

Parallel Market Environments for FinRL Contests

Keyi Wang
Columbia University
New York, NY, USA
kw2914@columbia.edu

Kairong Xiao
Business School, Columbia University
New York, NY, USA
kx2139@columbia.edu

Xiao-Yang Liu Yanglet
Columbia University
New York, NY, USA
XL2427@columbia.edu

Abstract—Financial reinforcement learning has attracted lots of attention recently. From 2023 to 2025, we have organized three FinRL Contests featuring different financial tasks. Large language models have a strong capability to process financial documents. By integrating LLM-generated signals into the state, trading agents can take smarter actions based on both structured market data and unstructured financial documents. In this paper, we summarize the parallel market environments for tasks used in FinRL Contests 2023-2025. To address the sampling bottleneck during training, we introduce GPU-optimized parallel market environments to address the sampling bottleneck. In particular, two new tasks incorporate LLM-generated signals and all tasks support massively parallel simulation. Contestants have used these market environments to train robust and powerful trading agents for both stock and cryptocurrency trading tasks.

Index Terms—FinRL trading agents, large language model, market environment, massively parallel simulation.

I. INTRODUCTION

Financial reinforcement learning (FinRL) [1]–[3] applies reinforcement learning algorithms to financial tasks, such as order execution, portfolio management, high-frequency trading, option pricing and hedging, and market making [4]–[6].

High-quality market environments are crucial to develop powerful trading agents. Financial data, featuring non-stationary and low signal-to-noise ratio, is processed into standard gym-style [7] market environments. A financial task is modeled as a Markov Decision Process (MDP) by defining the state, action, and reward. FinRL-Meta [8], [9] provided hundreds of market environments through an automated data curation pipeline.

In this paper, we summarize the parallel market environments used in FinRL Contests 2023-2025. We select several stable market environments from [1]–[3], [8], [9] and provide an ensemble learning approach [10] with GPU-optimized parallel environments. Moreover, we integrate large language models (LLMs), based on FinRL-DeepSeek [11] and FinRL-lama [12], by allowing the trading agent to leverage signals extracted from unstructured financial documents [11], [12]. We reported an improved sampling speed of $1,650\times$ when using 2,048 parallel environments. Contestants have used these environments to develop trading agents for both stock and cryptocurrency trading tasks and released their codes to the open-source community.

The remainder of this paper is organized as follows. Section II describes the challenges. Section III describes massively parallel environments. We conclude this paper in Section IV.

II. CHALLENGES

The tasks in FinRL Contests 2023-2025 encourage contestants to address key challenges:

- **Policy instability.** The performance of RL policies is sensitive to hyperparameters, market noise, and random seeds. Policy instability can also come from value function approximation errors. It is challenging to develop reliable and robust trading strategies.
- **Sampling bottleneck.** High-quality samples are crucial for training a powerful trading agent in highly dynamic and complex financial markets. Extensive sampling is necessary for low-variance gradient estimation and stable policy updates. In a traditional training process, data is sampled and stored into a replay buffer on CPU memory and then transferred to a GPU for policy updates. Frequent data transfers between CPU and GPU are a bottleneck too.
- **Signals from financial documents.** Unstructured financial texts, such as news and SEC filings, contain valuable information about market sentiment, risk, and events. LLMs have been used to generate signals from financial documents for informed decision-making [13]. Integrating LLM-generated signals into FinRL requires a market environment to support multimodal input and align market and textual data.

III. PARALLEL MARKET ENVIRONMENTS

We summarize trading tasks of FinRL Contests 2023-2025 in Fig. 1, including daily stock trading with OHLCV data and second-level cryptocurrency trading with LOB data. To address the sampling bottleneck of the training stage, we develop massively parallel market environments on GPUs. We encourage contestants to use ensemble methods to mitigate policy instability and improve models' robustness. In addition, we include two new tasks that integrate LLM-generated signals, i.e., FinRL-DeepSeek [11] and FinRLlama [12], respectively.

A. Task Description

We formulate stock and cryptocurrency trading tasks as Markov Decision Processes (MDPs) [9].

- **State** $\mathbf{s}_t = [b_t, \mathbf{p}_t, \mathbf{h}_t, \mathbf{f}_t] \in \mathbb{R}^{K(I+2)+1}$ represents market conditions at time t . $b_t \in \mathbb{R}_+$ is the account balance. $\mathbf{p}_t \in \mathbb{R}_+^K$ is the prices of stocks or cryptocurrencies, where K is the number of assets. $\mathbf{h}_t \in \mathbb{R}_+^K$ is the holding positions. $\mathbf{f}_t \in \mathbb{R}^{KI}$ is the feature vector, where there are I features for each asset, such as Moving Average Convergence Divergence (MACD) and sentiment score derived from the news.

Key Components	Attributes	FinRL Contest 2023	FinRL Contest 2024		FinRL Contest 2025	
		Task 1 Data-Centric Stock Trading	Task 1 Crypto Trading with Ensemble Learning	Task 2 LLM-Engineered Signals with RLMF	Task 1 FinRL-DeepSeek for Stock Trading	Task 2 FinRL-AlphaSeek for Crypto Trading
State	Balance; Shares	✓	✓	✓	✓	✓
	Open/high/low/close/volume (OHLCV) data	✓		✓	✓	
	Limit order book (LOB) data		✓			✓
	Technical indicators	✓	✓	✓	✓	✓
	Fundamental indicators					
	Social data; Sentiment data			✓	✓	
	Smart beta indexes, etc.					
Action	Buy/Sell/Hold	✓	✓		✓	✓
	Short/Long					
	Portfolio weights					
	Sentiment score			✓		
Rewards	Change of portfolio value	✓	✓		✓	✓
	Portfolio log-return					
	Sharpe ratio				✓	
	Penalize for high risk					
	Market feedback			✓		

Fig. 1. Tasks in FinRL Contests 2023-2025.

- **Action** $\mathbf{a}_t \in \mathbb{R}^K$ represents trading actions at time t , such that $\mathbf{h}_{t+1} = \mathbf{h}_t + \mathbf{a}_t$. An entry $a_t^i > 0, i = 1, \dots, K$ indicates buying a_t^i shares of assets i , while $a_t^i < 0$ indicates selling and $a_t^i = 0$ indicates holding the current position.
- **Reward function** $R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ is an incentive signal to motivate the trading agent to perform action $\mathbf{a}_t$ at state $\mathbf{s}_t$. The reward can be defined as the change in the total asset value, i.e., $R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) = v_{t+1} - v_t$, where $v_t = \mathbf{b}_t + \mathbf{p}_t^T \mathbf{h}_t$ is the total asset values at time t . The reward can also be penalized for high-risk signals.
- **Policy** $\pi(\cdot|\mathbf{s}_t)$ is a probability distribution over actions at state $\mathbf{s}_t$. It determines the likelihood of each possible trading action given the current market conditions.

LLM-generated signals. As in FinRL-DeepSeek [11], LLMs generate sentiment and risk scores from financial news:

- **Sentiment score.** DeepSeek-V3 [14] assigns a sentiment score u of 1 to 5 according to the news, with 1 for strongly negative and 5 highly positive. It is included in the feature vector $\mathbf{f}_t$ and is used to adjust actions via the sentiment factor $l_t^i = 1 + 0.05(u - 3)\text{sign}(a_t^i)$. The factor is close to 1 for the stability of the algorithm. The adjusted action is $a_t^{i'} = l_t^i a_t^i$, which is amplified under positive sentiment and dampened under negative sentiment.
- **Risk level.** DeepSeek-V3 [14] assigns a risk level q of 1 to 5 from the news, with 1 for low risk and 5 high risk. It is included in the feature vector $\mathbf{f}_t$ and is used to penalize rewards through the risk factor $f_t^i = 1 + 0.05(q_t^i - 3)$. The aggregated risk factor is $M_t = \sum_i^K w_t^i f_t^i$, where w_t^i is the portfolio weight of the stock i and $\sum w_t^i = 1$. $M_t > 1$ penalizes the reward for high risk; $M_t < 1$ increases the reward for low risk.

Market constraints. We incorporate near-real constraints:

- **Transaction costs.** We set a cost of 0.1% for each action {buy, sell}, accounting for commission and slippage.
- **Market volatility.** The turbulence index and VIX index are risk indicators. A large value signals heightened volatility from factors like investor fear and increased uncertainty, while a small value signals increased stability in markets.

B. Massively Parallel Environment on GPUs

Objective Function for Training. The goal of trading tasks is to learn a policy π_θ with parameter θ to maximize the expected return:

$$J(\theta) = \int_{\tau} P(\tau|\pi_\theta) R(\tau) = \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)], \quad (1)$$

where τ is a trajectory, $P(\tau|\pi_\theta)$ is the probability of τ following policy π_θ , $R(\tau)$ is the (discounted) cumulative return along τ . The gradient of $J(\theta)$ with respect to θ can be estimated using the Monte Carlo method [15]:

$$\begin{aligned} \nabla J(\theta) &= \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^T R(\tau) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right] \\ &\approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T R(\tau^{(i)}) \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}). \end{aligned} \quad (2)$$

Massively Parallel Environment. Stable policy updates require a low-variance gradient estimate $\nabla J(\theta)$ from a large N independent trading trajectories τ , where there is high parallelism. PyTorch’s `vmap` can map operations over some dimension (i.e., N) onto parallel GPU cores, exploiting the parallelism of (2). Therefore, we construct vectorized environments and provide GPU optimization via `vmap`. As shown in Fig. 2, a vectorized environment manages parallel sub-environments (SubEnvs). The operations in a SubEnv include

- **reset:** $s_t \rightarrow s_0$, resets the environment to its initial state.
- **step:** $(s_t, a_t) \rightarrow s_{t+1}$, executes a_t and updates s_t to s_{t+1} .
- **reward:** $(s_t, a_t, s_{t+1}) \rightarrow r_t$, computes the reward.

Using `vmap`, the step and reward functions are vectorized to operate in massively parallel environments. For example, the reward function, vectorized by `vmap`, computes the reward on (s_t, a_t, s_{t+1}) for each SubEnv simultaneously. This computation is dispatched to available GPU cores, each responsible for calculating its assigned data.

Data samples are stored as tensors in GPU memory. They have the shape $N \times T \times D$:

- N is the number of parallel SubEnvs.

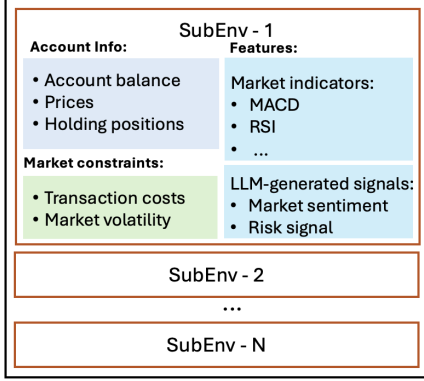


Fig. 2. Vectorized environment.

- T is the number of steps in a trajectory.
- D is the dimension as in Section III-A, where $D = K(I + 2) + 1$ for state, $D = K$ for action, and $D = 1$ for reward. The tensors for states ($\mathbf{s} \in \mathbb{R}^{K(I+2)+1}$), actions ($\mathbf{a} \in \mathbb{R}^K$), and rewards ($r \in \mathbb{R}$) are as follows:

$$\begin{bmatrix} \mathbf{s}_1^1 & \mathbf{s}_2^1 & \cdots & \mathbf{s}_T^1 \\ \mathbf{s}_1^2 & \mathbf{s}_2^2 & \cdots & \mathbf{s}_T^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{s}_1^N & \mathbf{s}_2^N & \cdots & \mathbf{s}_T^N \end{bmatrix}, \begin{bmatrix} \mathbf{a}_1^1 & \mathbf{a}_2^1 & \cdots & \mathbf{a}_T^1 \\ \mathbf{a}_1^2 & \mathbf{a}_2^2 & \cdots & \mathbf{a}_T^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_1^N & \mathbf{a}_2^N & \cdots & \mathbf{a}_T^N \end{bmatrix}, \begin{bmatrix} r_1^1 & r_2^1 & \cdots & r_T^1 \\ r_1^2 & r_2^2 & \cdots & r_T^2 \\ \vdots & \vdots & \ddots & \vdots \\ r_1^N & r_2^N & \cdots & r_T^N \end{bmatrix}.$$

Storing data samples as tensors in GPU memory bypasses the CPU-GPU bandwidth bottleneck.

Improved Sampling Speed with Massively Parallel Environments on GPU. We evaluated the sampling speed measured in samples per second using vectorized environments for stock trading. We used the PPO agent and the OHLCV data of 30 constituent stocks in the Dow Jones index, from 2020-01-01 to 2022-01-01. NVIDIA A100 GPU is used. The numbers of parallel environments vary from 1, 2, 4, ..., to 2,048. As shown in Fig. 3, the average sampling speed with 2,048 parallel environments is 227,212.54 samples per second. The sampling speed is improved by $1,649.93\times$ compared with a single environment. The sampling speed scales approximately linearly with the number of parallel environments. The results show the effectiveness of massively parallel simulation in improving sampling speed in FinRL Contest tasks.

IV. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we summarized the massively parallel market environments used in FinRL Contests 2023-2025. Two new environments incorporate LLM-generated signals, allowing the FinRL trading agents to leverage unstructured financial documents. The vectorized environments support massively parallel simulation on GPUs, addressing the sampling bottleneck.

For future work, we will continue exploring and integrating LLM-generated signals from multimodal financial data, such as SEC filings, earning conference calls, alternative data, etc.

ACKNOWLEDGEMENT

Keyi Wang and Xiao-Yang Liu Yanglet acknowledge the support from Columbia's SIRS and STAR Program, The Tang

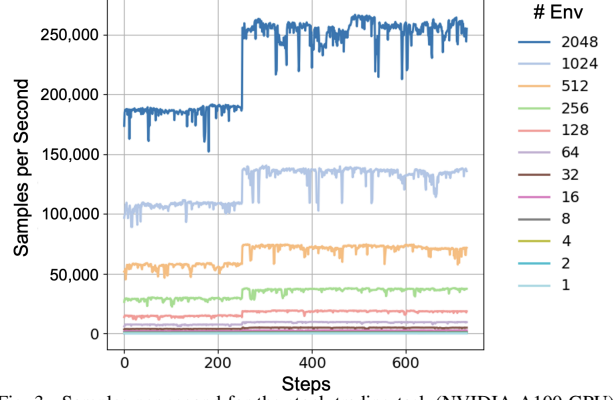


Fig. 3. Samples per second for the stock trading task (NVIDIA A100 GPU).

Family Fund for Research Innovations in FinTech, Engineering, and Business Operations.

REFERENCES

- [1] X.-Y. Liu, H. Yang, Q. Chen, R. Zhang, L. Yang, B. Xiao, and C. D. Wang, "FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance," *Deep Reinforcement Learning Workshop, NeurIPS*, 2020.
- [2] X.-Y. Liu, H. Yang, J. Gao, and C. D. Wang, "FinRL: deep reinforcement learning framework to automate trading in quantitative finance," *ACM International Conference on AI in Finance*, 2022.
- [3] X.-Y. Liu, Z. Xiong, S. Zhong, H. Yang, and A. Walid, "Practical deep reinforcement learning approach for stock trading," *Workshop on Challenges and Opportunities for AI in Financial Services, NeurIPS*, 2018.
- [4] B. Hambly, R. Xu, and H. Yang, "Recent advances in reinforcement learning in finance," *Mathematical Finance*, vol. 33, no. 3, pp. 437–503, 2023.
- [5] S. Sun, R. Wang, and B. An, "Reinforcement learning for quantitative trading," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 3, pp. 1–29, 2023.
- [6] Y. Bai, Y. Gao, R. Wan, S. Zhang, and R. Song, "A review of reinforcement learning in financial applications," *Annual Review of Statistics and Its Application*, vol. 12, no. 1, pp. 209–232, 2025.
- [7] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG et al., "Gymnasium: A standard interface for reinforcement learning environments," *arXiv:2407.17032*, 2024.
- [8] X.-Y. Liu, Z. Xia, J. Rui, J. Gao, H. Yang, M. Zhu, C. D. Wang, Z. Wang, and J. Guo, "FinRL-meta: market environments and benchmarks for data-driven financial reinforcement learning," in *International Conference on Neural Information Processing Systems*, NY, USA, 2022.
- [9] X.-Y. Liu, Z. Xia, H. Yang, J. Gao, D. Zha, M. Zhu, C. D. Wang, Z. Wang, and J. Guo, "Dynamic datasets and market environments for financial reinforcement learning," *Machine Learning - Nature*, 2024.
- [10] N. Holzer, K. Wang, K. Xiao, and X.-Y. L. Yanglet, "Revisiting ensemble methods for stock trading and crypto trading tasks at ACM ICAIF FinRL Contest 2023-2024," *arXiv:2501.10709*, 2025.
- [11] M. Benhenda, "FinRL-DeepSeek: Llm-infused risk-sensitive reinforcement learning for trading agents," *arXiv:2502.07393*, 2025.
- [12] A. Grover, "FinRLlama: A solution to llm-engineered signals challenge at finrl contest 2024," *arXiv:2502.01992*, 2025.
- [13] Y. Nie, Y. Kong, X. Dong, J. M. Mulvey, H. V. Poor, Q. Wen, and S. Zohren, "A survey of large language models for financial applications: Progress, prospects and challenges," *arXiv:2406.11903*, 2024.
- [14] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan et al., "DeepSeek-v3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.
- [15] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih, "Monte carlo gradient estimation in machine learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, 2020.