FinRL-X: An AI-Native Modular Infrastructure for Quantitative Trading

Hongyang Yang*, Boyu Zhang, Yang She, Xinyu Liao, and Xiaoli Zhang

AI4Finance Foundation
All authors contributed equally.

Abstract. We present FinRL-X, a modular and deployment-consistent trading architecture that unifies data processing, strategy construction, backtesting, and broker execution under a weight-centric interface. While existing open-source platforms are often backtesting- or model-centric, they rarely provide system-level consistency between research evaluation and live deployment. FinRL-X addresses this gap through a composable strategy pipeline that integrates stock selection, portfolio allocation, timing, and portfolio-level risk overlays within a unified protocol. The framework supports both rule-based and AI-driven components, including reinforcement learning allocators and LLM-based sentiment signals, without altering downstream execution semantics. FinRL-X provides an extensible foundation for reproducible, end-to-end quantitative trading research and deployment. The official FinRL-X implementation is available at https://github.com/AI4Finance-Foundation/FinRL-Trading.

Keywords: Quantitative Trading Systems · Deep Reinforcement Learning · Financial Portfolio Optimization · Systematic Trading.

1 Introduction

Quantitative trading research has rapidly progressed in recent years, producing increasingly sophisticated signal models, portfolio construction techniques, and learning-based trading agents [19,27,9]. However, many research prototypes remain difficult to reproduce and deploy. Practical trading systems must address broader engineering challenges, including data reliability, interface consistency, execution realism, and system robustness.

Existing open-source frameworks typically address isolated stages of the trading pipeline. Recent LLM-based approaches, such as BloombergGPT [22], FinGPT [24,20,12], and FinRobot [25,28], improve financial text understanding and signal generation but focus on modeling rather than end-to-end system integration. Research platforms such as FinRL [13] and Qlib [26] support rapid RL-based experimentation but lack deployment-consistent architectures. Engineering libraries including Backtrader [3], Zipline [18], `bt` [15], vectorbt [16], Qlib [26], and TradingAgents [23] provide strong backtesting utilities but are typically used

* Corresponding author: `contact@ai4finance.org`

as standalone components. In practice, building a reproducible end-to-end system still requires integrating data ingestion, unified strategy interfaces, and broker connectivity.

The transition from research backtesting to live deployment introduces system-level distortions that are rarely formalized in academic trading frameworks. We categorize these into two primary deployment gaps.

(1) Backtesting-to-paper-trading gap. Offline backtesting environments rely on simplified execution assumptions that diverge from broker-mediated trading environments. Common distortions include oversimplified execution logic (instant fills at bar prices), unrealistic transaction cost modeling, absence of market impact simulation, lack of order book dynamics, survivorship bias, and data feed inconsistencies [4,8]. These issues create a mismatch between simulated reality and brokered reality, leading to inflated performance metrics and unstable behavior once connected to a trading API.

(2) Paper-trading-to-live-trading gap. Even when strategies pass broker-integrated paper trading, additional execution and operational risks emerge in live markets. These include realistic fill uncertainty (latency, partial fills, slippage), liquidity and queue position effects, API behavior differences, infrastructure fragility (server crashes, disconnections), state recovery failures, real capital constraints (margin rules, settlement timing), and extreme systemic events such as flash crashes or faulty code deployments [5,2]. These factors introduce execution distortion and operational risk that are typically absent in academic simulations.

These two gaps reveal that reproducible modeling alone is insufficient. What is required is a deployment-aware system architecture that preserves interface consistency across research, backtesting, broker simulation, and live execution, while explicitly accounting for execution realism and operational resilience.

To address these challenges, we introduce **FinRL-X**, a modular, deployment-oriented trading system built around a unified weight-centric interface. It structures the workflow into four layers: data, strategy, backtesting, and broker-integrated execution, where the strategy layer composes modular decision components. By preserving consistent weight semantics across layers, FinRL-X reduces discrepancies between offline evaluation and live deployment.

Our contributions are summarized as follows:

- **Deployment-aware system architecture.** We formalize and address the backtesting-to-deployment gaps through a layered, weight-centric design that unifies research and execution interfaces.
- **Composable trading abstraction.** We structure trading workflows as modular transformations (selection–allocation–timing–risk), enabling seamless integration of rule-based and learning-based strategies without altering downstream components.
- **Execution-consistent evaluation.** We provide standardized backtesting, broker-integrated execution (e.g., Alpaca [1]), and monitoring mechanisms to ensure consistency between simulation and deployment.
- **Open-source release.** We release FinRL-X as an extensible library with reproducible workflows and runnable examples to facilitate both research and deployment experimentation.

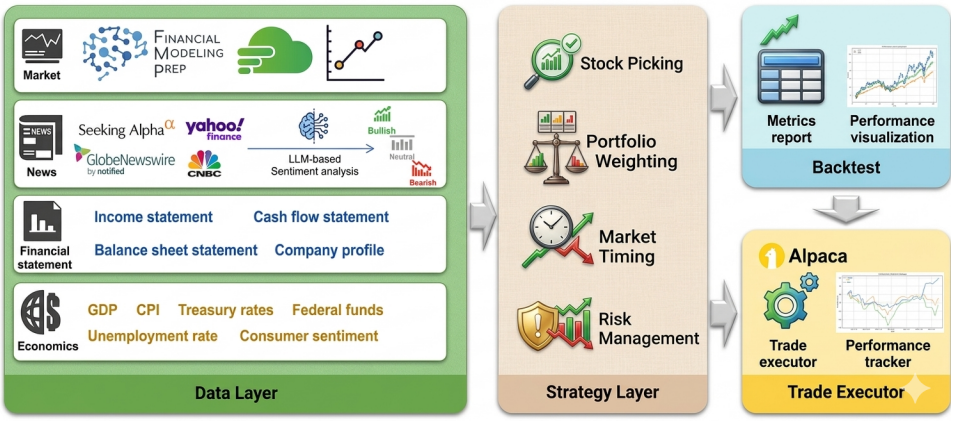

Fig. 1. FinRL-X Framework: A layered, end-to-end trading architecture that unifies data processing, strategy construction, backtesting, and broker-integrated execution within a consistent pipeline, illustrating the workflow from data ingestion to live execution.

2 Related Work

Open-source quantitative trading platforms are typically stage-specific: frameworks such as Zipline [18], Backtrader [3], bt [15], and vectorbt [16] focus on backtesting, while AI-oriented systems including Qlib [26], and TradingAgents [23] emphasize offline ML/RL research. QuantConnect Lean [17] offers broker-integrated trading, but is not structured as a modular research-oriented systems architecture. In contrast, FinRL-X adopts a deployment-aware, weight-centric design that unifies data, strategy, evaluation, and execution within a single interface (Table 1).

Table 1. Comparison of FinRL-X with representative open-source quantitative trading platforms.

Feature	FinRL-X	Qlib	TradingAgents	Zipline/Backtrader	QuantConnect Lean
Primary Orientation	End-to-End System	ML Research	Agent-Based Trading	Backtesting	End-to-End Platform
Broker Integration	Yes	No	No	No	Yes
Deployment-Consistent Interface	Yes	No	No	No	Partial
Reinforcement Learning Support	Yes	Limited	Yes	No	Partial
Modular Strategy Pipeline	Yes	No	No	No	Partial
Portfolio-Level Risk Overlay	Yes	No	No	No	Partial
Open Source License	Apache 2.0	MIT	Apache 2.0	Apache 2.0	Apache 2.0

3 Framework

FinRL-X is a modular, deployment-oriented trading platform that structures the quantitative trading workflow into four layers—data, strategy, backtesting, and execution—as shown in Figure 1. Its design goal is to reduce the engineering overhead of building end-to-end systems by enforcing clear module boundaries and stable interfaces, thereby enabling reproducible offline evaluation and seamless transition to paper or live trading.

3.1 Data Layer

The data layer provides a unified pipeline for ingesting and normalizing structured (market, fundamental, macro) and unstructured (news) inputs, with primary integration to FMP [7] and extensible provider support. All sources are aligned to a shared trading calendar to enable consistent rebalancing and evaluation, while news text is transformed into structured sentiment signals via LLM-based preprocessing for integration into the weight-centric strategy pipeline. Reproducibility is ensured through persistent storage of raw snapshots and processed features, reducing discrepancies between offline experiments and deployment.

3.2 Strategy Layer

The strategy layer adopts a *weight-centric* architectural principle. In FinRL-X, the target portfolio weight vector $w_t \in \mathbb{R}^n$ is treated as the sole interface contract between strategy logic and downstream evaluation or execution modules. Rather than emitting trading signals, position deltas, or broker-specific orders, every strategy component produces a target allocation vector that specifies the desired capital fraction assigned to each asset at time t.

Formally, let $\mathcal{U}_t$ denote the tradable asset universe at time t. The strategy layer defines a sequence of contract-preserving transformations that map time-aligned inputs into a feasible portfolio weight vector:

$$w_t = \mathcal{R}_t\big(\mathcal{T}_t(\mathcal{A}_t(\mathcal{S}_t(\mathcal{X}_{\le t})))\big),$$

where $\mathcal{S}$ denotes stock selection, $\mathcal{A}$ portfolio allocation, $\mathcal{T}$ timing adjustment, and $\mathcal{R}$ portfolio-level risk overlay.

This weight-centric abstraction provides three system-level advantages: (i) it decouples strategy construction from broker implementation details; (ii) it enables composable transformations across heterogeneous rule-based and learning-based modules; and (iii) it ensures deployment consistency, as both backtesting and live execution consume the same weight representation.

Algorithm 1 Weight-Centric Trading Pipeline

Require: Data streams $\mathcal{D}, \mathcal{F}, \mathcal{T}, \mathcal{R}$; rebalancing times $\{t_1, \dots, t_n\}$
Initialize portfolio value P_0
for each t **do**
  $\mathcal{C}_t \leftarrow \textsc{Select}(\mathcal{F}_{\le t}, \mathcal{U}_t)$
  $w_t^{base} \leftarrow \textsc{Allocate}(\mathcal{C}_t)$
  $w_t^{timing} \leftarrow \textsc{TimeAdjust}(w_t^{base}, \mathcal{T}_{\le t})$
  $w_t \leftarrow \textsc{RiskOverlay}(w_t^{timing}, \mathcal{R}_{\le t})$
  Observe realized returns r_t
  $P_t \leftarrow P_{t-1}(1 + w_t^\top r_t)$
end for
return P_n

Modular Components. The pipeline consists of four contract-preserving transformations. **Stock Selection** constructs a candidate set $\mathcal{C}_t \subseteq \mathcal{U}_t$ using fundamentals or learned scoring models under strict no-lookahead semantics. **Portfolio Allocation** maps $\mathcal{C}_t$ to feasible base weights w_t^{base} (e.g., equal-weight, mean–variance, minimum-variance, or DRL-based policies) under consistent normalization and leverage constraints. **Timing Adjustment** transforms w_t^{base} into w_t^{timing} using trend-based or learning-based signals without altering the weight interface. **Risk Overlay** applies volatility-aware exposure scaling (e.g., VIX-based) at the portfolio level, adjusting aggregate exposure while preserving relative allocations to produce final executable weights w_t.

3.3 Backtesting and Execution Layer

FinRL-X reuses a unified weight interface for both offline backtesting (via *bt* [15]) and live broker execution, ensuring consistent portfolio semantics across evaluation and deployment. The executor converts target weights into orders with configurable safeguards and logs realized allocations for post-trade consistency checks.

3.4 Deployment-Aware Design

Beyond modeling accuracy, quantitative trading systems face systematic distortions when transitioning from research backtesting to live deployment. Let $\mathcal{S}_{research}$, $\mathcal{S}_{paper}$, and $\mathcal{S}_{live}$ denote system behavior under offline simulation, broker-integrated paper trading, and live execution, respectively. In practice,

$$\mathcal{S}_{research} \neq \mathcal{S}_{paper} \neq \mathcal{S}_{live},$$

due to execution simplifications, infrastructure instability, and operational constraints.

FinRL-X narrows these deployment gaps architecturally. It reduces the backtesting-to-paper gap by enforcing consistent execution semantics across environments: strategies output broker-agnostic weight vectors, while simulation incorporates transaction costs, slippage modeling, and event-driven order handling aligned with broker APIs. Data ingestion follows a unified schema to ensure consistency between historical replay and live feeds, minimizing discrepancies caused by data formatting or synchronization differences.

To mitigate the paper-to-live gap, FinRL-X introduces deployment-oriented safeguards at the execution layer. These include state persistence for crash recovery, structured logging for post-trade reconciliation, and fault-tolerant broker interaction mechanisms that handle API interruptions and execution anomalies. Importantly, these mechanisms operate independently of strategy logic, preserving modularity while improving operational resilience.

By maintaining a unified weight interface across research, simulation, and execution layers, and by explicitly engineering for execution realism and robustness, FinRL-X reduces behavioral divergence between offline evaluation and live deployment.

4 Evaluation

We evaluate FinRL-X from a system-level perspective, emphasizing reproducibility, modular composability, and deployment consistency in addition to return performance. Experiments compare allocation paradigms, timing mechanisms, and risk overlays under a unified backtesting protocol with standardized metrics.

4.1 Experimental Setup and Metrics

Experiments are conducted on liquid U.S. equities and ETFs, with SPY and QQQ as benchmark indices. The historical backtesting horizon spans January 7, 2018 to October 24, 2025 under proportional transaction costs of 10 bps per side. A broker-integrated paper-trading evaluation (e.g., Alpaca [1]) is conducted from October 26, 2025 to March 12, 2026 to assess deployment behavior. All decisions at time t rely strictly on information available up to t, and learning-based models are evaluated using rolling out-of-sample validation.

Evaluation metrics. Return (cumulative, annualized), risk (volatility, maximum drawdown), risk-adjusted performance (Sharpe, Sortino, Calmar), and deployability (portfolio turnover).

4.2 Baselines

We compare FinRL-X with representative baselines from four categories:

- **Classical allocation.** Equal-weight serves as a reference, alongside Mean–Variance [14] and Minimum-Variance [6] methods.
- **Learning-based allocation.** DRL allocators generate portfolio weights via sequential decision-making [10], with rolling out-of-sample validation.
- **Timing strategies.** Trend-following methods such as KAMA [11] provide rule-based exposure control.
- **Risk overlays.** A VIX-based mechanism [21] scales portfolio exposure as a post-allocation overlay.

4.3 Portfolio Performance and Ablation Analysis

FinRL-X is designed to support composable strategy modules under identical execution semantics. We validate this modularity through controlled ablations that isolate timing and overlay effects while keeping the remaining pipeline unchanged.

Timing ablation (DRL). Figure 2 compares DRL allocation with and without timing against the SPY benchmark. The timing-enhanced variant achieves higher cumulative returns and lower drawdowns, demonstrating that timing can be integrated without modifying backtest or execution interfaces.

Cross-strategy ablation. Table 2 reports standardized return and risk metrics across representative strategies. Across MeanVar, MinVar, Equal, and DRL configurations, timing-enabled variants consistently improve risk-adjusted performance and moderate drawdown relative to their base counterparts.

Table 2. Performance and risk metrics across benchmarks

Strategy	Cum. Return	Ann. Return	Ann. Vol	Sharpe	Sortino	Calmar	Max DD	DD Duration
SPY	2.63	0.14	0.17	0.84	0.76	0.60	-0.23	23
QQQ	3.61	0.19	0.20	0.95	0.93	0.60	-0.33	23
KAMA	2.40	0.12	0.21	0.57	0.60	0.43	-0.29	23
MeanVar (No Timing)	2.19	0.11	0.22	0.53	0.56	0.37	-0.31	41
MeanVar (With Timing)	2.59	0.14	0.19	0.74	0.76	0.52	-0.27	38
MinVar (No Timing)	2.49	0.13	0.21	0.63	0.68	0.47	-0.28	25
MinVar (With Timing)	2.97	0.16	0.18	0.90	0.94	0.60	-0.27	23
Equal (No Timing)	2.11	0.11	0.20	0.54	0.53	0.40	-0.27	27
Equal (With Timing)	2.64	0.14	0.16	0.87	0.85	0.62	-0.23	23
DRL (No Timing)	2.33	0.12	0.22	0.55	0.54	0.40	-0.31	18
DRL (With Timing)	3.03	0.17	0.18	0.89	0.87	0.61	-0.27	18

4.4 Use Case Demonstrations

To illustrate end-to-end system flexibility, we present representative use cases that isolate the contribution of individual components while keeping the same workflow (data → strategy → backtest → optional execution) unchanged.

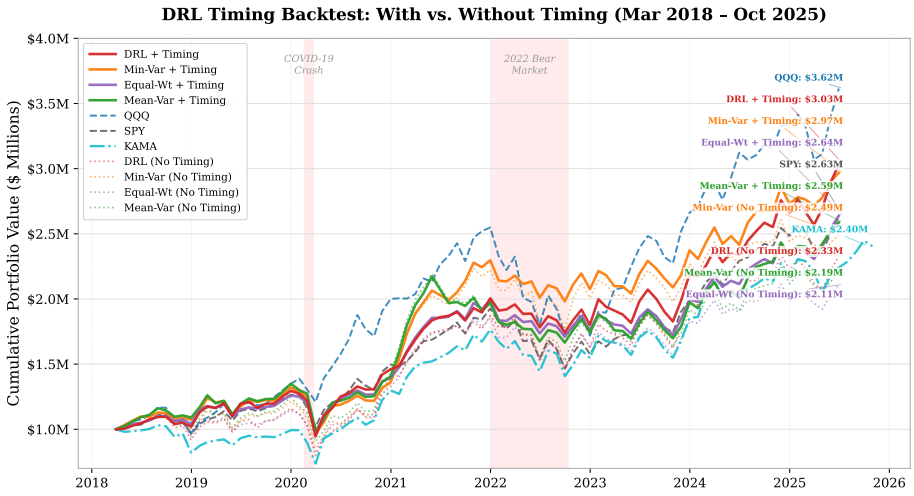

Fig. 2. Ablation study of DRL-based allocation with and without timing adjustment. Incorporating the timing module improves cumulative performance and moderates drawdown relative to both the base DRL strategy and the SPY benchmark.

Use Case 1: Portfolio Allocation Paradigms We evaluate heterogeneous allocation mechanisms under a unified weight-centric interface, including DRL-based, optimization-based (Mean–Variance, Minimum-Variance), equal-weight, and signal-driven strategies (e.g., KAMA). By enforcing identical data and execution semantics, FinRL-X enables fair comparison across paradigms without architectural changes. Results are reported in Figure 2 and Table 2.

Table 3. Performance comparison of representative use cases and benchmark indices (2018–2025).

Metric	Rolling Strategy	Adaptive Rotation	QQQ	SPY
Cumulative Return	5.98	4.80	4.02	2.80
Annualized Return (%)	25.85	22.32	19.56	14.14
Annualized Volatility (%)	27.85	20.30	24.20	19.61
Sharpe Ratio	0.93	1.10	0.81	0.72
Maximum Drawdown (%)	-38.95	-21.46	-35.12	-33.72
Calmar Ratio	0.66	1.04	0.56	0.42
Win Rate (%)	54.36	54.77	56.25	55.28

Use Case 2: Rolling Stock Selection This use case tests the rolling stock selection module, where the universe is updated upon new quarterly financial reports. We use all component stocks of the NASDAQ 100 index as candidates and select the top 25% to construct a portfolio with DRL-based allocation. Figure 3 (line **Rolling Selection**) shows cumulative returns. Table 3 (Rolling Strategy) reports performance relative to SPY and QQQ.

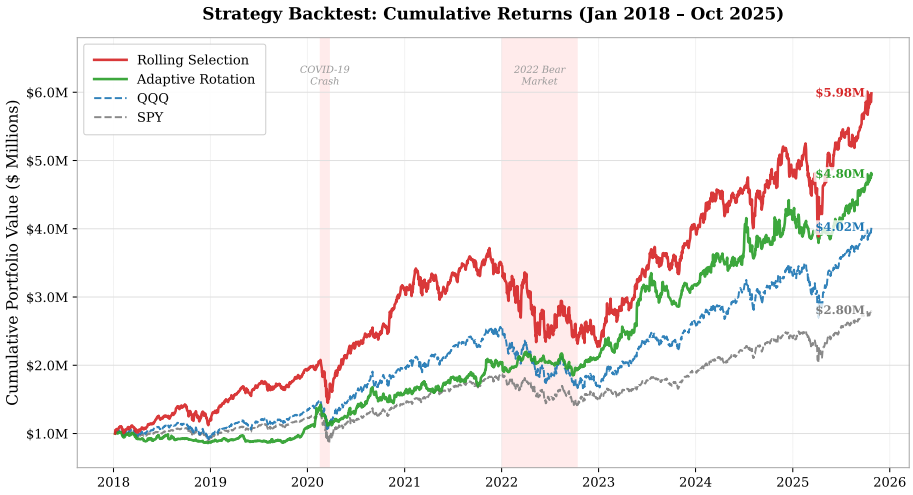

Fig. 3. Backtest performance comparison across representative strategy configurations under the unified weight-centric protocol (January 7, 2018 – October 24, 2025). Results illustrate cumulative portfolio trajectories relative to benchmark references.

Use Case 3: Adaptive Multi-Asset Rotation This use case presents an adaptive multi-asset rotation strategy designed to achieve stable excess returns relative to QQQ across regimes. Assets are grouped into Growth, Real Assets, and Defensive buckets, with at most two active groups selected per weekly rebalance. Group selection is driven by Information Ratio relative to QQQ, while intra-group

allocation uses residual momentum with robust exception handling. Regime indicators are used for risk gating rather than alpha generation.

Figure 3 (line **Adaptive Rotation**) shows sustained outperformance with improved drawdown control across cycles. Table 3 (Adaptive Rotation) reports risk-adjusted metrics and drawdown improvements relative to SPY and QQQ.

4.5 Paper Trading and Deployment Validation

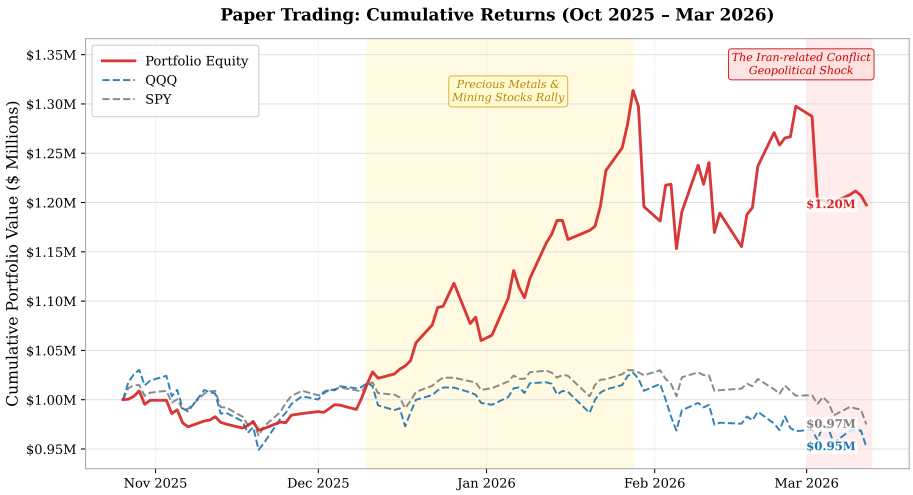

Fig. 4. Paper trading performance relative to benchmark indices (October 26, 2025 – March 12, 2026), demonstrating deployment-consistent execution under daily rebalancing.

Paper trading as deployment-consistency validation. To bridge offline evaluation and live deployment, we execute an ensemble strategy (Rolling Selection + Adaptive Rotation) in an Alpaca paper trading environment from October 2025 to March 2026 under daily rebalancing. Despite the limited horizon, results demonstrate stable deployment and consistent execution under real broker conditions, with low order rejection, minimal guardrail triggers, and small tracking error between target and realized allocations, indicating high execution fidelity. Figure 4 and Table 4 report the equity curve and performance statistics.

Paper Trading Analysis To evaluate deployment consistency beyond backtesting, we conduct a six-month paper trading experiment (Oct 26, 2025–Mar 12, 2026) under daily rebalancing. As shown in Figure 4, the strategy achieves a total return of $+$**19.76%**, outperforming SPY and QQQ. Despite the limited horizon, the results demonstrate stable execution and validate the end-to-end pipeline under live-like conditions.

Table 4. Performance comparison between paper trading and benchmark indices (Oct 26, 2025–Mar 12, 2026, Daily Turnover).

Metric	Strategy	SPY	QQQ
Cumulative Return	1.20	0.97	0.95
Total Return (%)	19.76	-2.51	-4.79
Annualized Return (%)	62.16	-6.60	-12.32
Annualized Volatility (%)	31.75	11.96	16.79
Sharpe Ratio	1.96	-0.55	-0.73
Maximum Drawdown (%)	-12.22	-5.35	-7.88
Calmar Ratio	5.09	-1.23	-1.56
Win Rate (%)	64.89	52.13	54.02

Allocation Trajectory Under Unified Execution Interface Figure 5 shows time-varying portfolio weights during paper trading, highlighting dynamic allocations that pass unchanged through the unified weight-based execution interface. The shifts reflect regime-aware adjustments driven by momentum and risk signals. Notably, no architectural changes are required from backtesting to broker execution, demonstrating deployment consistency.

Stress Event as Risk-Module Validation The paper-trading period includes an adverse episode with a peak-to-trough drawdown of **12.2%** following an extreme move in a leveraged instrument. We treat this as a stress case, revealing nonlinear leverage risk and motivating safeguards such as volatility-aware scaling and exposure caps. The unified weight interface enables consistent post-trade attribution without modifying strategy logic, reinforcing modularity and diagnosability.

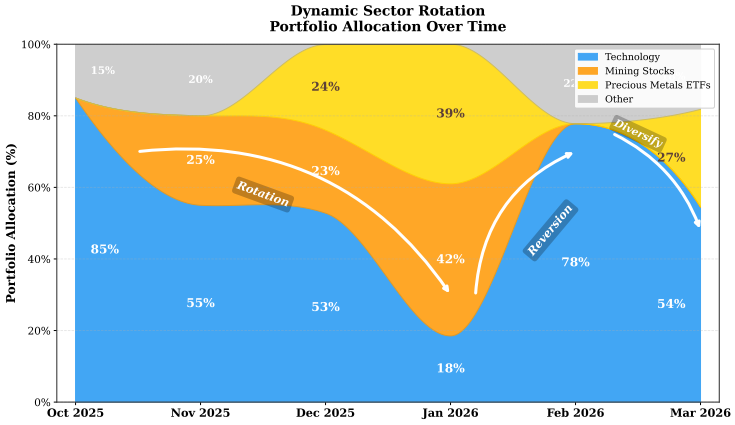

Fig. 5. Portfolio allocation trajectory under the unified weight-based execution framework during paper trading, showing time-varying exposures and directly executable modular allocations.

5 Conclusions

FinRL-X is a deployment-consistent, modular trading system that unifies data processing, strategy, evaluation, and broker execution. Its weight-centric interface enforces consistent decision semantics across research, backtesting, and live trading, reducing discrepancies between offline evaluation and deployment. The framework supports heterogeneous strategies while preserving reproducibility and composability, and empirical results, including paper trading, demonstrate stable execution under realistic conditions. Future work will extend FinRL-X to broader asset classes and more execution-aware strategies for scalable real-world deployment.

Acknowledgements

This work is developed and maintained under the AI4Finance Foundation open-source ecosystem. The AI4Finance Foundation[1] was founded in 2017 at Columbia University. Some authors contributed to this work while also enrolled as students at Columbia University. FinRL and the FinRL logo are trademarks of FinRL LLC and are used with permission.

References

1. Alpaca. Alpaca api documentation: Paper trading. `https://docs.alpaca.markets/docs/paper-trading`, 2025.
2. Alpaca Markets. Paper trading vs. live trading: A data-backed guide on when to start trading real money. `https://alpaca.markets/learn/paper-trading-vs-live-trading-a-data-backed-guide-on-when-to-start-trading-real-money`, 2025. Accessed: 2026-02.
3. Backtrader. Backtrader: A feature-rich python framework for backtesting and trading. `https://www.backtrader.com/`, 2015.
4. David Bailey, Jonathan Borwein, Marcos Lopez de Prado, and Qiji Jim Zhu. The probability of backtest overfitting. *The Journal of Computational Finance*, 20(4):39–69, 2017.
5. Álvaro Cartea, Sebastian Jaimungal, and José Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.
6. Roger Clarke, Harindra de Silva, and Steven Thorley. Minimum-variance portfolio composition. *The Journal of Portfolio Management*, 33(2):10–24, 2006.
7. Financial Modeling Prep. Financial modeling prep. `https://site.financialmodelingprep.com/`, 2026. Accessed: 2026-01-04.
8. Yash Ganar. Why backtesting environments differ from live markets: Technical factors explained. `https://algobulls.com/blog/algo-trading/backtesting-technical-factor`, 2026. Accessed: 2026-02.
9. Xuewen Han, Neng Wang, Shangkun Che, Hongyang Yang, Kunpeng Zhang, and Sean Xin Xu. Enhancing investment analysis: Optimizing ai-agent collaboration in financial research. In *ICAIF 2024: Proceedings of the 5th ACM International Conference on AI in Finance*, pages 538–546, 2024.

[1] `https://ai4finance.org`

10. Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for financial portfolio management. *arXiv preprint arXiv:1706.10059*, 2017.
11. Perry J. Kaufman. *Trading Systems and Methods*. Wiley, 1998.
12. Yixuan Liang, Yuncong Liu, Neng Wang, Hongyang Yang, Boyu Zhang, and Christina Dan Wang. Fingpt: enhancing sentiment-based stock movement prediction with dissemination-aware and context-enriched llms. *AAAI 2025 Workshop GoodData*, 2025.
13. Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*, 2020.
14. Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.
15. Philippe Morissette. bt: Flexible backtesting for python. `https://github.com/pmorissette/bt`, 2014.
16. Polakowo. vectorbt: Portfolio optimization and backtesting on pandas/numpy. `https://github.com/polakowo/vectorbt`, 2020.
17. QuantConnect. Lean algorithmic trading engine. `https://github.com/QuantConnect/Lean`, 2024.
18. Quantopian. Zipline: A pythonic algorithmic trading library. `https://github.com/quantopian/zipline`, 2014.
19. Santosh Kumar Sahu, Anil Mokhade, and Neeraj Dhanraj Bokde. An overview of machine learning, deep learning, and reinforcement learning-based techniques in quantitative finance: recent progress and challenges. *Applied Sciences*, 13(3):1956, 2023.
20. Neng Wang, Hongyang Yang, and Christina Dan Wang. Fingpt: Instruction tuning benchmark for open-source large language models in financial datasets. *NeurIPS Workshop on Instruction Tuning and Instruction Following*, 2023.
21. Robert E Whaley. The investor fear gauge. *The Journal of Portfolio Management*, 26(3):12–17, 2000.
22. Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.
23. Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. Tradingagents: Multi-agents llm financial trading framework. *arXiv preprint arXiv:2412.20138*, 2024.
24. Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*, 2023. First official FinGPT paper; FinLLM Workshop at IJCAI 2023.
25. Hongyang Yang, Boyu Zhang, Neng Wang, Cheng Guo, Xiaoli Zhang, Likun Lin, Junlin Wang, Tianyu Zhou, Mao Guan, Runjia Zhang, et al. Finrobot: An open-source ai agent platform for financial applications using large language models. *arXiv preprint arXiv:2405.14767*, 2024.
26. Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. Qlib: An ai-oriented quantitative investment platform, 2020. arXiv preprint arXiv:2009.11189.
27. Boyu Zhang, Hongyang Yang, tianyu Zhou, Ali Babar, and Xiao-Yang Liu. Enhancing financial sentiment analysis via retrieval augmented large language models. *ACM International Conference on AI in Finance (ICAIF)*, 2023.
28. Tianyu Zhou, Pinqiao Wang, Yilin Wu, and Hongyang Yang. Finrobot: AI agent for equity research and valuation with large language models. In *ICAIF 2024: The 1st Workshop on Large Language Models and Generative AI for Finance*, 2024.