5-cycle motif)	Graph binary classification	50.0% (balanced)	AUCROC, Recall

Table 9: Fraud detection tasks in OpenFinGym. The Anomaly rate column reports the fraction of positive (anomalous / fraud-class) labels in the training split.