

FinWorld: An All-in-One Open-Source Platform for End-to-End Financial AI Research and Deployment

Wentao Zhang*
Nanyang Technological University
Skywork AI
Singapore
zhangwent963@gmail.com

Yilei Zhao*
Nanyang Technological University
Singapore
YILEI002@e.ntu.edu.sg

Chuoqiao Zong
Nanyang Technological University
Singapore
ZONG0005@e.ntu.edu.sg

Xinrun Wang[†]
Singapore Management University
Singapore
xrwang@smu.edu.sg

Bo An
Nanyang Technological University
Skywork AI
Singapore
boan@ntu.edu.sg

Abstract

Financial AI holds great promise for transforming modern finance, with the potential to support a wide range of tasks such as market forecasting, portfolio management, quantitative trading, and automated analysis. However, existing platforms remain limited in task coverage, lack robust multimodal data integration, and offer insufficient support for the training and deployment of large language models (LLMs). In response to these limitations, we present FinWorld, an all-in-one open-source platform that provides end-to-end support for the entire financial AI workflow, from data acquisition to experimentation and deployment. FinWorld distinguishes itself through native integration of heterogeneous financial data, unified support for diverse AI paradigms, and advanced agent automation, enabling seamless development and deployment. Leveraging data from 2 representative markets, 4 stock pools, and over 800 million financial data points, we conduct comprehensive experiments on 4 key financial AI tasks. These experiments systematically evaluate deep learning and reinforcement learning algorithms, with particular emphasis on RL-based finetuning for LLMs and LLM Agents. The empirical results demonstrate that FinWorld significantly enhances reproducibility, supports transparent benchmarking, and streamlines deployment, thereby providing a strong foundation for future research and real-world applications. Code, tutorials, and paper full version are available at Github and Arxiv ¹.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence; Machine learning algorithms.**

*These authors contributed equally to this work.

[†]Corresponding author. Email: xrwang@smu.edu.sg

¹<https://github.com/DVampire/FinWorld>. <https://arxiv.org/abs/2508.02292>



This work is licensed under a Creative Commons Attribution 4.0 International License. *KDD '26, Jeju Island, Republic of Korea*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2258-5/2026/08

<https://doi.org/10.1145/3770854.3785689>

Keywords

Financial AI, LLMs, LLMs Agent, Reinforcement Learning, Open-source Platform

ACM Reference Format:

Wentao Zhang, Yilei Zhao, Chuoqiao Zong, Xinrun Wang, and Bo An. 2026. FinWorld: An All-in-One Open-Source Platform for End-to-End Financial AI Research and Deployment. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '26)*, August 09–13, 2026, Jeju Island, Republic of Korea. ACM, New York, NY, USA, 30 pages. <https://doi.org/10.1145/3770854.3785689>

1 Introduction

Table 1: Comparison of FinWorld and related platforms. ✓: Supported, ✗: Not supported, Partial: Partially supported.

Key Features	FinWorld	TradeMaster	Qlib	FinRL-Meta
Multi-task support	✓	Partial	Partial	Partial
Multi-modal data	✓	✗	✗	✗
ML, DL, RL	✓	Partial	Partial	Partial
LLMs	✓	✗	✗	✗
LLMs Agent	✓	✗	✗	✗
Extensibility	✓	✓	Partial	Partial
Auto presentation	✓	Partial	✗	✗
Distributed Training	✓	✗	✗	✗
Benchmarking	✓	✓	✓	Partial

Financial AI has revolutionized how we approach market analysis, trading strategies, and investment decisions. From traditional quantitative machine learning (ML) models [12, 21] to modern deep learning (DL) and reinforcement learning (RL) architectures [20, 27, 28, 62], the field has witnessed remarkable progress across diverse domains including time series forecasting [8, 14], algorithmic trading [27, 36], portfolio management [23, 71], and natural language analysis for financial documents [25, 30]. However, this rapid advancement has created a fragmented ecosystem where researchers and practitioners must navigate multiple specialized tools, each optimized for specific tasks but lacking seamless integration.

Current financial AI platforms, while valuable in their respective domains, face several critical limitations. For instance, TradeMaster [40] provides RL methods for four financial trading tasks but lacks compatibility with traditional quantitative models and financial LLMs. Qlib [60] provides robust quantitative investment tools

but offers limited support for modern AI paradigms such as LLMs and RL. Similarly, FinRL-Meta [26] focuses specifically on RL for trading but suffers from a rigid framework architecture that makes it difficult to integrate new algorithms and lacks support for DL and LLM-based approaches. Most existing platforms are designed for narrow task scopes: some excel at time series forecasting but lack trading simulation, while others offer robust algorithmic trading or portfolio management yet have limited support for modern LLMs and agent paradigms. Furthermore, the integration of heterogeneous financial data ranging from structured market data to unstructured news and reports remains challenging, often requiring complex preprocessing pipelines and custom development efforts.

In summary, current financial AI platforms still face four major challenges: i) **Limited Task Coverage**: Insufficient support for emerging paradigms such as large language models and autonomous agents. ii) **Heterogeneous Data Integration**: Inadequate integration of heterogeneous data sources, including structured market data, unstructured news, and multimodal financial information. iii) **Rigid Framework Architecture**: Framework architectures that impede the seamless integration of novel algorithms and methodologies. iv) **Standardized Evaluation and Presentation**: Lack of standardized evaluation protocols and presentation frameworks for comprehensive performance assessment.

To address these challenges, we propose FinWorld, an all-in-one open-source platform for end-to-end financial AI research and deployment. FinWorld provides a unified framework that seamlessly integrates diverse AI paradigms, heterogeneous data sources, and modern technologies to enable comprehensive financial AI development and evaluation. Note that FinWorld is designed as a scalable foundation for research and experimentation, rather than for live high-frequency trading which relies on proprietary low-latency systems. However, its modular architecture emphasizes streaming data support, modular deployment, and decoupled design, making it extensible to real-time research applications. As shown in Table 1, FinWorld offers comprehensive support across all key features compared to existing platforms. The key features of FinWorld include:

- **Multi-task Support**: Unified platform supporting time series forecasting, algorithmic trading, portfolio management, and LLM applications, while existing platforms focus on narrow task scopes.
- **Multimodal Data Integration**: Native support for heterogeneous financial data, including structured market data, unstructured news, and multimodal information.
- **Comprehensive AI Paradigms**: Full support for ML, DL, RL, LLMs, and LLM agents, enabling seamless integration of traditional and modern AI approaches.
- **High Extensibility**: Modular and extensible framework design enabling rapid integration of novel algorithms, with flexibility facilitating research prototyping and real-world applications.
- **Advanced Automation**: With automated presentation and reporting features, and support for distributed multi-GPU training and testing, the system enables efficient exploration in multi-environments and supports high-performance computing.

To demonstrate the effectiveness of our proposed platform, we conduct extensive experiments and provide comprehensive benchmarks. Our main contributions are threefold:

- **Unified Framework**: We propose a unified, end-to-end framework for training and evaluation of ML, DL, RL, LLMs, and LLM agents, covering four critical financial AI task types including time series forecasting, algorithmic trading, portfolio management, and LLM applications.
- **Modular Design**: The framework features a modular architecture that enables flexible construction of custom models and tasks, including the development of personalized LLM agents. The system supports efficient distributed training and testing across multiple environments.
- **Comprehensive Benchmark**: We provide support for multimodal heterogeneous data with over 800 million samples, establishing a comprehensive benchmark for the financial AI community. Extensive experiments across four task types demonstrate the framework’s flexibility and effectiveness.

2 Related Work

2.1 ML, DL, and RL in Financial AI

Financial time series forecasting has evolved from traditional statistical methods like ARIMA [3] and GARCH [11] to modern DL approaches. ML methods such as LightGBM [21] and XGBoost [4] demonstrated superior performance in capturing complex financial patterns, while DL models including LSTM [19] and Transformer [48] architectures have revolutionized temporal modeling. Recent advances in financial forecasting include TimesNet [55], DLinear [66], and Timexer [29], which leverage attention mechanisms for improved prediction accuracy. In algorithmic trading, RL methods such as PPO [37] and DQN [32] have enabled adaptive trading strategies [20, 27], with recent work exploring transformer-based approaches [36]. Portfolio management has similarly progressed from classical mean-variance optimization [31] to data-driven approaches using ML [18] and RL [23, 71]. Despite these advances, existing approaches often focus on individual tasks, lacking unified frameworks for comprehensive financial AI development.

2.2 LLMs and LLM Agents in Financial AI

The integration of LLMs into financial decision-making has followed two main paths. The traditional approach uses pre-training and supervised fine-tuning (SFT), with models like FinBERT [2] and FLANG [38] excelling in financial text understanding. Recent progress includes FinGPT [25], BloombergGPT [57] with domain-adapted tokenization, and FinQA [5] featuring numerical reasoning. A newer direction combines pre-training with reinforcement learning (RL) to enhance reasoning, as shown by Fin-R1 [30] and Fino1 [35]. Despite their strengths, LLM-based methods often lack sequential decision-making, are computationally expensive, and struggle with non-stationary markets.

To overcome the limitations of traditional LLMs in sequential decision-making, researchers have introduced agentic mechanisms such as memory, tool use, and self-reflection. Early systems like FinMem [64] and FinRobot [74] rely on text-only LLMs enhanced with layered memory, profiling, and chain-of-thought reasoning

to improve single-asset trading performance. Building on this, FinAgent [70] represents the first multimodal trading agent, which jointly processes news, structured price data, and K-line chart visuals via a tool-augmented dual-reflection architecture, achieving significant improvements in profit. Recent efforts shift toward collaborative multi-agent systems: FinCon [65] adopts a manager–analyst hierarchy with verbal reinforcement, while TradingAgents [58] models a full trading firm with specialized agents coordinating to enhance returns and risk control. Beyond these architectural innovations, recent work has also explored using RL to finetune LLMs Agents [13, 34], enabling dynamic exploration and reasoning capabilities in real-world environments.

2.3 Financial AI Platforms

The development of financial AI platforms has been driven by the need to provide standardized environments for research and deployment. Qlib [60] provides a quantitative investment platform that integrates traditional ML pipelines with financial data processing, offering tools for feature engineering, model training, and portfolio optimization. TradeMaster [40] offers a comprehensive RL-based trading framework with modular components for strategy development, backtesting, and evaluation across multiple financial markets. FinRL-Meta [26] focuses on reinforcement learning environments and benchmarks, providing standardized market simulators and evaluation metrics for RL-based trading strategies. These platforms have significantly contributed to the advancement of financial AI research by providing researchers and practitioners with accessible tools and standardized evaluation frameworks. However, most existing platforms primarily focus on traditional ML, DL, and RL methods, with limited support for LLMs and LLMs Agents, and lack comprehensive multimodal data integration capabilities. This motivates our proposal of FinWorld, a comprehensive framework that covers most critical financial models and tasks.

3 Preliminaries

In this section, we formally define the core tasks considered in this work. For completeness, we provide mathematical formulations for time series forecasting, algorithmic trading, portfolio management, and LLMs applications in the financial domain. It is important to note that FinWorld is a unified platform framework that supports multiple AI paradigms (ML, DL, RL, LLMs, and LLM Agents), rather than a specific method itself. The definitions and formulations presented below describe the tasks and methods that the platform supports, helping readers understand the diverse approaches that can be implemented within the FinWorld framework.

We first introduce the notations used throughout this paper. For single asset scenarios, $\mathbf{x}_{1:T} = \{x_1, \dots, x_T\} \in \mathbb{R}^{T \times D}$ denote the historical endogenous time series (e.g., open, high, low, close price), and for multi-asset scenarios, $\mathbf{x}_{1:T} \in \mathbb{R}^{N \times T \times D}$ denote the historical price sequences for N assets. $\mathbf{z}_{1:T_{\text{ex}}} = \{\mathbf{z}_1, \dots, \mathbf{z}_{T_{\text{ex}}}\} \in \mathbb{R}^{T_{\text{ex}} \times C}$ denote exogenous covariate series (e.g., technical indicators, financial factors, news-based sentiment scores, or macroeconomic variables), where D is the dimension of the endogenous series and C is the number of exogenous features. Other symbols will be defined as needed in subsequent sections.

3.1 Time Series Forecasting Task

Unlike traditional time series forecasting, which often aims to predict the future value of a single series, financial forecasting has two key distinctions. First, directly forecasting asset prices is problematic due to their non-stationary nature and sensitivity to external shocks, so financial models typically predict returns, which are more stable and meaningful for investment decisions. Second, financial forecasting usually involves predicting the future returns of multiple assets simultaneously, reflecting the real-world requirements of portfolio management and quantitative strategies.

DEFINITION 1 (TIME SERIES FORECASTING). *Given the historical prices of N stocks $\mathbf{x}_{1:T} \in \mathbb{R}^{N \times T \times D}$ and multiple exogenous variables $\mathbf{z}_{1:T_{\text{ex}}}$, the goal of time series forecasting in finance is to predict the future S -day relative returns for all assets:*

$$\hat{\mathbf{R}}_{T+1:T+S} = \mathcal{F}_{\theta}(\mathbf{x}_{1:T}, \mathbf{z}_{1:T_{\text{ex}}}), \quad (1)$$

where $\hat{\mathbf{R}}_{T+1:T+S} \in \mathbb{R}^{N \times S}$ denotes the predicted relative returns (e.g., $\mathbf{R}_t = \frac{x_t}{x_T} - 1$), and $\mathcal{F}_{\theta}$ is a forecasting model parameterized by θ . Note that we use $\mathbf{R}$ to specifically denote predicted returns of multiple stocks, distinguishing it from the general target variable y used in other tasks.

3.2 Algorithmic Trading Task

This task involves the design, simulation, and evaluation of systematic trading strategies across single asset, emphasizing real-time decision making and risk management.

DEFINITION 2 (ALGORITHMIC TRADING). *Given historical observations $\mathbf{x}_{1:T}$ and exogenous variables $\mathbf{z}_{1:T_{\text{ex}}}$, the algorithmic trading problem can be addressed via two mainstream approaches:*

(a) ML&DL-based Approach. *The goal is to predict future price returns or movement directions using a predictive model $\mathcal{F}_{\theta}$:*

$$\hat{y}_{T+1:T+S} = \mathcal{F}_{\theta}(\mathbf{x}_{1:T}, \mathbf{z}_{1:T_{\text{ex}}}), \quad (2)$$

where $\hat{y}_t$ is a general target variable that can represent the predicted return, probability of upward/downward movement, risk score, or other trading signals, distinguishing it from $\mathbf{R}$ which specifically denotes multi-stock returns. The final trading actions $a_{T+1:T+S}$ are then determined by applying a pre-defined decision rule to $\hat{y}_{T+1:T+S}$ (e.g., buy if $\hat{y}_t > \tau$, sell if $\hat{y}_t < -\tau$, hold otherwise).

(b) RL-based Approach. *The trading task is modeled as a Markov decision process (MDP). At each time step t , the agent observes state s_t (constructed from $\mathbf{x}_1 : t$ and $\mathbf{z}_1 : t$), chooses action $a_t \in \mathcal{A}$, earns reward r_t , and moves to the next state s_{t+1} . The goal is to learn a policy $\pi_{\theta}(a_t | s_t)$ that maximizes expected cumulative reward:*

$$\max_{\theta} \mathbb{E}_{\pi_{\theta}} \left[\sum_{t=T+1}^{T+S} r_t \right]. \quad (3)$$

3.3 Portfolio Management Task

This task focuses on the construction, optimization, and dynamic rebalancing of investment portfolios, subject to real-world operational constraints (e.g., transaction costs, position limits), and supports various objective functions and risk measures such as return maximization, volatility minimization, and sharpe ratio optimization.

DEFINITION 3 (PORTFOLIO MANAGEMENT). Given historical price sequences $\mathbf{x}_{1:T} \in \mathbb{R}^{N \times T \times D}$ for N assets and exogenous variables $\mathbf{z}_{1:T_{\text{ex}}} \in \mathbb{R}^{N \times T_{\text{ex}} \times C}$, the portfolio management problem can be approached in two mainstream ways:

(a) ML&DL-based Approach. The goal is to predict future asset returns or risk estimates using a predictive model $\mathcal{F}_\theta$:

$$\hat{\mathbf{y}}_{T+1:T+S} = \mathcal{F}_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T_{\text{ex}}}), \quad (4)$$

where $\hat{\mathbf{y}}_t$ may represent the predicted return, risk score, or other signals for the N assets. The allocation weights $\mathbf{w}_{T+1:T+S}$ (where $\sum_{i=1}^N w_{t,i} = 1$ and $w_{t,i} \geq 0$ for all t) are then determined by applying an optimization procedure or rule to $\hat{\mathbf{y}}_{T+1:T+S}$ (e.g., mean-variance optimization, risk parity, or rule-based allocation).

(b) RL-based Approach. This task is formulated as a sequential decision process, where at each time t , the agent observes a state s_t (e.g., constructed from $\mathbf{x}_{1:t}$ and $\mathbf{z}_{1:t}$), selects allocation weights $\mathbf{w}_t \in \Delta^{N+1}$ (where $\sum_{i=1}^{N+1} w_{t,i} = 1$ and $w_{t,i} \geq 0$, with $w_{t,0}$ representing cash position), receives a reward r_t (e.g., portfolio return or risk-adjusted reward), and transitions to the next state s_{t+1} . The objective is to learn a policy $\pi_\theta(\mathbf{w}_t | s_t)$ that maximizes expected cumulative utility:

$$\max_{\theta} \mathbb{E}_{\pi_\theta} \left[\sum_{t=T+1}^{T+S} U(\mathbf{w}_t, \mathbf{x}_t) \right], \quad (5)$$

where $U(\cdot)$ represents cumulative portfolio return.

3.4 LLMs Applications

Encompassing two main categories of LLM applications in finance: i) General language understanding tasks, including SFT and RL training for LLMs and LLM agents on financial text analysis, financial QA, and similar language comprehension tasks, and ii) Sequential decision-making tasks, involving RL training for LLMs or LLM agents through real-world environment interactions, such as trading in live market environments.

DEFINITION 4 (LLMs APPLICATIONS). Given financial-domain inputs from multiple modalities, such as unstructured text (e.g., news, reports), structured time series (e.g., open, high, low, close price), images (e.g., Kline charts), and audio or video data (e.g., financial broadcasts), the goal is to train and deploy large language models $\mathcal{M}_\phi$ for two primary application types: general language understanding and sequential decision-making. Formally,

$$\mathbf{y} = \mathcal{M}_\phi(\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_K), \quad (6)$$

where each $\mathbf{D}_k$ represents an input modality, and $\mathbf{y}$ is the task-specific output in this financial context.

Depending on the specific downstream task, LLMs can flexibly serve as predictive models within ML&DL-based pipelines, or act as autonomous agents capable of RL-based decision-making and advanced tool utilization in finance.

3.5 Reinforcement Learning for LLMs

Group Relative Policy Optimization (GRPO) [39] is a policy optimization algorithm designed to efficiently train LLMs in both language and environment-interactive settings. In the context of financial AI, GRPO can be leveraged to endow LLMs with either general financial knowledge and reasoning ability, and specialized trading skills. For example, when applied to financial document

analysis or financial question answering, GRPO enables the LLM to align with environment feedback and develop robust understanding of domain-specific content. In these tasks, the LLM is trained to produce high-quality financial responses, with group-level reward normalization providing stable learning signals for fine-tuning.

Moreover, when applied in simulated trading environments, GRPO can be used to train LLMs through direct environment interaction. The LLM receives financial observations (such as price time series, technical indicators, and news), generates actions (e.g., BUY, HOLD, SELL), and is rewarded according to realized trading returns or risk-adjusted performance. This allows the LLM to acquire real trading capabilities, strategy adaptation, and risk management skills beyond static data learning.

GRPO is applied to our LLM training through a two-stage RL paradigm, where the first stage focuses on equip the model's financial reasoning abilities using financial reasoning datasets, and the second stage immerses the model in real or simulated market environments to develop practical decision-making skills. For more details, please refer to Appendix E.2.

4 FinWorld

This section introduces FinWorld, a unified, modular platform designed to overcome the limits of current financial-AI frameworks. A clean, layered architecture lets researchers combine traditional ML, DL, RL, LLMs, and LLMs Agents-based methods for mainstream financial task while keeping concerns clearly separated. Integrated dataset management, standardized model APIs, and a scalable training back-end support both academic studies and real-world deployment. Architectural details appear in Appendix F.

4.1 Design Principles

The design of FinWorld is underpinned by foundational principles, delivering a robust, extensible, research-oriented platform for the development and evaluation of financial AI models and systems:

- **Layered and Object-Oriented Architecture.** We employ a hierarchical, layered architecture with object-oriented design principles to ensure clear separation of concerns and facilitate both flexibility and scalability across the platform. The seven-layer architecture (Configuration, Dataset, Model, Training, Evaluation, Task, and Presentation) represents a complete experimental system design for financial ML/LLM workflows, rather than a stackable model structure. Each layer serves an essential role in standardized workflows, and removing any layer would break the system's integrity, making it a platform-level design rather than a model stack.
- **Modular and Decoupled Design.** FinWorld adopts a fully modular architecture. Each component is developed as a self-contained unit with well-defined interfaces, enabling separate optimization and streamlined integration of custom components.
- **Extensibility and Paradigm Fusion.** FinWorld emphasizes extensibility at all levels, providing standardized extension points for seamless integration of novel algorithms and datasets. This architecture natively supports the fusion of diverse paradigms across financial AI research.

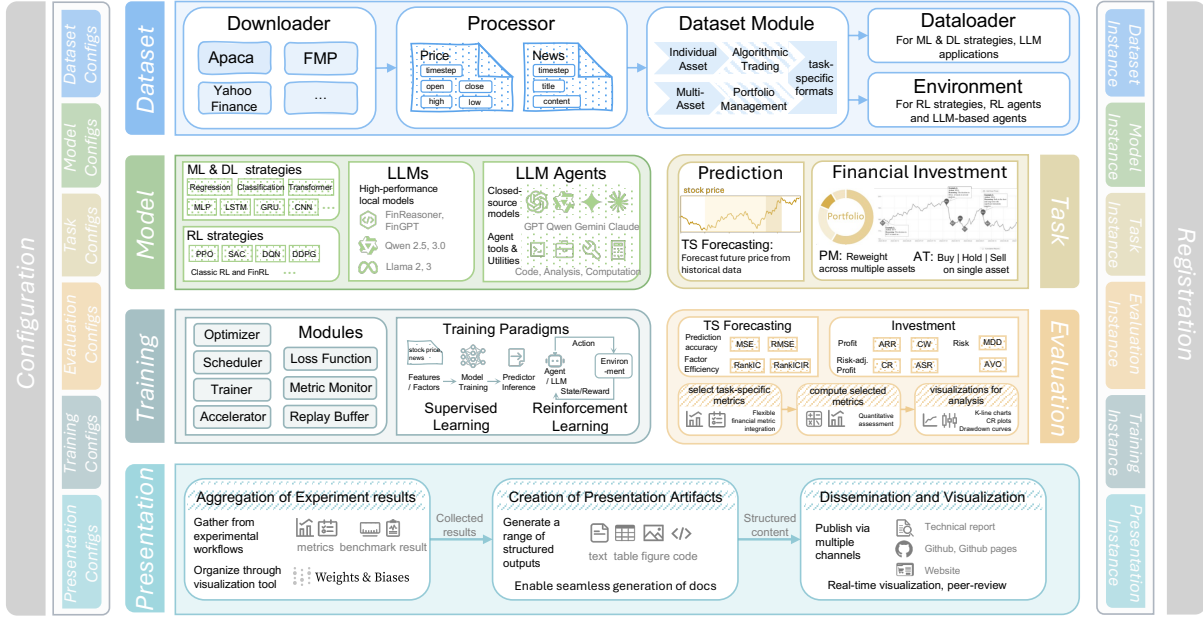


Figure 1: Overview of FinWorld.

4.2 Configuration Layer

The configuration layer of FinWorld is built on `mmengine` [7] and provides a unified and extensible system using dictionaries. It centralizes all experimental settings, including datasets, models, training, and evaluation, in a readable and modular format. Support for configuration inheritance and overrides ensures reproducibility, flexibility, and collaborative development. In the meantime, FinWorld uses a registry mechanism to manage core components such as the dataset and the environment. Each class is registered by its type and instantiated from the configuration, allowing for flexible component management and a decoupled system architecture.

4.3 Dataset Layer

The dataset layer of FinWorld comprises multiple functional modules designed to enable standardized, extensible, and task-oriented data management for financial AI research. This layer abstracts the complexities of diverse data sources and modalities, offering a unified interface for data acquisition, preprocessing, and task-specific organization. Specifically, it consists of five main modules: data downloader module, data processor module, dataset module, dataloader module, and environment module, each supporting flexible customization and extensibility.

Downloader Module. This module consists of two main components: i) Market Data Downloader, which standardizes access to heterogeneous financial market data sources, such as FMP [42] and Alpaca [41], supporting multiple time resolutions (e.g., daily, minute-level) and data types (e.g., OHLCV, news); and ii) LLM Reasoning Dataset Downloader, which provides access to financial reasoning datasets including FinQA, professional certification materials (e.g., ACCA, CFA), and other financial knowledge bases. The unified downloader abstraction streamlines data acquisition from various APIs and databases, ensuring consistency and scalability across research tasks.

Processor Module. Building on the standardized data access, this module enables users to configure essential preprocessing steps, such as factor computation (e.g., `alpha158` [60]), feature selection, normalization, outlier filtering, and metadata tracking. The module supports multimodal alignment and quality analysis (e.g., anomaly detection) to mitigate data biases, particularly addressing single-market bias through multi-market, multi-frequency, and long-horizon data integration. LLM reasoning data processing and other preprocessing steps can be configured according to specific research needs. The processor abstraction facilitates reproducible and customizable data transformations for downstream modeling, with the modular pipeline supporting user-defined quality analysis.

Dataset Module. This module organizes processed data into task-specific formats suitable for various financial applications. For algorithmic trading tasks, it encapsulates data as individual asset datasets; for portfolio management and multi-asset problems, it constructs unified multi-asset datasets. The module seamlessly handles multiple data modalities, including numerical time series, structured financial data, and unstructured textual information, enabling comprehensive modeling capabilities for both traditional financial AI tasks and LLM-based applications.

Dataloader Module. This module provides standardized dataloaders for ML, DL, or LLMs applications, such as time series forecasting. It enables efficient batching and sampling, facilitating seamless integration with other frameworks.

Environment Module. Designed for reinforcement learning paradigms, this module encapsulates data as interactive environments. It supports both conventional RL agents and LLM-based agent interactions, enabling unified experimentation and promoting reproducibility across a variety of agent-based learning tasks.

4.4 Model Layer

The model layer of FinWorld consists of multiple specialized modules that enable unified definition, management, and invocation

of diverse modeling paradigms within the platform. This layer abstracts the construction and orchestration of traditional ML models, deep neural network, and LLMs, providing standardized interfaces for consistent training, inference, and integration across multi-tasks. Specifically, it is organized into four main modules:

ML Models. This module defines classical model structures in a unified declarative form. It includes linear and logistic regression, decision trees, random forests, and gradient boosting ensembles (for example, XGBoost and LightGBM). Each specification exposes a consistent input and output schema and a task head (regression, binary classification, or multiclass classification), with structural attributes such as regularization.

DL Models. This module organizes neural architectures into components, layers, and model networks. Components include data embedding for financial time series [55], patch embedding [9], position encodings, attention, transformer blocks, and activation functions. Layers compose these into an embedding layer, an encoder, and a decoder with consistent interfaces. Model networks such as Autoformer, VAE, and GPT-style decoders are instantiated by wiring the layers under a common specification, independent of training and inference.

RL Models. This module defines RL network structures with an actor-critic abstraction. Policy and value networks are composed from DL models components, including embedding layers and backbones such as MLP, LSTM/GRU, and Transformer. Specifications cover discrete or continuous action heads and value heads, with shared or separate encoders and optional memory. Financial constraints are represented as structural hooks for transaction costs, slippage and market impact, risk limits, and trading calendars, supporting trading and portfolio tasks.

LLMs Models. This module centralizes access to both proprietary and open-source LLMs via a unified interface. It supports seamless switching among commercial providers (e.g., GPT-4.1, Claude-4-Sonnet, Gemini-2.5-Pro) and efficient local models (e.g., Qwen2.5, Qwen3), with standardized controls for context length, sampling, and cost/latency tracking. Built-in tools include function calling, tool use, and retrieval-augmented generation for financial documents. The module integrates naturally with agents and downstream models, enabling document understanding, information extraction, instruction following.

4.5 Training Layer

The training layer in FinWorld offers a modular scaffold that abstracts every element required to optimise all method pipelines for financial applications. A uniform interface guarantees experiment reproducibility, smooth scaling from single GPU notebooks to distributed clusters, and quick transfer of best practices across tasks such as time series forecasting, trading, portfolio management, and LLM applications.

Optimizer. A rich catalogue of first order methods, ranging from classic stochastic gradient descent (SGD) to adaptive variants such as Adam and AdamW, lets practitioners match optimisation dynamics to task characteristics. A common wrapper normalises hyperparameter signatures and supports gradient centralisation, decoupled weight decay, and mixed precision updates, ensuring robust convergence on noisy and non stationary market data.

Loss. A flexible factory produces objective functions that cover regression losses (mean squared error and mean absolute error), classification losses for event prediction, and reinforcement learning surrogates for policy and value objectives. Composite losses can be declared with a single line, enabling multi-task learning or the joint optimisation of risk adjusted return and prediction accuracy.

Scheduler. Static strategies (e.g., step, cosine, linear) and adaptive schemes (e.g., warm-up then decay) are available for both learning rates and regularisation coefficients. Schedulers are time-aware, resuming the exact trajectory when checkpoints are reloaded.

Metrics. In addition to generic accuracy and error scores, the library ships with finance specific diagnostics such as ARR, SR, and MDD. Each metric is logged at configurable intervals and can trigger early stopping or hyperparameter sweeps, enabling data-driven model selection. While the current version focuses on a unified pipeline for data processing, model training, and evaluation rather than a built-in hyperparameter tuning platform, interfaces are provided for users to integrate external optimization methods such as Optuna, grid search, and Bayesian optimization.

Trainer. Acting as the orchestrator, the trainer pipes data loading, forward and backward passes, gradient clipping, metric evaluation, checkpointing, and experiment logging (TensorBoard or WandB). Specialised variants provide task specific logic, for example the forecasting trainer, trading trainer, portfolio trainer, and large language model trainer. Clear callback hooks let researchers inject custom steps, such as on-the-fly data augmentation or bespoke risk constraints, without changing the core loop.

Together, these components form a coherent architecture that accelerates experimentation, strengthens reproducibility, and lowers the barrier to launching state-of-the-art AI solutions in the demanding environment of financial markets.

4.6 Evaluation Layer

The evaluation layer in FinWorld provides a comprehensive and extensible framework for assessing financial AI models and strategies. It dynamically selects and applies appropriate evaluation protocols and metrics based on the specific task and model type, supporting both established benchmarks and user-defined criteria. The framework incorporates a diverse library of financial and predictive metrics, such as ARR, MDD, SR, and MSE, which can be flexibly combined or extended as needed. In addition to quantitative assessment, the evaluation layer provides advanced visualization tools, including candlestick (K-line) charts, cumulative return plots, draw-down curves, and trade annotation overlays, to facilitate intuitive interpretation and diagnosis of model performance. This flexible and adaptive evaluation process not only streamlines model assessment for diverse financial tasks, but also facilitates systematic comparison, rapid diagnosis of strengths and weaknesses, and iterative improvement of financial AI methods within the platform. Integrated into the trainer's validation and test stages, it ensures consistency with training-phase evaluation and supports systematic comparison, diagnosis, and iterative improvement.

4.7 Task Layer

The task layer is responsible for the systematic definition, abstraction, and encapsulation of financial AI task types. It systematically

supports a wide spectrum of financial AI tasks, providing unified abstractions and modular interfaces that facilitate integration with upstream data modules and downstream modeling components. This layer centers on several core financial tasks, including time series forecasting, algorithmic trading, portfolio management, and LLMs applications, which are formally defined in Section 3.

Each task is specified by configurable input/output schemas and standardized evaluation protocols, supporting both established benchmarks and user-customized scenarios. The unified task architecture enables rapid prototyping, cross-task generalization, and reproducible research across financial AI applications. Additionally, the layer provides a flexible framework for deploying LLM-based agents in asynchronous, single-agent, or multi-agent configurations, allowing users to compose and customize financial agents for diverse research and application needs. The framework’s extensibility is demonstrated through support for extensible task types such as Factor Models and Market Representation Models, as well as integration with AgentOrchestra, enabling users to build agents beyond finance, thereby showcasing the platform’s strong extensibility across domains.

4.8 Presentation Layer

The presentation layer in FinWorld is designed for automated dissemination and documentation of experimental results. Central to this layer is a dedicated presentation agent that orchestrates the aggregation of evaluation outputs, automatic generation of technical reports, and the creation of interactive web pages for result interpretation and dissemination. The platform provides open visualization interfaces using Plotly and Wandb, enabling users to customize metrics, tables, and charts for their specific experiments. Users can extend or replace visualization templates to suit custom research needs. The open-source release includes example configuration files and scripts demonstrating how to load data, define tasks, configure models, and execute training workflows, enabling users to start training and testing with a single command. Standardized APIs, example scripts, and templates enable easy customization and rapid adoption. Experimental findings, along with key visualizations and benchmark summaries, are systematically compiled into structured documents and published to collaborative platforms such as GitHub, ensuring transparent sharing and long-term accessibility. Furthermore, seamless integration with experiment tracking tools like Wandb enables real-time visualization and comparison of metrics throughout the research lifecycle. This automated, multi-channel presentation workflow enhances reproducibility, supports peer review, and amplifies the visibility and impact of financial AI research conducted within the platform.

5 Empirical Evaluation

5.1 Dataset

We utilize two representative markets, the US and China, covering four major stock pools: DJ30, SP500, SSE50, and HS300. The dataset spans from 1995-05-01 to 2025-05-01, comprising daily and minute-level **price** as well as **news**, totaling over 800 million data points. This extensive dataset supports tasks such as time series forecasting, algorithmic trading, and portfolio management. Additionally, we collect a comprehensive **LLM Reasoning dataset** in both Chinese

and English, covering diverse financial scenarios for training LLMs and agents, with over 80,000 samples across multiple reasoning benchmarks. The LLM evaluation encompasses multi-language, multi-scenario, and multi-task applicability: datasets cover both Chinese and English languages with numerical (tabular) and textual modalities, spanning diverse financial scenarios such as QA, domain-specific corpora, professional exams (e.g., ACCA, CFA), business knowledge, tabular data, market analysis, multi-turn dialogue, and quantitative trading. Each scenario is carefully designed to reflect realistic financial reasoning and decision-making challenges, ensuring comprehensive validation across multi-language, multi-task, and multi-scenario financial applications. All experiments are conducted on 2 NVIDIA H100 GPUs. Unless otherwise specified, all reported results are averaged over three runs with different random seeds, and the best results in each column are underlined. Detailed information can be found in Appendix A.

6 Experimental Setup

To ensure fair comparison and reproducibility across all experiments, we adopt a unified experimental configuration. For trading tasks (including DL, RL, and LLM Agents), we define trading actions as follows: the **BUY** action corresponds to taking a full long position, and the **SELL** action corresponds to taking a full short position. Both trading and portfolio management tasks start with an initial capital of 100,000, employ a transaction fee rate of 1×10^{-4} , and operate without leverage. This standardized setup enables consistent evaluation across different methods and facilitates fair comparison of their performance.

6.1 Time Series Forecasting

Dataset Setup. We use daily OHLCV and Alpha158 features for DJ30, SP500, SSE50, and HS300 from 2015-05-01 to 2025-05-01, with per-stock normalization and data split at 2023-05-01 for training and validation. Detailed information can be found in Appendix D.1. **Metrics.** We evaluate forecasting performance using four metrics: MAE, MSE, RankIC, and RankICIR. MAE and MSE focus on absolute prediction accuracy, while RankIC and RankICIR are particularly relevant for evaluating the model’s effectiveness in capturing return-based or ranking-based financial relationships. **Methods.** We evaluate several ML-based and DL-based time series forecasting models, including: i) ML-based methods: **LightGBM** [60], **XGBoost** [60]; ii) DL-based methods: **Autoformer** [56], **Crossformer** [72], **ETSformer** [54], **DLinear** [67], **TimesNet** [55], **PatchTST** [33], **TimeMixer** [50], and **TimeXer** [51]. These models are selected based on their effectiveness in capturing complex patterns and their ability to handle high-dimensional data.

Experiment Results As shown in the results Table 2, on the DJ30 dataset, the TimeXer model achieves a MAE of 0.0529 and an MSE of 0.0062, significantly lower than LightGBM (MAE 0.1392, MSE 0.0235). TimeXer also attains the highest RankICIR of 0.4889, compared to 0.2017 for LightGBM. Similarly, on the HS300 dataset, models such as TimeMixer and TimeXer outperform ML methods, with MAEs of 0.3804 and 0.3727, and MSEs of 0.0442 and 0.0529, respectively. Overall, deep learning models achieve lower prediction

errors and higher rank-based metrics than machine learning methods, highlighting the advantages of DL approaches for financial time series forecasting.

Table 2: Comparison of models for time series forecasting.

Model	DJ30				SP500			
	MAE ↓	MSE ↓	RankIC ↑	RankICIR ↑	MAE ↓	MSE ↓	RankIC ↑	RankICIR ↑
ML-based								
LightGBM	0.1392	0.0235	0.0157	0.2017	0.6741	0.0788	0.0216	0.2242
XGBoost	0.1476	0.0251	0.0138	0.1684	0.7025	0.0814	0.0199	0.1925
DL-based								
Autoformer	0.1044	0.0182	0.0193	0.3265	0.4568	0.0671	0.0312	0.3379
Crossformer	0.1624	0.0173	0.0150	0.3561	0.6448	0.0774	0.0219	0.2984
ETsformer	0.0640	0.0115	0.0209	0.2411	0.2560	0.0448	0.0295	0.3125
DLinear	0.0867	0.0122	0.0144	0.2247	0.3537	0.0416	0.0177	0.2783
TimesNet	0.1242	0.0142	0.0211	0.2642	0.4013	0.0660	0.0483	0.3512
PatchTST	0.0830	0.0112	0.0140	0.2134	0.3682	0.0498	0.0157	0.2927
TimeMixer	0.0620	0.0081	0.0294	0.4741	0.2278	0.0279	0.0349	0.1193
TimeXer	0.0529	0.0062	0.0302	0.4889	0.1871	0.0240	0.0425	0.3255

Model	SSE50				HS300			
	MAE ↓	MSE ↓	RankIC ↑	RankICIR ↑	MAE ↓	MSE ↓	RankIC ↑	RankICIR ↑
ML-based								
LightGBM	0.1540	0.0267	0.0211	0.1688	0.7836	0.1097	0.0219	0.1961
XGBoost	0.1632	0.0293	0.0184	0.1497	0.8140	0.1123	0.0193	0.1724
DL-based								
Autoformer	0.1163	0.0212	0.0336	0.1848	0.5696	0.1213	0.0364	0.2493
Crossformer	0.1756	0.0197	0.0225	0.1318	0.9829	0.1082	0.0187	0.1525
ETsformer	0.0737	0.0125	0.0452	0.2107	0.4413	0.0546	0.0332	0.2406
DLinear	0.0932	0.0154	0.0343	0.1722	0.4315	0.0717	0.0285	0.2078
TimesNet	0.1329	0.0168	0.0251	0.1465	0.7118	0.0997	0.0241	0.1894
PatchTST	0.0907	0.0135	0.0377	0.1934	0.3253	0.0595	0.0276	0.1971
TimeMixer	0.0724	0.0119	0.0402	0.2086	0.3804	0.0442	0.0348	0.2417
TimeXer	0.0625	0.0093	0.0425	0.2173	0.3727	0.0529	0.0352	0.2478

Table 3: Comparison of models for algorithmic trading.

Model	AAPL					AMZN				
	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓
Rule-based										
BUY&HOLD	13.0620	0.6654	17.6828	0.7387	0.9606	0.2210	43.9823	1.4190	19.6246	2.2412
MACD	9.7042	0.6964	11.5621	0.8393	1.0251	0.1487	1.4856	0.0028	19.4850	0.0762
ML-based										
LightGBM	7.4450	0.5112	16.7097	0.4456	0.7363	0.1679	2.6266	-0.1886	38.1339	-0.0689
XGBoost	15.7943	1.0359	17.6320	2.0695	2.0720	0.1185	20.8648	0.7175	26.5675	0.3134
DL-based										
Transformer	23.3469	0.8788	18.7073	1.2480	1.6204	0.2809	5.8822	0.3716	25.5574	0.0884
LSTM	11.2387	0.5262	35.9190	0.2448	0.9589	0.2647	15.5243	0.8552	40.3636	0.3846
DLinear	9.5539	0.4686	34.7311	0.2136	0.8594	0.2644	9.9999	0.7221	30.5673	0.3272
RL-based										
PPO	31.5838	1.0483	18.7102	1.6222	1.9040	0.2956	37.3429	1.2514	22.2379	1.6793
SAC	28.2963	0.8758	17.7384	0.6467	1.5797	0.2970	44.4502	1.6389	23.4218	1.4709

Model	GOOGL					META				
	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓
Rule-based										
BUY&HOLD	28.1837	1.0236	22.2309	1.1678	1.4589	0.2811	67.2719	1.7264	18.8226	1.5740
MACD	10.0432	0.6068	12.6638	0.7931	0.8620	0.1862	40.3374	1.5468	12.7635	1.1604
ML-based										
LightGBM	16.4245	1.0908	21.0885	0.7788	1.6732	0.1468	39.6612	1.1212	23.4416	0.7655
XGBoost	7.5362	0.6783	20.6611	0.3648	1.2239	0.1171	32.0035	1.6461	24.2472	1.3199
DL-based										
Transformer	10.3923	0.6683	36.7980	0.2824	0.9652	0.1468	48.2834	1.6819	29.0003	1.6649
LSTM	18.7416	0.8849	36.8169	0.5091	1.2871	0.2221	44.6391	1.1864	30.0527	0.8918
DLinear	21.4708	1.0461	36.8203	0.5831	1.5575	0.2063	35.3426	1.445	34.0928	1.0366
RL-based										
PPO	20.5532	1.0145	16.7733	1.2254	1.5855	0.2048	72.0104	2.1031	18.6421	1.5360
SAC	30.6081	1.2203	35.4036	1.7233	1.8833	0.2038	56.8891	1.5085	34.6565	1.6415

Model	MSFT					TSLA				
	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓
Rule-based										
BUY&HOLD	13.3812	0.6991	18.6840	0.7162	0.9635	0.2116	30.8051	0.7469	53.6797	0.5739
MACD	6.4616	0.4703	13.7325	0.4705	0.6337	0.1606	68.8393	1.5085	21.7414	2.1663
ML-based										
LightGBM	20.7821	0.9031	37.8864	0.5485	1.4983	0.2400	49.2144	1.3086	56.0589	0.8779
XGBoost	14.4887	0.6633	49.8944	0.2904	1.0676	0.2497	39.7010	0.9788	58.5673	0.5053
DL-based										
Transformer	23.2662	1.0207	32.8318	0.6037	1.4478	0.2180	73.1291	1.2581	48.5669	0.9308
LSTM	16.5063	0.6185	31.8587	0.3910	1.0206	0.2162	44.7985	1.0768	53.6992	1.0904
DLinear	9.4778	0.5599	31.7271	0.2987	0.7792	0.1962	41.1897	1.0612	62.3380	0.6608
RL-based										
PPO	29.5326	1.1044	18.6900	1.5801	1.7624	0.2654	58.5511	1.0402	48.5536	0.5703
SAC	22.2784	0.9515	27.7286	0.8035	1.5859	0.2405	70.548	1.6474	57.4088	1.5065

6.2 Algorithmic Trading

Dataset Setup. We use daily OHLCV and Alpha158 features for six US stocks, AAPL, AMZN, GOOGL, META, MSFT, and TSLA, from 1995-05-01 to 2025-05-01, with per-stock normalization and data split at 2023-05-01 for training and validation. Detailed information can be found in Appendix D.2.

Metrics. We evaluate trading performance using 6 metrics: **ARR**, **MDD**, **VOL**, **CR**, **SR**, and **SoR**. ARR measures profitability. MDD captures the worst drawdown, and VOL reflects return variability, both indicating risk exposure. CR evaluates returns relative to drawdown risk, and SR and SoR assess risk-adjusted returns based on total and downside volatility, respectively. Together, these metrics provide a comprehensive view of profitability and risk.

Methods. We systematically evaluate several Rule-based, ML-based, DL-based, and RL-based methods, including i) Rule-based methods: **BUY&HOLD**, **MACD**; ii) ML-based Methods: **LightGBM**, **XGBoost**; iii) DL-based Methods: **Transformer** [48], **LSTM**, **DLinear** [67]; iv) RL-based Methods: **PPO** [37], **SAC** [16].

Experiment Results. Table 3 shows that RL methods achieve the best overall trading performance, with SAC and PPO delivering the highest annualized returns and SR across most stocks (e.g., SAC achieves 101.55% ARR on TSLA and PPO attains 2.10 SR on META). DL models like Transformer also outperform ML-based and rule-based methods in both return and risk metrics (e.g., 73.13% ARR on MSFT). Compared to ML, DL and rule-based strategies, RL methods show clear advantages in both returns and risk-adjusted performance.

Table 4: Comparison of methods for portfolio management.

Model	DJ30					SP500						
	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓		
Rule-based												
BUY&HOLD	9.4049	0.8792	9.2585	1.0158	1.2779	0.1089	8.9875	0.7617	13.7434	0.6539	1.1161	0.1228
ML-based												
LightGBM	16.3302	0.5758	30.7444	1.1107	1.7603	0.5162	13.4204	0.5916	30.7037	0.7206	0.9005	0.4149
XGBoost	10.6903	0.3776	30.4737	1.6504	1.3008	0.5167	10.2609	0.3437	30.5994	0.8202	0.7601	0.4135
DL-based												
Autoformer	-4.1781	0.3611	34.2251	-0.0650	0.5267	0.1123	9.7203	0.3724	33.5112	0.4257	0.5913	0.1563
Crossformer	9.6460	0.4474	33.8498	0.1840	0.6459	0.2135	8.2108	0.3879	29.9936	0.5341	0.928	0.1289
ETsformer	1.5512	0.3662	31.5739	0.0300	0.5085	0.0918	12.0507	0.4951	30.7685	0.7532	0.657	0.2342
DLinear	17.4357	0.7461	35.2342	0.5593	1.0521	0.1787	17.1605	0.7827	32.9918	1.1329	1.012	0.0934
TimesNet	16.7235	0.8123	29.4762	0.9821	1.0844	0.2129	12.9701	0.6474	29.8908	0.9891	0.9136	0.1662
PatchTST	13.5049	0.7344	31.0133	0.8961	1.0317	0.1036	9.2306	0.4719	31.6127	0.7654	0.8338	0.0904
TimeMixer	9.1872	0.5795	30.9910	0.6928	0.8996	0.1861	14.4202	0.7523	30.4321	1.0876	1.0284	0.2172
TimeXer	7.4083	0.4567	32.6578	0.5129	0.8103	0.3184	19.8309	0.8692	28.3254	1.2768	1.1825	0.2841
RL-based												
PPO	11.7241	1.0339	20.1129	1.2865	1.5246	0.2133	28.5408	1.1136	29.5874	1.3761	1.5589	0.3146
SAC	19.0272	1.5967	20.0846	2.0945	2.4453	0.2117	31.2104	1.5987	30.3381	2.1362	2.4625	0.2338

Model	SSE50					HS300						
	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓	ARR ↑	SR ↑	MDD ↓	CR ↑	SoR ↑ VOL ↓		
Rule-based												
BUY&HOLD	2.8799	0.2512	0.7742	0.1420	0.3868	0.1706	2.7940	0.2448	19.7412	0.1415	0.0319	0.1738
ML-based												
LightGBM	13.4202	0.5915	0.7032	0.5973	0.9632	0.1472	26.5361	0.6509	0.9407	1.2177	1.4964	0.1693
XGBoost	14.3904	0.3439	0.5991	0.5945	0.8581	0.1358	17.2650	0.3718	0.5823	0.5876	0.9741	0.2116
DL-based												
Autoformer	9.4209	0.5162	8.8093	0.8494	0.9662	0.1586	18.7107	0.7968	1.1079	1.2124	1.1872	0.1943
Crossformer	10.5902	0.5731	9.0107	0.9798	1.1162	0.1741	21.8814	0.8283	1.1523	1.2981	1.2357	0.2162
ETsformer	12.1572	0.6703	0.9755	1.0621	1.1603	0.1418	15.2038	0.7741	1.0635	1.0816	1.1341	0.2328
DLinear	17.7206	0.8581	1.0781	1.1941	1.2786	0.1913	29.9904	0.8931	1.1172	1.2425	1.2456	0.2872
TimesNet	14.4103	0.8382	1.0473	1.1495	1.2639	0.1684	24.3309	0.9207	1.1594	1.2930	1.2440	0.2933
PatchTST	15.1308	0.9072	0.9750	1.2366	1.2821	0.1924	23.6607	0.9153	1.1333	1.2645	1.1875	0.2067
TimeMixer	12.8201	0.8894	0.9456	1.2902	1.3488	0.2031	19.1205	0.9172	1.0581	1.1426	1.2396	0.2486
TimeXer	11.4708	0.8781	0.8942	1.3179	1.4239	0.1172	18.1713	0.9137	1.0503	1.1104	1.2938	0.2173
RL-based												
PPO	15.1502	1.0013	0.6331	1.3956	1.4811	0.1835	22.2032	1.0578	0.8171	1.1763	1.2541	0.1568
SAC	20.0670	1.0625	0.7132	1.5320	1.5781	0.1497	39.0005	1.1606	0.8205	1.2303	1.3703	0.1686

ML&DL-based methods include all ML&DL-based forecasting models. Specifically, we consider: i) Rule-based methods: **BUY&HOLD**; ii) ML-based Methods: **LightGBM**, **XGBoost**; iii) DL-based Methods: **Autoformer** [56], **Crossformer** [72], **ETSformer** [54], **DLinear** [67], **TimesNet** [55], **PatchTST** [33], **TimeMixer** [50], and **TimeXer** [51]; iv) RL-based Methods: **PPO** [37], **SAC** [16].

Experiment Results Overall, the RL-based methods, especially SAC, achieve the best results on all benchmarks, with annualized returns up to 31.2% (SP500) and Sharpe ratios above 1.5. In contrast, rule-based and ML-based methods show much lower returns (e.g., Buy&Hold: 9.4% on DJ30), and DL-based methods generally perform between the two. Notably, SAC consistently delivers higher returns and better risk-adjusted metrics, demonstrating clear advantages for portfolio management.

6.4 LLMs Applications

Dataset Setup. For comprehensive evaluation, we assess LLMs and LLM Agents on a suite of established financial reasoning benchmarks, including FinQA, FinEval, ConvFinQA, and CFLUE, which cover a broad range of business scenarios. For trading evaluation, we follow the same experimental setup as in algorithmic trading task and use daily-level OHLCV data together with news for the 6 U.S. technology stocks (AAPL, AMZN, GOOGL, META, MSFT, and TSLA) over the period from 2015-05-01 to 2025-05-01.

Metrics. For financial reasoning task, we evaluate performance using **Score**. For trading abilities, we use the same six financial metrics as in the above trading task: **ARR**, **SR**, **MDD**, **CR**, **SoR**, and **VOL**.

Methods. For financial reasoning, we evaluate our proposed models FinReasoner against open-source LLMs on the test dataset, including **DeepSeek-R1** [15], **Qwen3-8B** [59], **Fin-R1-7B** [15], and **Qwen2.5-7B-Instruct** [44]. For trading abilities, we evaluate our own proposed FinReasoner with **DeepSeek-R1** [15], **Qwen3-8B** [59], **Fin-R1-7B** [15], **Qwen2.5-7B-Instruct** [44], **GPT-4.1**, **Claude-4-Sonnet**, all serving as the backbone LLMs within FinAgent framework.

Implementation Details. LLM training is performed on 16 NVIDIA A100 GPUs, while both LLMs inference and LLMs Agents experiments are conducted on 2 NVIDIA H100 GPUs, with evaluation results reported from a single run.

Experiment Results As shown in Figure 2, FinReasoner leads all four financial-reasoning benchmarks, outperforming every other LLM and underscoring the value of domain-specific reasoning over generic instruction tuning. In Figure 3, we rescale all metrics (with inverted MDD and VOL) to a 0–100 range for area-based comparison. FinReasoner achieves strong performance on all stocks, validating its effectiveness in both reasoning and trading.

7 Conclusion and Future Work

In this paper, we introduce FinWorld, a comprehensive framework for financial AI research and development. We provide a detailed description of the framework, including the data, model, training, evaluation, and task layers, along with empirical evaluations on four financial AI tasks: time series forecasting, algorithmic trading, portfolio management, and LLMs applications. The results demonstrate

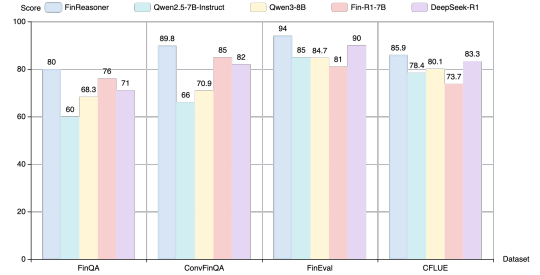


Figure 2: Comparison of LLMs on financial reasoning.

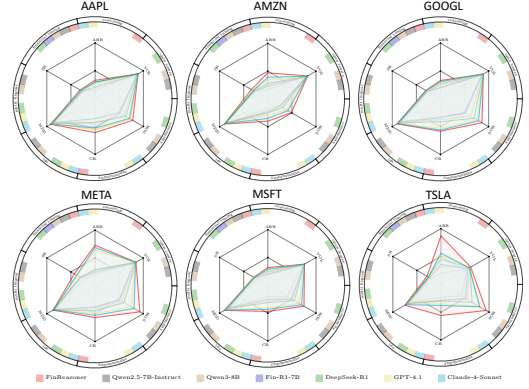


Figure 3: Comparison of LLMs on trading.

that FinWorld effectively supports financial AI research and development. Through its unified architecture supporting multiple AI paradigms (ML, DL, RL, LLMs, and LLM Agents) and comprehensive multimodal data integration, FinWorld addresses key limitations of existing platforms and provides a scalable foundation for both academic research and practical applications.

While FinWorld provides a comprehensive platform for financial AI research and deployment, several limitations should be noted. First, the current framework focuses on research and experimentation rather than live high-frequency trading systems that require sub-second latency, with real-time deployment validation remaining for further development. Second, while our experiments demonstrate consistent performance, the current evaluation does not yet cover all potential financial AI applications, particularly multilingual information retrieval, cross-market forecasting, and real-time decision pipelines. We are developing latency simulation modules and online integration interfaces to support low-latency financial applications, including minute-level trading LLM Agent evaluation. Additionally, we plan to develop quality assurance frameworks and statistical testing interfaces to enable rigorous model comparison and bias evaluation across markets.

8 Acknowledgments

This research is supported by the Joint NTU-WeBank Research Centre on Fintech, Nanyang Technological University, Singapore. This research is also supported by Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (No. MSS24C005).

References

- [1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as I Can, not as I Say: Grounding Language in Robotic Affordances. *arXiv preprint arXiv:2204.01691* (2022).
- [2] Dogu Araci. 2019. FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. *arXiv preprint arXiv:1908.10063* (2019).
- [3] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 1976. Time Series Analysis: Forecasting and Control. *San Francisco: Holden-Day* (1976).
- [4] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), 785–794.
- [5] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, et al. 2021. FinQA: A Dataset of Numerical Reasoning over Financial Data. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021), 3697–3711.
- [6] Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022. ConVinqa: Exploring the chain of numerical reasoning in conversational finance question answering. *arXiv preprint arXiv:2210.03849* (2022).
- [7] MMEngine Contributors. 2022. MMEngine: OpenMMLab Foundational Library for Training Deep Learning Models. <https://github.com/open-mmlab/mengine>. (2022).
- [8] Qianggang Ding, Sifan Wu, Hao Sun, Jiadong Guo, and Jian Guo. 2020. Hierarchical Multi-scale Gaussian Transformer for Stock Movement Prediction. In *IJCAI*. 4640–4646.
- [9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xi-aohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929* (2020).
- [10] Yitong Duan, Lei Wang, Qizhong Zhang, and Jian Li. 2022. FactorVAE: A Probabilistic Dynamic Factor Model Based on Variational Autoencoder for Predicting Cross-sectional Stock Returns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 4468–4476.
- [11] Robert F Engle. 1982. Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica: Journal of the Econometric Society* (1982), 987–1007.
- [12] Eugene F Fama and Kenneth R French. 2015. A Five-factor Asset Pricing Model. *Journal of Financial Economics* 116, 1 (2015), 1–22.
- [13] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group Policy Optimization for LLM Agent Training. *arXiv preprint arXiv:2505.10978* (2025).
- [14] Shibo Feng, Chunyan Miao, Ke Xu, Jiaxiang Wu, Pengcheng Wu, Yang Zhang, and Peilin Zhao. 2023. Multi-scale Attention Flow for Probabilistic Time Series Forecasting. *IEEE Transactions on Knowledge and Data Engineering* 36, 5 (2023), 2056–2068.
- [15] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948* (2025).
- [16] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft Actor-Critic Algorithms and Applications. *arXiv preprint arXiv:1812.05905* (2018).
- [17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked Autoencoders are Scalable Vision Learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16000–16009.
- [18] J Heaton, N Polson, and J Witte. 2017. Deep Portfolio Theory. *Journal of Financial Data Science* 1, 1 (2017), 1–28.
- [19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.
- [20] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. 2017. A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem. *arXiv preprint arXiv:1706.10059* (2017).
- [21] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems* 30 (2017).
- [22] Zixuan Ke, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. 2025. Demystifying Domain-adaptive Post-training for Financial LLMs. *arXiv preprint arXiv:2501.04961* (2025).
- [23] Juhyeong Kim. 2025. Semi-Decision-Focused Learning with Deep Ensembles: A Practical Framework for Robust Portfolio Optimization. *arXiv preprint arXiv:2503.13544* (2025).
- [24] Diederik P Kingma, Max Welling, et al. 2013. Auto-encoding Variational Bayes.
- [25] Xiao-Yang Liu, Guoxuan Wang, Hongyang Yang, and Daochen Zha. 2023. FinGPT: Democratizing Internet-scale Data for Financial Large Language Models. *arXiv preprint arXiv:2307.10485* (2023).
- [26] 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 (2022), 1835–1849.
- [27] Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. 2020. FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance. *arXiv preprint arXiv:2011.09607* (2020).
- [28] Yang Liu, Qi Liu, Hongke Zhao, Zhen Pan, and Chuanren Liu. 2020. Adaptive Quantitative Trading: An Imitative Deep Reinforcement Learning Approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2128–2135.
- [29] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2024. TimeXer: Empowering Transformers for Time Series Forecasting with Exogenous Variables. *arXiv preprint arXiv:2401.13795* (2024).
- [30] Zhaowei Liu, Xin Guo, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Zixuan Wang, Jiajie Xu, Weige Cai, Ziwei Yang, Xueqian Zhao, et al. 2025. Fin-r1: A Large Language Model for Financial Reasoning Through Reinforcement Learning. *arXiv preprint arXiv:2503.16252* (2025).
- [31] Harry Markowitz. 1952. Portfolio Selection. *The Journal of Finance* 7, 1 (1952), 77–91.
- [32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level Control Through Deep Reinforcement Learning. *nature* 518, 7540 (2015), 529–533.
- [33] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *arXiv preprint arXiv:2211.14730* (2022).
- [34] Jie Ouyang, Ruiran Yan, Yucong Luo, Mingyue Cheng, Qi Liu, Zirui Liu, Shuo Yu, and Daoyu Wang. 2025. Training Powerful LLM Agents with End-to-End Reinforcement Learning. <https://github.com/0russwest0/Agent-R1>
- [35] Lingfei Qian, Weipeng Zhou, Yan Wang, Xueqing Peng, Han Yi, Yilun Zhao, Jimin Huang, Qianqian Xie, and Jian-yun Nie. 2025. Fino1: On the Transferability of Reasoning-Enhanced LLMs and Reinforcement Learning to Finance. *arXiv preprint arXiv:2502.08127* (2025).
- [36] Molei Qin, Shuo Sun, Wentao Zhang, Haochong Xia, Xinrun Wang, and Bo An. 2024. Earnhft: Efficient Hierarchical Reinforcement Learning for High Frequency Trading. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 14669–14676.
- [37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).
- [38] Sameena Shah, Zhiyu Chen, Charese Smiley, Shubhra De, Xiuxi Wang, Ming Qin, Yiming Zheng, Pranjal Arora, Hongxia Liu, et al. 2022. FLANG: Financial Language Model for Domain-Specific Tasks. *arXiv preprint arXiv:2205.06031* (2022).
- [39] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300* (2024).
- [40] 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 (2023), 59047–59061.
- [41] Alpaca Team. 2025. Alpaca Markets. <https://alpaca.markets/>.
- [42] Financial Modeling Prep Team. 2025. Financial Modeling Prep. <https://site.financialmodelingprep.com/>.
- [43] Pyecharts Team. 2025. Pyecharts: Python Echarts Plotting Library. <https://github.com/pyecharts/pyecharts> Version 2.0.8.
- [44] Qwen Team. 2025. Qwen2.5 Technical Report. arXiv:2412.15115 [cs.CL] <https://arxiv.org/abs/2412.15115>
- [45] TSLib Team. 2025. Time Series Library. <https://github.com/thuml/Time-Series-Library>
- [46] Verl Team. 2025. Volcano Engine Reinforcement Learning for LLMs. <https://github.com/volcengine/verl>
- [47] Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural Discrete Representation Learning. *Advances in Neural Information Processing Systems* 30 (2017).
- [48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems* 30 (2017).
- [49] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291* (2023).
- [50] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. 2024. Timemixer: Decomposable Multiscale Mixing for Time Series Forecasting. *arXiv preprint arXiv:2405.14616* (2024).
- [51] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qi, Jianmin Wang, and Mingsheng Long. 2024. TimeXer: Empowering Transformers for Time Series Forecasting with Exogenous Variables. *Advances in Neural Information Processing Systems* 37 (2024), 469–498.

- [52] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 2025. RAGEN: Understanding Self-Evolution in LLM Agents via Multi-Turn Reinforcement Learning. *arXiv:2504.20073 [cs.LG]* <https://arxiv.org/abs/2504.20073>
- [53] Zikai Wei, Anyi Rao, Bo Dai, and Dahua Lin. 2023. HireVAE: An Online and Adaptive Factor Model Based on Hierarchical and Regime-Switch VAE. *arXiv preprint arXiv:2306.02848* (2023).
- [54] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. 2022. Etsformer: Exponential Smoothing Transformers for Time-series Forecasting. *arXiv preprint arXiv:2202.01381* (2022).
- [55] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. Timesnet: Temporal 2d-variation Modeling for General Time Series Analysis. *arXiv preprint arXiv:2210.02186* (2022).
- [56] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition Transformers with Auto-correlation for Long-term Series Forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 22419–22430.
- [57] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhjyoti Kambhadi, David Rosenberg, and Gideon Mann. 2023. BloombergGPT: A Large Language Model for Finance. *arXiv preprint arXiv:2303.17564* (2023).
- [58] Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. 2024. TradingAgents: Multi-agents LLM Financial Trading Framework. *arXiv preprint arXiv:2412.20138* (2024).
- [59] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388* (2025).
- [60] 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* (2020).
- [61] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.
- [62] Pengqian Yu, Joon Sern Lee, Ilya Kulyatin, Zekun Shi, and Sakyasingha Dasgupta. 2019. Model-based Deep Reinforcement Learning for Dynamic Portfolio Optimization. *arXiv preprint arXiv:1901.08740* (2019).
- [63] Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. 2025. DAPO: An Open-source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476* (2025).
- [64] Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W Suchow, and Khaldoun Khashanah. 2024. Finmem: A Performance-enhanced LLM Trading Agent with Layered Memory and Character Design. In *Proceedings of the AAAI Symposium Series*, Vol. 3. 595–597.
- [65] Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan Suchow, Zhenyu Cui, Rong Liu, et al. 2024. Fincon: A Synthesized LLM Multi-Agent System with Conceptual Verbal Reinforcement for Enhanced Financial Decision Making. *Advances in Neural Information Processing Systems* 37 (2024), 137010–137045.
- [66] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 9 (2023), 11121–11128.
- [67] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting?. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11121–11128.
- [68] Liwen Zhang, Weige Cai, Zhaowei Liu, Zhi Yang, Wei Dai, Yujie Liao, Qianru Qin, Yifei Li, Xingyu Liu, Zhiqiang Liu, et al. 2023. Fineval: A Chinese Financial Domain Knowledge Evaluation Benchmark for Large Language Models. *arXiv preprint arXiv:2308.09975* (2023).
- [69] Wentao Zhang, Ce Cui, Yilei Zhao, Yang Liu, and Bo An. 2025. AgentOrchestra: A Hierarchical Multi-Agent Framework for General-Purpose Task Solving. *arXiv preprint arXiv:2506.12508* (2025).
- [70] Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, et al. 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*. 4314–4325.
- [71] Wentao Zhang, Yilei Zhao, Shuo Sun, Jie Ying, Yonggang Xie, Zitao Song, Xinrun Wang, and Bo An. 2024. Reinforcement Learning with Maskable Stock Representation for Portfolio Management in Customizable Stock Pools. In *Proceedings of the ACM Web Conference 2024*. 187–198.
- [72] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer Utilizing Cross-dimension Dependency for Multivariate Time Series Forecasting. In *The Eleventh International Conference on Learning Representations*.
- [73] Yilei Zhao, Wentao Zhang, Tingran Yang, Yong Jiang, Fei Huang, and Wei Yang Bryan Lim. 2024. STORM: A Spatio-Temporal Factor Model Based on Dual Vector Quantized Variational Autoencoders for Financial Trading. *arXiv preprint arXiv:2412.09468* (2024).
- [74] Tianyu Zhou, Pinqiao Wang, Yilin Wu, and Hongyang Yang. 2024. Finrobot: AI Agent for Equity Research and Valuation with Large Language Models. *arXiv preprint arXiv:2411.08804* (2024).
- [75] Jie Zhu, Qian Chen, Huaixia Dou, Junhui Li, Lifan Guo, Feng Chen, and Chi Zhang. 2025. Dianjin-r1: Evaluating and enhancing financial reasoning in large language models. *arXiv preprint arXiv:2504.15716* (2025).
- [76] Jie Zhu, Junhui Li, Yalong Wen, and Lifan Guo. 2024. Benchmarking Large Language Models on CFLUE - A Chinese Financial Language Understanding Evaluation Dataset. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL-2024)*.

Appendix

A FinWorld as a Dataset Platform

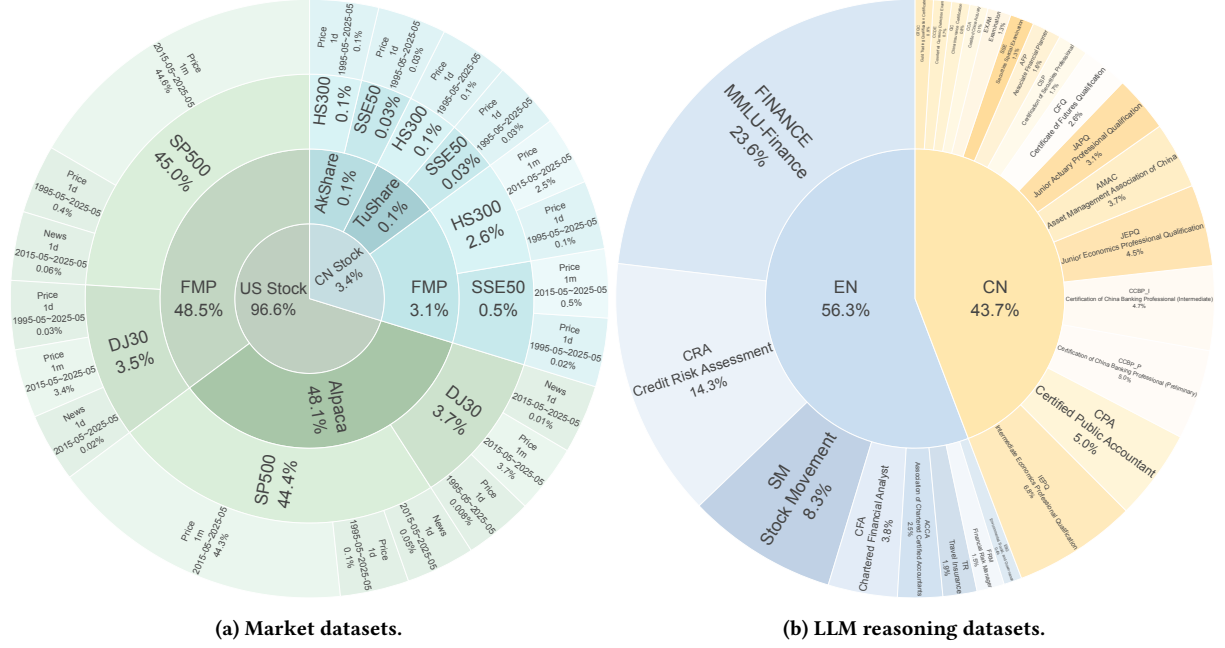


Figure 4: Overview of dataset.

A.1 Market Dataset

We select two representative markets for comprehensive financial AI research: the US market, representing developed markets with high liquidity and extensive data availability, and the Chinese market, representing emerging markets with unique characteristics and regulatory environments. For the US stock pools, we choose the Dow Jones Industrial Average (DJ30) and S&P 500 (SP500), which are widely recognized benchmarks covering large-cap and blue-chip stocks. For the Chinese stock pools, we select the SSE 50 Index (SSE50) and CSI 300 Index (HS300), representing the most liquid and representative stocks in the Shanghai and Shenzhen markets.

Our dataset integrates multiple data sources to ensure comprehensive coverage and data quality. We utilize four primary data providers: Financial Modeling Prep (FMP), Alpaca Markets, AkShare, and TuShare. These sources provide daily and minute-level price data, along with news data from various financial platforms including Seeking Alpha, Bloomberg, and other major financial news outlets. The dataset spans from 1995 to 2025, offering extensive historical coverage for robust model training and evaluation.

The data processing pipeline includes comprehensive preprocessing steps to ensure data quality and usability. For price data, we implement filtering procedures to remove non-trading days and invalid data points, followed by the calculation of technical indicators using the Alpha158 [60] (Table 5) framework, which provides a solid foundation for deep learning modeling. For news text data, we employ a two-stage processing approach: first, we use web scraping scripts to extract raw content from original news links, then we utilize Qwen3-32B to generate concise summaries, enabling efficient text analysis and feature extraction for financial AI applications.

After processing all stock data, we obtain over 800 million valid data points, providing comprehensive support for time series forecasting, algorithmic trading, portfolio management, and LLM applications. Furthermore, we provide unified download and data processing interfaces that theoretically enable unlimited data expansion by simply specifying stock pools and date ranges. Processed data is uniformly uploaded to Hugging Face for storage, and we offer unified dataset encapsulation for both single-stock and multi-stock scenarios, allowing flexible loading of different data types. This constitutes an efficient one-stop framework that integrates data downloading, processing, storage, and retrieval capabilities.

A.2 LLM Reasoning Dataset

Our LLM reasoning dataset collection process is designed to provide comprehensive coverage of financial reasoning capabilities, drawing inspiration from established datasets such as FinQA [5], FinEval [68], CFLUE [76], Salesforce FinEval [22], and ConvFinQA [6]. We systematically collect diverse types of financial reasoning data through multiple channels and sources, ensuring broad coverage of financial knowledge domains and reasoning patterns.

Table 5: Alpha158 Technical Indicators Overview

Indicator	Formula	Description
Kline-based Indicators		
kmid	$(close - open) / close$	K-Mid price ratio
kmid2	$(close - open) / (high - low)$	Normalized K-Mid price ratio
klen	$(high - low) / open$	K-Length price ratio
kup	$(high - \max(open, close)) / open$	K-Up price ratio
kup2	$(high - \max(open, close)) / (high - low)$	Normalized K-Up price ratio
klow	$(\min(open, close) - low) / open$	K-Low price ratio
klow2	$(\min(open, close) - low) / (high - low)$	Normalized K-Low price ratio
ksft	$(2 \times close - high - low) / open$	K-Shift price ratio
ksft2	$(2 \times close - high - low) / (high - low)$	Normalized K-Shift price ratio
Price-based Rolling Indicators (for each window w)		
roc_w	$close.shift(w) / close$	Rate of change over w periods
ma_w	$close.rolling(w).mean() / close$	Moving average ratio over w periods
std_w	$close.rolling(w).std() / close$	Standard deviation ratio over w periods
beta_w	$(close.shift(w) - close) / (w \times close)$	Beta coefficient over w periods
max_w	$close.rolling(w).max() / close$	Maximum price ratio over w periods
min_w	$close.rolling(w).min() / close$	Minimum price ratio over w periods
qtl_u_w	$(close - close.rolling(w).quantile(0.8)) / close$	Upper quantile ratio over w periods
qtl_d_w	$(close - close.rolling(w).quantile(0.2)) / close$	Lower quantile ratio over w periods
rank_w	$close.rolling(w).rank(pct = True).iloc[-1] / w$	Price rank percentile over w periods
Position-based Indicators (for each window w)		
imax_w	$high.rolling(w).argmax() / w$	Position of maximum high over w periods
imin_w	$low.rolling(w).argmin() / w$	Position of minimum low over w periods
imxd_w	$(high.rolling(w).argmax() - low.rolling(w).argmin()) / w$	Distance between max/min positions
RSV and Count Indicators (for each window w)		
rsv_w	$(close - \min(low, close.shift(w))) / (\max(high, close.shift(w)) - \min(low, close.shift(w)))$	RSV (Raw Stochastic Value)
cntp_w	$ret1.gt(0).rolling(w).sum() / w$	Count of positive returns over w periods
cntn_w	$ret1.lt(0).rolling(w).sum() / w$	Count of negative returns over w periods
cntd_w	$cntp_w - cntn_w$	Difference between positive and negative return counts
Correlation Indicators (for each window w)		
corr_w	$close.rolling(w).corr(log(volume + 1).rolling(w))$	Correlation between close and log volume
cord_w	$(close / close.shift(1)).rolling(w).corr(log(volume / volume.shift(1) + 1).rolling(w))$	Normalized correlation
Sum-based Indicators (for each window w)		
sump_w	$pos_ret1.rolling(w).sum() / abs_ret1.rolling(w).sum()$	Sum of positive returns ratio
sumn_w	$1 - sump_w$	Sum of negative returns ratio
sumd_w	$2 \times sump_w - 1$	Difference between positive and negative return sums
Volume-based Indicators (for each window w)		
vma_w	$volume.rolling(w).mean() / volume$	Volume moving average ratio
vstd_w	$volume.rolling(w).std() / volume$	Volume standard deviation ratio
wvma_w	$ close / close.shift(1) - 1 .rolling(w).std() / close / close.shift(1) - 1 .rolling(w).mean()$	Weighted volume moving average
vsump_w	$pos_vchg1.rolling(w).sum() / abs_vchg1.rolling(w).sum()$	Sum of positive volume changes ratio
vsumn_w	$1 - vsump_w$	Sum of negative volume changes ratio
vsumd_w	$2 \times vsump_w - 1$	Difference between positive and negative volume change sums
Volume Logarithmic Indicator		
logvol	$\log(volume + 1)$	Logarithm of volume

The dataset encompasses both Chinese and English content, covering diverse financial scenarios including financial question-answering data, industry-specific financial corpus, professional certification examination materials, business domain knowledge, tabular financial data, market analysis reports, multi-turn conversational interactions, and quantitative investment scenarios. Each scenario is carefully curated to reflect real-world financial applications and reasoning challenges, with particular emphasis on comprehensive coverage of financial industry qualification examination data.

In total, we have collected over 80k samples including both training and testing data, providing comprehensive support for RL-based LLM reasoning training and evaluation across diverse financial scenarios and professional domains.

B FinWorld as a Visualization Toolkit

Effective visualization is crucial for understanding complex financial AI systems, encompassing datasets, algorithms, models, performance metrics, experimental results and so on. To support comprehensive research and analysis across the entire financial AI workflow, we have developed a comprehensive suite of visualization tools that provide intuitive and interactive representations of our platform’s capabilities. These tools enable researchers to quickly grasp the scope and characteristics of our datasets, understand algorithm performance and model behaviors, analyze training dynamics and convergence patterns, compare different approaches across multiple metrics, and visualize complex financial decision-making processes. The visualization framework facilitates informed decision-making in model selection, hyperparameter tuning, performance evaluation, and research design across all four core financial AI tasks. The visualization tools are implemented using Python, LaTeX, Pyecharts [43], and other visualization libraries.

B.1 Dataset Visualization

We develop a set of visualization tools to help researchers explore and analyze our financial datasets. These tools are designed to provide intuitive and interactive visual representations of dataset composition, distribution, and characteristics, enabling researchers to quickly understand the scope and diversity of available financial data.

Dataset Statistics Tools. As illustrated in Figure 4, we employ hierarchical sunburst diagrams to comprehensively visualize the composition, distribution, and characteristics of our financial datasets. These visualizations provide an intuitive representation of data types, proportions, and hierarchical relationships across different market segments and financial instruments. These hierarchical visualizations serve as powerful tools for dataset exploration and analysis, providing researchers with immediate insights into data availability, distribution patterns, and potential applications. The interactive nature of these visualizations allows for dynamic exploration of different data segments, supporting both high-level overview and detailed examination of specific data categories. This comprehensive visualization framework enhances the usability and accessibility of our financial datasets, facilitating efficient research workflow and promoting transparent data understanding across the financial AI community.



Figure 5: Overview of double modes of Kline Chart.

Kline Chart Tools. To effectively visualize financial data, we employ two distinct Kline Chart display modes tailored to different analysis requirements. The first mode is designed for long-term datasets where detailed granularity is not essential, as illustrated in Figure 5a, which displays simplified Kline charts with basic moving averages (MA5, MA10, MA20) and trading volume. This streamlined visualization enables researchers to quickly grasp overall market trends and price movements across extended time periods without being overwhelmed by excessive technical details. Additionally, our Kline Chart visualizations are automatically generated as interactive HTML files, enabling online browsing and interactive exploration of the visualization information through web-based interfaces.

The second mode is optimized for short-term data analysis requiring detailed technical indicators and trading signals, as shown in Figure 5b. This comprehensive display includes multiple moving averages (MA5, MA10, MA20), MACD (Moving Average Convergence Divergence) with its components (DIF and DEA), and provides real-time technical signals such as golden cross formations, death crosses, and other critical trading indicators. This detailed visualization mode supports precise technical analysis, enabling researchers to identify specific entry and exit points, analyze momentum shifts, and evaluate the effectiveness of various technical trading strategies. The dual-mode approach ensures that researchers can select the appropriate level of detail based on their specific analysis needs, from high-level trend analysis to granular technical signal detection.

Dataset Distribution Tools. To better understand and analyze the structure of our dataset, we employ a t-SNE-based visualization technique to represent the distribution of price and technical indicators in a low-dimensional space. This approach reveals intrinsic patterns and temporal dynamics embedded in the data. The visualization is conducted in two distinct modes: asset-level and pool-level.

Asset-level visualization focuses on a few selected individual stocks. For each stock, we sample its price at different time points, and then apply t-SNE to project these temporal snapshots into a two-dimensional space. This mode allows us to observe how the price of a single asset evolve over time, and to compare such temporal trajectories across different individual assets.

Pool-level visualization, on the other hand, examines a large number of stocks within different asset pools (e.g., sectors or universes) at the same or selected time points. To construct these visualizations, we compute a comprehensive set of technical indicators using the Alpha158 feature set. The high-dimensional feature representations are then transformed via t-SNE into a more interpretable 2D layout, facilitating insight into the temporal and cross-sectional structure of the data. Here, we sample many stocks from each pool at specific times, and use t-SNE to project their high-dimensional features into two dimensions. This mode helps uncover broader distributional patterns across asset pools—such as clustering behavior, outlier structures, or inter-stock similarities within and across pools.

B.2 Performance Visualization

Trading Curve Tools. The proposed module replicates the trading-curve display of FinAgent [70] while focusing on a single, essential feature. It renders an interactive curve that tracks cumulative return and overlays each model decision with a clearly labelled BUY or

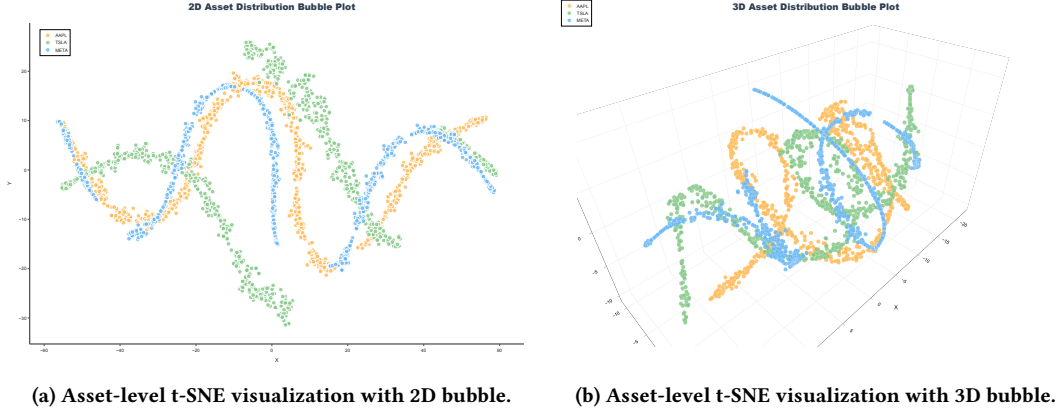


Figure 6: Overview of asset-level t-SNE visualization.

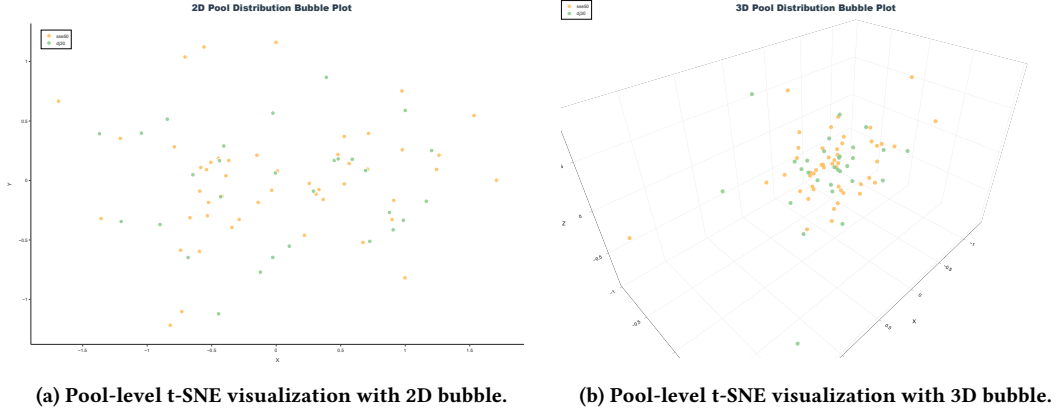


Figure 7: Overview of pool-level t-SNE visualization.

SELL marker. An accompanying block of interpretable text records the reasoning behind every trade, summarising the key signals and risk constraints that guided the action. This streamlined presentation allows researchers to relate model logic directly to observed performance, offering a transparent basis for auditing algorithmic behaviour and evaluating how trading decisions drive the evolution of returns over time.

Score Visualization Tools. As illustrated in Figure 2, the module renders an interactive bar chart that aggregates the performance of multiple large language models (LLMs) across a suite of reasoning benchmarks. Each bar encodes the absolute score of an individual model on a given task, while colour grouping facilitates quick visual association between benchmarks. By presenting all results in a single, normalised view, Score Visualization Tools makes it straightforward to identify relative strengths and weaknesses, rank models at a glance, and track progress as new architectures or training regimes are evaluated.

Trading Performance Visualization Tools. As illustrated in Figure 3, the module presents an interactive radar chart that overlays multi metrics (e.g., ARR, SR, MDD, CR, SoR, VOL) for every evaluated model. Each polygon traces the scores of one model, and the enclosed area provides an intuitive proxy for holistic performance: the larger the area, the better the overall trade-off between return and risk. The radial format allows direct, dimension-by-dimension comparison, making it straightforward to pinpoint where a model excels or lags relative to its peers. Although the example focuses on trading-specific indicators, the framework is extensible; additional axes can represent intrinsic LLMs attributes such as reasoning abilities or computational efficiency, thereby enabling a unified visual assessment of both financial and cognitive capabilities.

C FinWorld as a Development Framework

Beyond the four core financial AI tasks (time series forecasting, algorithmic trading, portfolio management, and LLM applications) discussed in the main text, FinWorld is designed as a comprehensive and extensible development framework that supports unlimited expansion for



Figure 8: Overview of trading curve visualization.

diverse financial applications. The platform’s modular architecture and flexible design principles enable researchers and practitioners to easily implement and experiment with various financial AI methodologies beyond the standard tasks.

The framework supports additional financial AI tasks such as factor model, financial market representation learning, and other domain-specific applications. For factor model, FinWorld provides built-in support for traditional statistical factors, and deep learning factor extraction methods. The platform’s representation learning capabilities enable the development of sophisticated market embeddings that capture complex financial relationships and temporal dynamics.

The extensible nature of FinWorld is achieved through its layered architecture, where each component can be independently modified or extended without affecting other parts of the system. This design allows for seamless integration of new algorithms, models, and tasks while maintaining consistency with the existing framework. Researchers can leverage the platform’s unified data interfaces, standardized evaluation metrics, and comprehensive visualization tools to develop and validate novel financial AI approaches.

C.1 Factor Model

FinWorld provides comprehensive support for factor model, encompassing both traditional statistical approaches and advanced deep learning methodologies. For traditional statistical factors, we primarily utilize the Alpha158. For deep learning dynamic factor models, we implement state-of-the-art approaches including FactorVAE [10] and HireVAE [53]. FactorVAE leverages variational autoencoders to learn latent factor representations that capture complex market dynamics and temporal dependencies. HireVAE extends this framework with hierarchical structures to model multi-scale temporal patterns in financial data. Notably, the factor model STORM [73] is developed based on this framework, demonstrating the platform’s capability to support cutting-edge research and development in financial AI.

The factor model framework in FinWorld is designed with modularity and extensibility in mind, allowing researchers to easily implement and experiment with new factor extraction methods. The platform provides standardized interfaces for factor computation, evaluation, and backtesting, ensuring consistent and reproducible results across different factor models and datasets.

C.2 Financial Market Representation Learning

FinWorld implements comprehensive financial market representation learning capabilities, focusing on stock pool-based representation learning approaches. The platform supports various state-of-the-art representation learning methods, including VAE (Variational Autoencoder) [24], MAE (Masked Autoencoder) [17], and VQVAE (Vector Quantized Variational Autoencoder) [47] for capturing complex market dynamics and relationships. These representation learning methods enable the development of sophisticated market embeddings that capture both local and global patterns in financial data: VAE captures smooth state transitions via continuous latents, MAE learns robust features through masked reconstruction, and VQVAE models discrete regimes with categorical representations.

The representation learning framework in FinWorld is designed to work seamlessly with different stock pools and market segments, allowing researchers to develop market-specific embeddings that capture the unique characteristics of different financial instruments and market conditions. This capability is particularly valuable for understanding cross-asset relationships, market regime detection, and developing more sophisticated financial AI models.

D Implementation Details

D.1 Time Series Forecasting

Dataset Setup. We focus on daily-level financial data for time series forecasting. We select DJ30 and SP500 from the U.S. market and SSE50 and HS300 from the Chinese market, covering both daily OHLCV data with Alpha158 technical indicators for the period 2015-05-01 to 2025-05-01, sourced from FMP. We employ a StandardScaler for feature normalization and utilize both dense features (145 dimensions from Alpha158 indicators) and sparse temporal features (4 dimensions: days, months, weekdays, years). The data is split at 2023-05-01 for training and validation separation. All features are standardized for each stock individually to ensure comparability across assets and time periods.

Metrics. We evaluate forecasting performance using four metrics: i) **Mean Absolute Error (MAE)**, which measures point prediction accuracy by averaging the absolute differences between predicted and true values; ii) **Mean Squared Error (MSE)**, which computes the average of the squared differences between predictions and actual outcomes; iii) **Rank Correlation Coefficient (RankIC)**, which assesses the correlation between the predicted and actual rankings using Spearman’s rank correlation; and iv) **RankIC Information Ratio (RankICIR)**, which evaluates the stability of ranking performance over time by dividing the average RankIC by its standard deviation. MAE and MSE focus on absolute prediction accuracy, while RankIC and RankICIR are particularly relevant for evaluating the model’s effectiveness in capturing return-based or ranking-based financial relationships. Detailed information can be found in Table 6.

Table 6: Forecasting Metrics Overview

Name	Direction	Formula	Description
MAE	↓	$MAE = \frac{1}{TN} \sum_{t=1}^T \sum_{i=1}^N \hat{y}_{i,t} - y_{i,t} $	Mean Absolute Error; average absolute deviation over all assets and time steps.
MSE	↓	$MSE = \frac{1}{TN} \sum_{t=1}^T \sum_{i=1}^N (\hat{y}_{i,t} - y_{i,t})^2$	Mean Squared Error; penalizes large prediction errors more heavily.
RankIC _t	↑	$RankIC_t = \frac{\sum_{i=1}^N (R(\hat{y}_{i,t}) - \bar{R}_{\hat{y},t}) (R(y_{i,t}) - \bar{R}_{y,t})}{\sqrt{\sum_{i=1}^N (R(\hat{y}_{i,t}) - \bar{R}_{\hat{y},t})^2} \sqrt{\sum_{i=1}^N (R(y_{i,t}) - \bar{R}_{y,t})^2}}$	Spearman rank correlation computed across assets at each time step t .
RankIC	↑	$RankIC = \frac{1}{T} \sum_{t=1}^T RankIC_t$	Average cross-sectional Spearman correlation over all time steps.
RankICIR	↑	$RankICIR = \frac{\frac{1}{T} \sum_{t=1}^T RankIC_t}{\sqrt{\frac{1}{T-1} \sum_{t=1}^T (RankIC_t - \bar{\rho})^2}}$	Rank Information Coefficient Information Ratio; mean RankIC divided by its standard deviation over time, with $\bar{\rho}$ being the mean RankIC.

Methods. We evaluate several state-of-the-art ML-based and DL-based time series forecasting models, including: i) ML-based methods: **LightGBM** [60] and **XGBoost** [60]; ii) DL-based methods: **Autoformer** [56], **Crossformer** [72], **ETSformer** [54], **DLinear** [67], **TimesNet** [55], **PatchTST** [33], **TimeMixer** [50], and **TimeXer** [51]. These models are selected based on their effectiveness in capturing complex financial time series patterns and their ability to handle high-dimensional data.

Implementation Details. All experiments are conducted on 2 NVIDIA H100 GPUs. Following the implementation practices of Qlib [60] and TSLib [45], we adopt consistent architectural configurations across different models while preserving their unique characteristics. For dataset, we utilize a multi-GPU parallel dataloader with a batch size of 64. Each batch contains sequences of 64 historical days and 32 future days, meaning the model observes the past 64 days and predicts the returns over the next 32 days. For network architecture, we use a standardized setup with encoder depth of 4 and decoder depth of 4, where the decoder typically consists of a single MLP layer for output mapping unless the model has specific decoder requirements. The embedding dimension is set to 128 across all models. For optimization, we employ AdamW optimizer with a learning rate of 1e-5 and implement a warmup scheduler for learning rate adjustment, with maximum training epochs of 2,000 and warmup epochs of 200. The loss function is Mean Squared Error (MSE). Additional model-specific configurations are applied based on each model’s architectural requirements and design principles. The reported results are averaged over three runs with different random seeds.

D.2 Algorithmic Trading

Dataset Setup. We focus on daily-level financial data for trading. We select AAPL, AMZN, GOOGL, META, MSFT, and TSLA from the U.S. market, covering both daily OHLCV data with Alpha158 technical indicators for the period from 1995-05-01 to 2025-05-01, sourced from FMP. We employ a StandardScaler for feature normalization and utilize both dense features (145 dimensions from Alpha158 indicators) and sparse temporal features (4 dimensions: days, months, weekdays, years). The data is split at 2023-05-01 for training and validation separation. All features are standardized for each stock individually to ensure comparability across assets and time periods.

Metrics. We evaluate algorithmic trading performance using six financial metrics: i) **Annual Rate of Return (ARR)**, which measures the annualized profitability of a strategy based on the change in portfolio value over time, adjusted by an annualization factor (e.g., 252 for daily trading); ii) **Sharpe Ratio (SR)**, which quantifies risk-adjusted return by comparing the average return to its standard deviation; iii) **Maximum Drawdown (MDD)**, which measures the largest peak-to-trough decline in the cumulative return, indicating the worst observed loss during a period; iv) **Calmar Ratio (CR)**, which evaluates the return-to-risk tradeoff by comparing average return to maximum drawdown; v) **Sortino Ratio (SoR)**, which is similar to the Sharpe Ratio but considers only downside volatility, thus emphasizing negative return risks. vi) **Volatility (VOL)**, which captures the standard deviation of returns and reflects the level of return fluctuation over time.

ARR reflects pure profitability, SR, CR, and SoR assess performance adjusted for different aspects of risk, and MDD and VOL evaluate risk exposure. Together, these metrics offer a comprehensive assessment of trading strategy effectiveness from both return and risk perspectives.

Table 7: Trading Metrics Overview

Name	Direction	Formula	Description
ARR	↑	$ARR = \left(\prod_{t=1}^T (1 + r_t) \right)^{\frac{1}{T}} - 1$	Annualized Rate of Return; compounded growth rate over time, using N periods per year and return series rets.
SR	↑	$SR = \frac{\mathbb{E}[\text{rets}] - r_f}{\text{Std}(\text{rets})} \times \sqrt{N}$	Sharpe Ratio; annualized excess return per unit of total volatility, computed from return series rets.
MDD	↓	$MDD = \max_{t \in [1, T]} \left(\frac{\max_{s \in [1, t]} V_s - V_t}{\max_{s \in [1, t]} V_s} \right)$	Maximum Drawdown; largest observed decline from peak to trough in cumulative portfolio value derived from rets.
CR	↑	$CR = \frac{ARR(\text{rets})}{ MDD(\text{rets}) }$	Calmar Ratio; annualized return divided by absolute maximum drawdown, both computed from rets.
SoR	↑	$SoR = \frac{\mathbb{E}[\text{rets}] - r_f}{DD(\text{rets})} \times \sqrt{N}$	Sortino Ratio; risk-adjusted return using downside deviation, penalizing only negative returns in rets.
VOL	↓	$VOL = \text{Std}(\text{rets}) \times \sqrt{N}$	Volatility; annualized standard deviation of return series rets, capturing total return variability.
DD	↓	$DD = \sqrt{\frac{1}{T} \sum_{t=1}^T \min(r_t - r_f, 0)^2} \times \sqrt{N}$	Downside Deviation; annualized standard deviation of returns below the risk-free rate, focusing on downside risk.

Note: rets = $[r_1, r_2, \dots, r_T]$ is the return series, where r_t is the return on day t ; N is the number of periods per year; r_f is the risk-free rate.

Methods. We systematically evaluate several Rule-based, ML-based, DL-based, and RL-based methods, including i) Rule-based methods: **BUY&HOLD**, **MACD**; ii) ML-based Methods: **LightGBM**, **XGBoost**; iii) DL-based Methods: **Transformer** [48], **LSTM**, **DLinear** [67]; iv) RL-based Methods: **PPO** [37], **SAC** [16]. These models represent a diverse set of approaches commonly used in quantitative finance research and practice.

Implementation Details. All experiments are conducted on 2 NVIDIA H100 GPUs. For all DL-based methods, we use similar dataset splits, network architectures, and optimizer configurations as in the forecasting task. For RL-based methods, the initial cash in the environment is set to 1×10^5 , and the transaction fee rate is 1×10^{-4} . We use 4 parallel environments for data sampling, with a sampling step size of 512. The policy mini-batch size is set to 128 and the value mini-batch size to 16. The optimizer used is AdamW, with the policy network learning rate set to 1×10^{-5} and the value network learning rate set to 1×10^{-6} . The maximum number of training steps is 1×10^8 , and the learning rate warmup is applied starting from step 0. Additional model-specific configurations are applied based on each model’s architectural requirements and design principles. The reported results are averaged over three runs with different random seeds.

D.3 Portfolio Management

Dataset Setup. The dataset setup is the same as in the time series forecasting task.

Metrics. The evaluation metrics for portfolio management are consistent with those used in the algorithmic trading task. Specifically, we report **ARR**, **SR**, **MDD**, **CR**, **SoR**, and **VOL**, following the definitions and formulas described in the previous section.

Methods. We systematically evaluate several ML-based, DL-based, and RL-based methods. Since we can apply the top- k dropout strategy [60] after any ML&DL-based forecasting model to construct a portfolio, our ML&DL-based methods include all ML&DL-based forecasting models. Specifically, we consider: i) Rule-based methods: **BUY&HOLD**; ii) ML-based Methods: **LightGBM**, **XGBoost**; iii) DL-based Methods: **Autoformer** [56], **Crossformer** [72], **ETSformer** [54], **DLinear** [67], **TimesNet** [55], **PatchTST** [33], **TimeMixer** [50], and **TimeXer** [51]; iv) RL-based Methods: **PPO** [37], **SAC** [16]. These models represent a diverse set of approaches commonly used in quantitative finance research and practice.

Implementation Details. All experiments are conducted on 2 NVIDIA H100 GPUs. For all DL-based methods, we use similar dataset splits, network architectures, and optimizer configurations as in the forecasting task. In addition, all ML&DL-based methods use the Top- k Dropout strategy [60] for portfolio construction. For RL-based methods, the initial cash in the environment is set to 1×10^5 , and the transaction fee rate is 1×10^{-4} . We use 4 parallel environments for data sampling, with a sampling step size of 512. The policy mini-batch size is set to 128 and the value mini-batch size to 16. The optimizer used is AdamW, with the policy network learning rate set to 1×10^{-5} and the value network learning rate set to 1×10^{-6} . The maximum number of training steps is 1×10^8 , and the learning rate warmup is applied starting from step 0. Additional model-specific configurations are applied based on each model’s architectural requirements and design principles. The reported results are averaged over three runs with different random seeds.

D.4 LLM Applications

Given that LLM Applications encompass both the training and deployment of LLMs as well as LLM Agents, we have allocated a dedicated chapter to this subject. Further details can be found in Appendix E.

E LLM Applications

In this section, we provide a comprehensive overview of the development background of both LLMs and LLM agents in financial AI. We discuss the historical context, the key advancements that have shaped their evolution, and the main challenges encountered along the way. Furthermore, we elaborate on several specific implementation details within our own work, illustrating how these developments have informed and influenced our approach. This in-depth analysis aims to offer readers a deeper understanding of the foundations and practical considerations involved in working with LLMs and LLM agents.

E.1 Background

LLMs for Financial AI. The integration of LLMs into financial decision-making processes has fundamentally transformed the landscape of financial AI, giving rise to two predominant developmental trajectories. The first, traditional paradigm leverages large-scale pre-training on generic corpora, followed by supervised fine-tuning (SFT) on domain-specific datasets, a process that has enabled the creation of financial-domain LLMs such as FinBERT [2], FLANG [38], and FinGPT [25]. These models excel at classic financial NLP tasks including sentiment analysis, news classification, and financial document understanding, benefiting from the inclusion of specialized financial terminology and context during pre-training. For example, BloombergGPT [57] introduced domain-adapted tokenization and large-scale financial corpus curation, enabling robust out-of-domain generalization while maintaining strong in-domain accuracy on a wide range of financial tasks. Similarly, FinQA [5] targets numerical reasoning within financial texts, highlighting the importance of reasoning abilities in addition to conventional text understanding.

The second, more recent development path centers on enhancing the reasoning capabilities of LLMs using RL, inspired by breakthroughs such as DeepSeek-R1 [15], which demonstrated substantial improvements in complex reasoning and decision-making tasks. In the context of financial AI, RL-based approaches have facilitated the creation of models capable of more sophisticated financial reasoning, including complex question answering (QA) and multi-step analysis over noisy and unstructured market data. State-of-the-art systems such as Fin-R1 [30], Fin-o1 [35], and Dianjin-R1 [75] typically adopt a two-stage pipeline: they first apply SFT on curated financial QA datasets to capture domain knowledge, then employ RL to further optimize the models' reasoning and generalization abilities with task-specific reward functions on challenging financial benchmarks. Notably, this line of work has shown that RL fine-tuning can help mitigate some limitations of SFT-only models, such as shallow pattern matching or overfitting to specific data distributions. However, it is also evident that LLM-based methods remain challenged by issues such as the absence of explicit sequential decision-making mechanisms, high computational overhead (particularly with RL-based training), and an inability to robustly handle the non-stationarity and regime shifts that typify real-world financial markets. Moreover, most current benchmarks still emphasize static QA or single-step predictions, rather than end-to-end financial decision-making in dynamic environments.

LLM Agents for Financial AI. To overcome the intrinsic limitations of static LLM-based models in handling sequential and interactive decision-making tasks, recent research has shifted towards constructing LLM-powered financial agents equipped with advanced agentic mechanisms. These mechanisms, such as external memory, dynamic tool use, self-reflection, and collaborative reasoning, enable LLMs to act not just as passive information processors, but as proactive, context-aware agents capable of adaptive and continual learning. Early efforts, including FinMem [64] and FinRobot [74], demonstrated that the addition of structured memory modules and chain-of-thought reasoning could substantially boost single-asset trading performance and stability, particularly in volatile or data-scarce settings. These systems typically maintain layered or episodic memories that allow agents to recall past market events, dynamically update strategies, and profile assets or markets over time.

Building on these foundations, more recent agentic systems such as FinAgent [70] have introduced multimodal reasoning capabilities, allowing LLMs to jointly process textual news, structured numerical data, and even K-line (candlestick) chart images via tool-augmented and dual-reflection architectures. This multimodal approach is shown to yield substantial profit gains and improved generalization across diverse datasets. Meanwhile, there has been a pronounced trend toward collaborative and hierarchical multi-agent systems, inspired by real-world financial organizations. For instance, FinCon [65] adopts a manager–analyst architecture, combining conceptual verbal reinforcement and episodic self-critique to facilitate robust sequential decision-making in both single-stock and multi-asset (portfolio) trading. Similarly, TradingAgents [58] simulate an entire trading firm with specialized agents (e.g., fundamental, technical, sentiment, and risk analysts), whose structured debate and coordination have been shown to improve both returns and risk control.

Recent advances have further explored the combination of RL with agentic LLMs to enable dynamic exploration and continual adaptation in non-stationary environments [13, 34]. For example, group-based RL finetuning enables multiple LLM agents to interact, learn, and adapt strategies in simulated market environments, fostering resilience against sudden market changes. Beyond performance, there is an increasing focus on interpretability and safety in financial LLM agents, with research exploring techniques such as role assignment, self-consistency checking, and real-time human-in-the-loop supervision. Despite these advances, key open challenges remain: efficiently scaling agentic LLMs, mitigating hallucinations and overfitting, improving sample efficiency in RL settings, and reliably deploying such systems in the face of adversarial and ever-evolving market conditions.

In summary, based on the development trajectory of LLMs and LLM agents described above, we can categorize the most popular LLM applications in the financial sector into four main types of tasks. Three of them require training, including **SFT for LLMs**, **RL for LLMs**, and **RL for LLM Agents**. The fourth type does not require training and mainly relies on prompt engineering for **LLM Agents** to perform downstream tasks. In FinWorld, the primary focus is on two types of tasks: **RL for LLMs** and **LLMs Agents**.

E.2 Reinforcement Learning for LLMs

The reinforcement learning implementation for LLMs in FinWorld follows a two-stage training paradigm designed to progressively develop both reasoning capabilities and decision-making skills in financial contexts.

Stage I: Financial Reasoning Fine-tuning. In the first stage, LLMs undergo reinforcement learning fine-tuning on conventional financial LLM reasoning datasets (refer to the LLM reasoning dataset section A.2). This stage focuses on developing fundamental financial reasoning abilities, including understanding financial concepts, analyzing market data, and performing quantitative calculations. The model learns to generate coherent and accurate responses to financial questions, interpret complex financial scenarios, and apply domain-specific knowledge. However, this stage alone is insufficient for real-world financial decision-making applications, as it lacks exposure to the dynamic and uncertain nature of actual market environments.

Stage II: Market Environment Learning. The second stage addresses this limitation by immersing the LLM in real market environments (refer to the market dataset section A.1) for exploration, trial-and-error learning, and decision-making skill development. In this stage, the model learns to make sequential decisions under uncertainty, adapt to changing market conditions, and optimize for long-term financial outcomes. The model interacts with simulated or real market data, receiving feedback on its decisions through reward signals that reflect financial performance metrics such as returns, risk-adjusted performance, and transaction costs. This stage enables the LLM to develop practical decision-making capabilities that go beyond theoretical reasoning, preparing it for deployment in actual financial applications. We primarily focus on the single asset trading task in this stage.

The two-stage approach ensures that LLMs first establish a solid foundation in financial reasoning before developing the practical skills necessary for effective decision-making in dynamic market environments. This progressive learning strategy enhances both the theoretical understanding and practical applicability of LLM-based financial AI systems.

Stage I: Financial Reasoning Fine-tuning

E.2.1 Training Dataset. The training dataset for Stage I is used to train the LLM to reason about financial questions. We use the financial reasoning dataset provided by the LLM reasoning dataset section A.2. We collected over 80k training samples to ensure comprehensive coverage of financial reasoning patterns and scenarios. We evaluate our model on test datasets from FinQA, FinEval, ConvFinQA, and CFLUE to assess its reasoning capabilities across diverse financial domains. Since our dataset contains both Chinese and English content, we follow conventional practices by using Chinese prompts for Chinese data and English prompts for English data templates.

The templates of the prompts for English and Chinese are as follows:

English Prompt Template

You FIRST think about the reasoning process as an internal monologue and then provide the final answer. The reasoning process MUST BE enclosed within <think> </think> tags. The final answer MUST BE put in \boxed{ }.

Chinese Prompt Template

你首先将思考过程作为内部独白，然后给出最终答案。推理过程必须用<think> </think>标签括起来。最终答案必须放在\boxed{ }中。

E.2.2 Training Procedure. For our financial reasoning tasks, we employ Group Relative Policy Optimization (GRPO) [39] as the primary training methodology. GRPO is specifically designed for training large language models in reasoning-intensive scenarios, making it particularly well-suited for financial question answering and document analysis tasks. The algorithm enhances the model's capability to generate accurate and well-reasoned financial responses through group-based reward normalization and policy optimization.

In our implementation, GRPO samples groups of candidate outputs for each financial task instance and computes group-normalized advantages,

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G)}, \quad (7)$$

and optimizes a clipped objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\min \left(r_{i,t}(\theta) \hat{A}_{i,t}, \right. \right. \right. \\ \left. \left. \left. \text{clip} \left(r_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right) - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right) \right], \quad (8)$$

where $r_{i,t}(\theta) = \frac{\pi_{\theta}(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}$ is the importance sampling ratio, and ϵ is the clipping parameter. Following the approach in DAPO [63], we omit the KL divergence regularization term as it has been shown to be unnecessary for chain-of-thought reasoning scenarios.

Following the design of DeepSeek-R1 [15], our reward function consists of two main components: format reward and accuracy reward, designed to encourage both proper reasoning structure and correct answers.

Format Reward. The format reward encourages the model to follow a specific output structure with reasoning steps enclosed in `<think>...</think>` tags and the final answer in `\boxed{}` tag. The format reward is defined as:

$$R_f(y) = \begin{cases} 1, & \text{if format matches} \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

where y denotes the model's output. Format matching requires the output to contain exactly one pair of `<think>` tags and one `\boxed{}` tag, with no additional content outside these tags.

Accuracy Reward. Given the complexity of financial scenarios where exhaustive enumeration of answer patterns through rule-based methods is challenging, we adopt Qwen3 [59] as the judge for answer evaluation. The content within the `\boxed{}` tag is extracted using regular expressions and serves as the solution. The accuracy reward is defined as:

$$R_a(y, y^*) = \begin{cases} 1, & \text{if } y = y^* \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where y is the model's output (extracted from `\boxed{}` tag) and y^* is the standard answer. In our specific implementation, since financial problems are mostly multiple-choice questions, multi-select questions, financial calculation problems, and QA questions, we have implemented a set of regular expression matching rules to determine answer correctness based on the answer type.

The final composite reward is computed as:

$$R = \alpha \cdot R_f(y) + \beta \cdot R_a(y, y^*) \quad (11)$$

where α and β are weighting parameters (typically $\alpha = 0.1$, $\beta = 0.9$) to balance format compliance and answer correctness.

Metrics. For evaluating the performance of our financial reasoning models, we employ **Score** (accuracy score) as the primary metric. The score measures the proportion of correctly answered questions in the test dataset and is defined as:

$$\text{Score} = \frac{N_{\text{correct}}}{N_{\text{total}}} \times 100\% \quad (12)$$

where N_{correct} represents the number of questions answered correctly and N_{total} denotes the total number of questions in the evaluation dataset. A question is considered correctly answered if the model's output within the `\boxed{}` tags matches the standard answer according to our regular expression matching rules.

Methods. We evaluate our model **FinReasoner** against open-source LLMs on the test dataset. We systematically evaluate several open-source LLM methods, including **DeepSeek-R1** [15], **Qwen3-8B** [59], **Fin-R1-7B** [15], and **Qwen2.5-7B-Instruct** [44]. We use the same evaluation protocol as the original paper to ensure fair comparison.

Implementation Details. Our model, **FinReasoner**, is built on the **Qwen-8B** language model as the base LLM. In our implementation, we integrate Verl 0.5.0 [46] as an external dependency library within the `libs` directory. We customize both the trainer and the reward computation as plugins, which are injected into Verl's trainer (details can be found in the `mverl` module of our codebase). This implementation approach offers significant advantages: since Verl is frequently updated, the plugin injection method provides greater flexibility for maintaining synchronization with the official Verl library.

All experiments are conducted on 16 NVIDIA A100 GPUs distributed across 2 nodes with 8 GPUs per node. We employ tensor model parallelism with a parallel size of 4 to efficiently handle large model training. The training configuration includes a batch size of 64, with micro batch sizes of 16 per GPU for both PPO training and log probability computation. We set the maximum prompt length and response length to 4096 tokens each, with left truncation for input sequences. For GRPO training, we use a rollout size of 8 and train for 50 epochs. The learning rate is set to 1e-6 for stable convergence during the reinforcement learning phase.

Stage II: Market Environment Learning

E.2.3 Training Dataset. For Stage II training, we employ the market dataset, where we use prompt templates to provide historical OHLCV data and news text for LLM analysis and reasoning. We select 6 US stocks (AAPL, AMZN, GOOGL, META, MSFT, and TSLA) with nearly 10 years of data from 2015-05-01 to 2025-05-01. In the data processing pipeline, we first download OHLCV and news data from FMP. However, since the news text provided by FMP contains some advertisements and irrelevant content, we employ web crawling tools to scrape original news webpages, then filter out extremely long and short texts. Subsequently, we use Qwen3-32B to summarize and extract effective textual information from each news webpage, and finally use the summary as part of the news content in our prompts. The distribution of news tokens before and after summarization is shown in Figure 9.

In our specific implementation, we use historical 7-day OHLCV data and news as input. Since there are many news texts per day, we sample 3-5 news articles per day and organize historical prices, news, and historical effective decisions (BUY, SELL) records into prompts as LLM input. The prompt structure is as follows:

Trading Prompt Template

You are a highly experienced trader. Your task is to carefully analyze the provided financial news, current price, and historical trading records. Based on your professional judgment, provide the single best trading decision for the current situation.

Here is the task:

Name: Apple Inc., Symbol: (AAPL)

Price (7 days OHLCV data)

close	high	low	open	volume
173.56	174.03	171.9	173.02	5.37245e+07

News (3-5 news articles)

Timestamp | Title | Content

2023-05-10 15:39:19 | Unleash Your iPhone's Inner Energizer Bunny: Battery Life Tips For The Tech-Savvy | One of the most common concerns among Apple Inc...

Historical Trading Records (7 days historical trading records)

Timestamp	Decision	Price	Volume
2023-05-01 09:30:00	BUY	173.56	5.37245e+07

Record

timestamp	open	high	low	close	volume	price	cash	position	pre_value	action	post_value
2023-04-25	165.19	166.31	163.73	163.77	4.87141e+07	169.59	100000...				

History Valid Action

timestamp	open	high	low	close	volume	price	cash	position	pre_value	action	post_value
2023-05-05	170.98	174.3	170.76	173.57	1.13453e+08	173.57	100000...				

Note

1. `timestamp`: the timestamp of the record
2. `open`: Open price
3. `high`: High price
4. `low`: Low price
5. `close`: Close price
6. `volume`: Volume of the asset traded
7. `price`: Current price (adj_close price)
8. `cash`: Current cash
9. `position`: Current position
10. `pre\_value`: Previous total value, `value = cash + position \* price`
11. `action`: Action taken, `BUY`, `SELL`, or `HOLD`
12. `post\_value`: Current total value
13. `ret`: Return, `ret = (post\_value - pre\_value) / pre\_value`

Today is 2023-05-16 23:59:59, and the current price, cash, and position are 172.07, 98359.28, and 0000.

You should follow the instructions below in detail when making your decision.

1. Your full reasoning process must be enclosed within <think></think> tags.
2. Your final answer must be one of the following three options: BUY, HOLD, or SELL, and be presented only inside a single boxed {}.
3. Do not output anything else except the reasoning in <think>...</think> and the final answer in boxed {}.

Example output:

<think>Based on the news indicating strong earnings, a rising price trend, and positive historical returns, a BUY decision is justified.</think>
boxed{BUY}

E.2.4 Training Procedure. Following the design principles of RAGEN [52], our reward function in the single-stock trading environment consists of two main components: format reward and trading reward, which are designed to ensure standardized action outputs and effective trading decisions.

Format Reward. The format reward encourages the model to adhere to a strict action output format. For each trading action a , the format reward is defined as:

$$R_f(a) = \begin{cases} 1, & \text{if the action format is correct} \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

where the action output must specify a valid trading operation (e.g., buy, sell, or hold) and the corresponding quantity, without any extraneous or malformed content.

Trading Reward. The trading reward evaluates the profit and loss generated by the model’s trading decisions over a trajectory of T trading days. It is computed as the relative change in the agent’s portfolio value, taking transaction costs into account:

$$R_t(\tau) = \frac{V_T - V_0}{V_0} \quad (14)$$

where V_t is the portfolio value at time t :

$$V_t = \text{Cash}_t + \text{Position}_t \times \text{Price}_t \quad (15)$$

Here, Position_t is the number of shares held at time t , and Price_t is the stock price at time t . At each trading step, a transaction cost proportional to the traded volume is deducted:

$$C_t = \lambda |\Delta \text{Position}_t| \times \text{Price}_t \quad (16)$$

$$V_t \leftarrow V_t - C_t \quad (17)$$

where λ is the transaction fee rate (e.g., 1×10^{-4}).

The final reward for each trajectory is a weighted combination of the format and trading rewards:

$$R = \gamma R_f(a) + (1 - \gamma) R_t(\tau) \quad (18)$$

where γ is a hyperparameter (e.g., $\gamma = 0.1$) balancing the importance of action format and trading performance.

Metrics. We evaluate the performance of the trained model using the following trading metrics: **ARR**, **SR**, **MDD**, **CR**, **SoR**, and **VOL**.

Methods. We evaluate our trading models using a simplified FinAgent-based AI Agent framework developed in this work. This framework supports both commercial models (e.g., GPT-4.1, Claude-4-Sonnet) and local models (e.g., Qwen), allowing the underlying LLM to be flexibly replaced. Therefore, our comparison includes **DeepSeek-R1** [15], **Qwen3-8B** [59], **Fin-R1-7B** [15], **Qwen2.5-7B-Instruct** [44], **GPT-4.1**, **Claude-4-Sonnet**, as well as our own **FinReasoner**, all serving as the backbone LLMs for the FinAgent. It is worth noting that a detailed description of our FinAgent (LLMs Agents) implementation will be provided in the next section; please refer to Appendix E.3 for details.

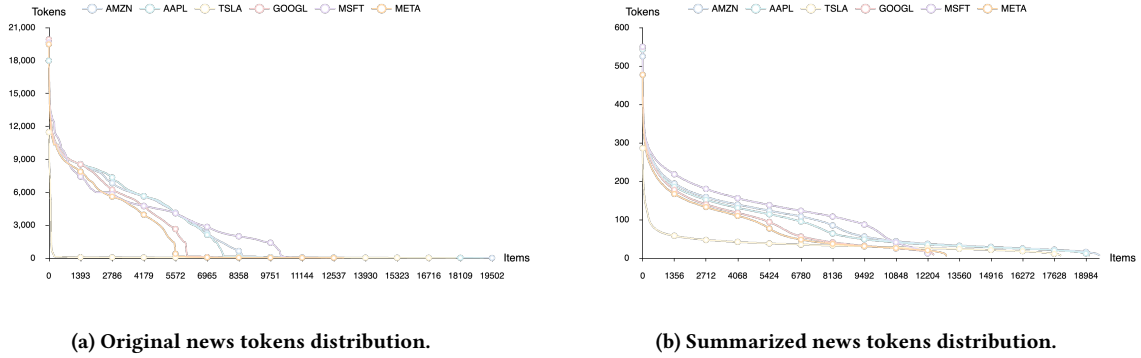


Figure 9: News tokens distribution before and after summarization.

Implementation Details. Similar to Stage I, we implement this using Verl and external plugins to inject additional trainers and evaluation functions. Notably, we also need to inject an environment implementation to enable dynamic interaction between the LLMs and the environment.

All experiments are conducted on 16 NVIDIA A100 GPUs distributed across 2 nodes with 8 GPUs per node. We employ tensor model parallelism with a parallel size of 4 to efficiently handle large model training. The training configuration includes a batch size of 64, with micro batch sizes of 16 per GPU for both PPO training and log probability computation. We set the maximum prompt length and response length to 8192 tokens each, with left truncation for input sequences. For GRPO training, we use a rollout size of 8 and train for 50 epochs. The learning rate is set to $1e-6$ for stable convergence during the reinforcement learning phase.

Conclusion of Reinforcement Learning for LLMs Regarding the design choices in our financial LLM component, our primary contribution focuses on a two-stage reinforcement learning framework rather than complex prompt engineering. In the first stage, RL equips the LLM with core financial knowledge and reasoning capabilities; in the second stage, it enables exploration and trading decisions in simulated market environments. To highlight the effectiveness of RL, we intentionally keep prompts simple, providing only essential inputs such as market news, price trends, and historical actions. Similarly, we adopt a unified and simple reward interface to ensure the training framework remains easy to use and reproducible, allowing researchers to freely extend or customize more sophisticated reward designs for richer research applications. Furthermore, all data used in RL-for-LLM experiments are locally stored market datasets, without relying on online retrieval or external tools. This simplification ensures a stable and fully offline-reproducible training process, avoiding the influence of tool-related factors such as API budgets and allowing experiments to focus purely on validating the effectiveness of RL in financial LLM tasks.

E.3 LLMs Agents for Financial AI

Before introducing financial AI agents, it is important to first accept the concept of a Tool-Calling Agent. In the early stages of agent development, action-based agents played a central role. Notable examples include SayCan [1] and ReAct [61], which are built around action selection: they interpret high-level instructions via a language model and choose among a set of predefined skills or actions based on feasibility and expected utility. In contrast, agents such as Voyager [49] represent the action generation paradigm, where executable behaviors are dynamically synthesized, often in open-ended environments, allowing the agent to build a library of composable skills without being constrained by a fixed action space.

With the emergence of function calling, the rise of tools became evident. Most contemporary agents began to define functions as tools through standardized interfaces, with large language models learning to decide when and how to invoke these tools. It is important to distinguish between the concepts of “action” and “tool”: a tool can serve as a unified interface encapsulating multiple actions, and in some cases, a tool may itself be implemented as an agent.

A further shift occurred with the advent of Anthropic’s Model Context Protocol (MCP), which introduced a standardized method for tool invocation via a JSON-RPC protocol. The appearance of MCP led to the development of a new generation of tool-calling agents that are constructed on this protocol, enabling agents to reliably discover and utilize external tools in a more dynamic and interoperable manner.

These advancements culminate in the design of our general-purpose task agent AgentOrchestra [69], which is a hierarchical multi-agent framework designed to systematically address the key challenges of generalization, multimodal reasoning, scalability, and collaboration in complex task-solving environments. The framework adopts a two-tier architecture. At the top level, a planning agent is responsible for understanding user tasks, decomposing them into manageable sub-tasks, and orchestrating the workflow by dynamically assigning these sub-tasks to specialized lower-level agents. The lower level consists of several specialized agents: a Deep Analyzer for in-depth analysis of input data and extraction of key insights, a Deep Researcher that conducts comprehensive research and automatically generates research reports or knowledge summaries, and a Browser Use agent that automates browser-based web searches and information extraction to assist the research process. Additionally, a General Tool Calling Agent provides a unified interface for invoking various external tools and APIs through function calling, enabling efficient execution of specific tasks and integration with diverse external services. Through this layered design, the system achieves flexible coordination, efficient information processing, and robust tool integration.

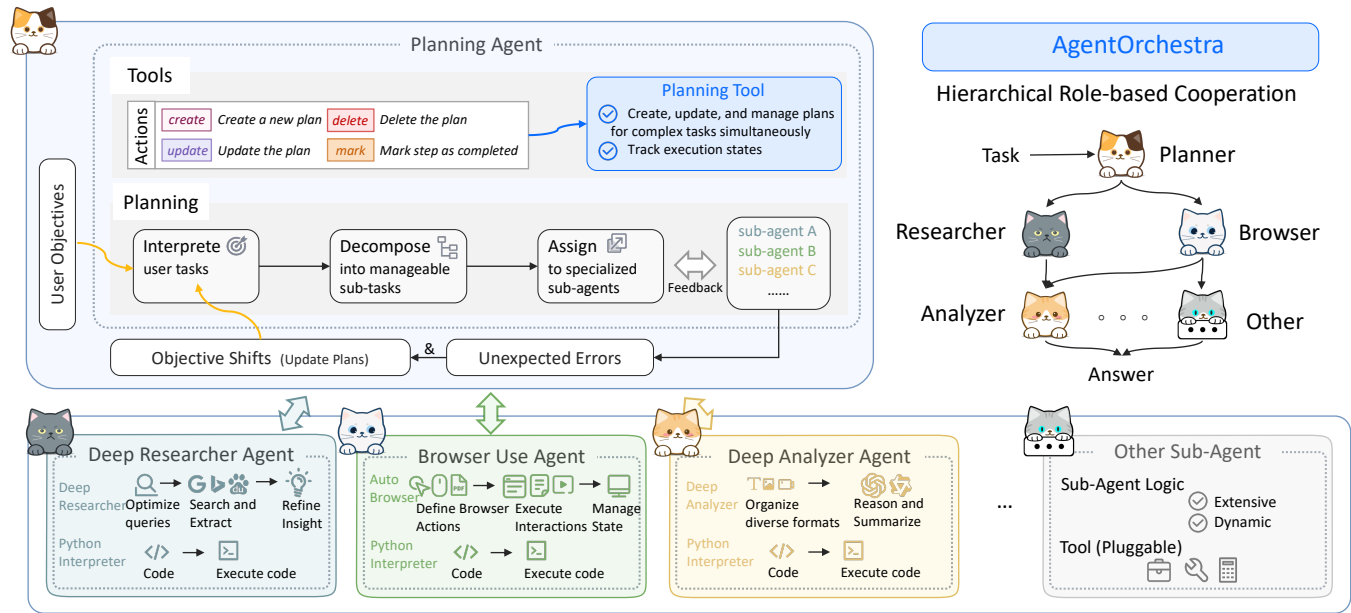


Figure 10: AgentOrchestra Architecture.

AgentOrchestra incorporates several innovative technical features to ensure flexible and efficient handling of complex tasks. It leverages asynchronous coroutine scheduling for high-concurrency collaboration among agents, greatly improving system responsiveness and throughput. The framework natively supports seamless switching between leading commercial large models and open-source local models, balancing capability, privacy, and cost. Comprehensive support for both local and remote MCP, enables secure and unified integration of agents and tools across on-premises and cloud environments. Additionally, AgentOrchestra is fully compatible with OpenAI Function Calling and standard JSON invocation, allowing structured, automated cooperation between tools and sub-agents, and significantly enhancing interoperability and automation.

We have now integrated AgentOrchestra into the FinWorld framework. The core modules consist of three parts: agents, models, and tools. The agent module defines the basic async coroutine concurrent agents, as well as the execution logic and prompt templates for deep analyzers, browser-use agents, general agents and so on. As described in Appendix F.3, we encapsulate the unified interface for commercial LLMs within the LLM API module. The tools module provides several basic tools, such as web crawlers, python code interpreters, arbitrary format file readers, and asynchronous tools like the MCP tools. Furthermore, after integrating the AgentOrchestra modules into FinWorld, it can be used to develop other agents. For example, it can be used to build agents that can retrieve the latest news from the internet in real time and make decisions based on the news, or to develop multi-agent systems capable of quantitative company analysis. This framework provides a fundamental infrastructure for developing financial agents and offers the flexibility to create customized financial agents based on individual ideas and creativity.

Our FinAgent is developed based on the general agent, and its prompt template, as described in Trading Prompt Template E.2.3, enables the agent to focus on financial decision-making tasks. The main workflow of FinAgent is to retrieve the last two rounds of dialogue history from memory, embed this historical information into the current prompt template, and then use function calling to let the LLM decide whether the action is BUY, HOLD, or SELL. This process is repeated for each day’s decision until the task is completed. To fairly compare the open-source FinR1 and our Finreasoner models, we only need to load the corresponding model from the models module.

F Architecture of FinWorld

FinWorld adopts a layered architecture design where each layer has clear responsibilities and functional boundaries. This layered design not only improves system modularity and maintainability but also provides flexible usage patterns for users at different levels. This section provides a detailed introduction to each layer of FinWorld, including the configuration layer, data layer, model layer, task layer, and interface layer.

F.1 Configuration Layer

The configuration layer of FinWorld is built on top of mmengine [7], providing a unified and extensible configuration system based on Python dictionaries. This design allows all experimental parameters such as dataset sources, model architectures, training procedures, and evaluation settings to be defined in a centralized, human-readable, and programmatically modifiable format. By leveraging mmengine’s support for hierarchical and modular configuration, users can construct complex experimental setups through inheritance, parameter overriding, and dynamic imports. This approach enhances reproducibility, transparency, and collaboration while separating configuration logic from implementation details.

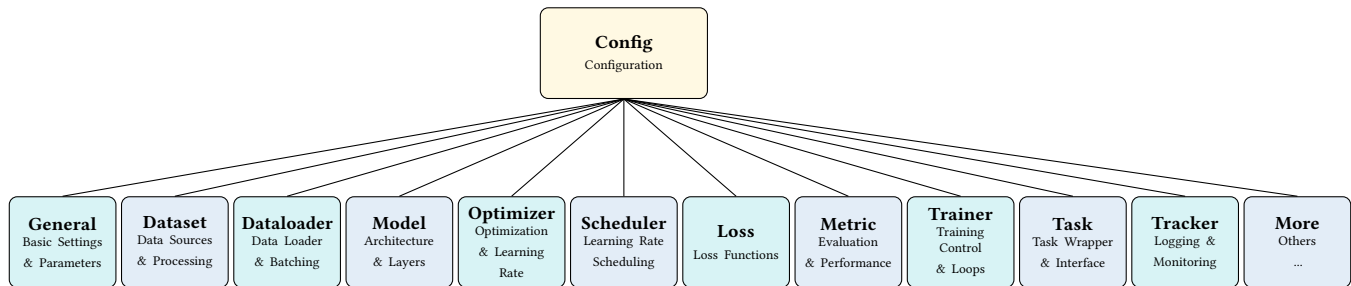


Figure 11: Configuration Layer Architecture of FinWorld

To enable flexible and decoupled component management, FinWorld adopts a registry mechanism for managing all class declarations. Key components such as Dataset and Environment are registered under a global registry and referenced through the type field in the configuration. When loading configurations, the system retrieves the corresponding class from the registry and instantiates it using the provided arguments. This object-oriented design supports extensibility and consistent runtime behavior across the entire system.

F.2 Dataset Layer

The dataset layer of FinWorld comprises multiple functional modules designed to enable standardized, extensible, and task-oriented data management for financial AI research. This layer abstracts the complexities of diverse data sources and modalities, offering a unified interface for data acquisition, preprocessing, and task-specific organization. Specifically, it consists of five main modules: data downloader module, data processor module, dataset module, dataloader module, and environment module, each supporting flexible customization and extensibility.

Downloader Module. This module is composed of two tightly integrated components. The first is the Market Data Downloader, which provides a unified interface for heterogeneous financial data providers such as FMP [42] and Alpaca [41]. It supports multiple temporal resolutions including daily and minute level data, and accommodates diverse data modalities such as OHLCV time series, fundamentals, news, and alternative signals. The component performs schema harmonization, symbol normalization, and calendar alignment across exchanges. It

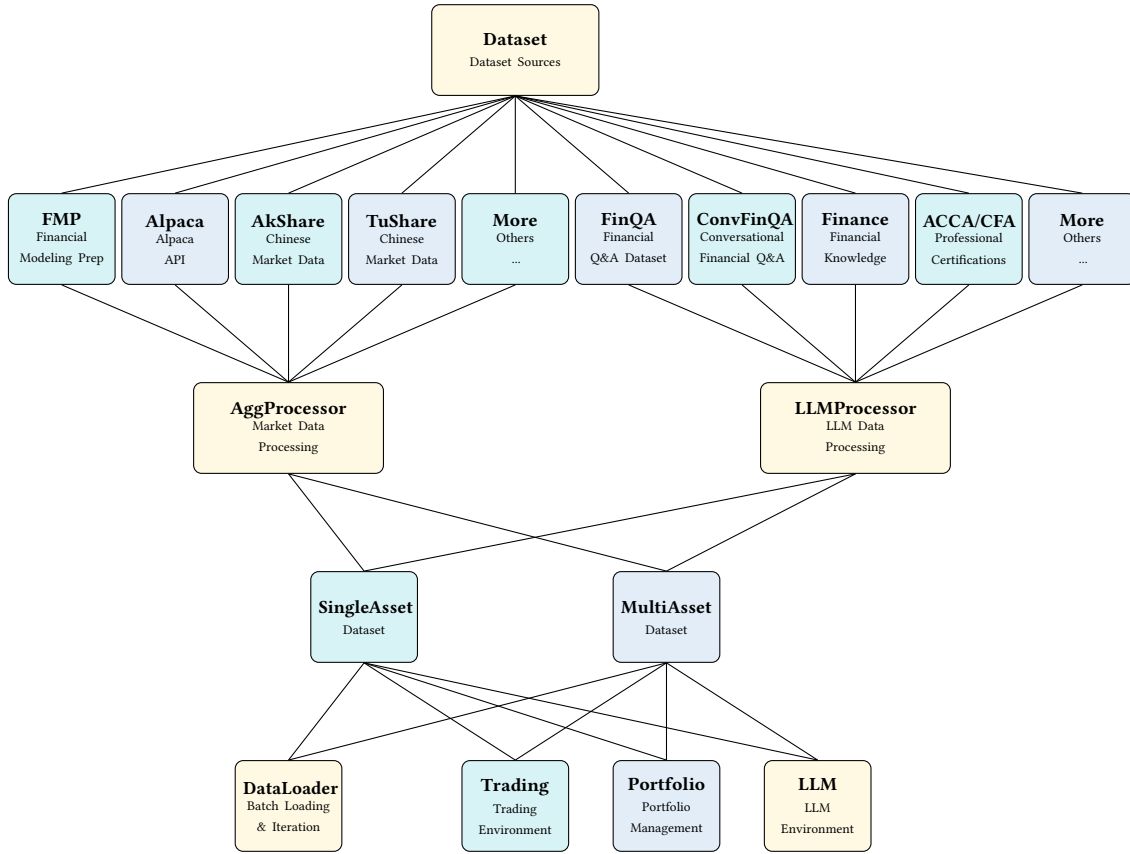


Figure 12: Dataset Layer Architecture of FinWorld

manages authentication, pagination, and provider specific rate limits, and includes robust retry logic and fault tolerance. All ingested records are validated against a canonical schema and enriched with provenance metadata, including provider, version, retrieval timestamp, and transformation history, in order to ensure complete reproducibility. The second component is the LLM Reasoning Dataset Downloader, which offers programmatic access to financial reasoning corpora such as FinQA, professional certification materials including ACCA and CFA, and curated financial knowledge bases. It enforces licensing checks, version pinning, and deterministic train, validation, and test splits. For textual resources, it provides optional document cleaning, de duplication, and citation preservation. Together, these components implement a single abstraction for acquisition from application programming interfaces and databases, with caching and incremental updates that support scalable and repeatable data collection for research.

Processor Module. Building on standardized access, this module implements configurable preprocessing pipelines for both numerical and textual data. For market data, it supports factor construction such as alpha158 [60], rolling window feature engineering, calendar and timezone alignment, corporate action adjustment, missing value imputation, outlier handling, and normalization at the asset level or the cross sectional level. Leakage prevention is enforced through proper temporal ordering, expanding windows, and walk forward splits. For reasoning datasets, the module performs tokenization, span alignment, answer normalization, and the construction of chain of thought or rationales when available. It also supports weak supervision signals, label smoothing, and schema mapping across heterogeneous datasets. All pipelines are declared through a concise configuration that is version controlled and executable, enabling exact reproduction of preprocessing steps across experiments. Comprehensive logging records parameters, random seeds, and checksums for every intermediate artifact.

Dataset Module. This module materializes processed data into task specific datasets that are directly consumable by downstream models. For single asset or algorithmic trading tasks, it packages aligned windows of features and targets at the individual asset level with optional masks for market holidays and missing intervals. For portfolio management and multi asset settings, it builds unified panels that synchronize symbols onto a common trading calendar and provide consistent feature spaces, along with instrument level metadata such as sector, exchange, and liquidity attributes. The module natively supports multiple modalities, including numerical time series, structured tabular fundamentals, and unstructured text such as news, filings, and analyst commentary. It can produce formats suitable for supervised learning, sequence to sequence modeling, retrieval augmented generation, and question answering over financial documents. Dataset objects include explicit train, validation, and test partitions with clear time boundaries, as well as standardized evaluation splits for rigorous comparison.

Dataloader Module. This module exposes performant dataloaders for machine learning, deep learning, and applications that rely on large language models. It provides streaming and memory mapped reading for large time series, efficient mini batch construction with sliding or expanding windows, negative and positive sampling strategies, class balancing, and sequence padding with attention masks. The interface integrates naturally with common training loops, supports mixed precision, and enables deterministic shuffling that respects temporal constraints. Collation utilities handle variable length sequences, multi asset batching, and multimodal inputs that combine numerical features with tokenized text. These capabilities reduce input output overhead and simplify integration with external frameworks.

Environment Module. This module encapsulates data within interactive environments designed for reinforcement learning. It provides standardized definitions of episodes, actions, and rewards for tasks such as single asset trading, multi asset allocation, and execution with partial fills. The environment simulates realistic market frictions, including transaction costs, slippage, liquidity constraints, position limits, and optional market impact models. Evaluation protocols include walk forward backtesting, out of sample testing on rolling windows, and live paper trading interfaces for prospective validation. In addition to conventional agents, the environment supports agents driven by large language models through tool interfaces for observation summarization, query of external knowledge, and action justification. All environment configurations are specified declaratively and are accompanied by seeded randomness and event logs, which promotes rigorous ablation studies and reproducibility across a broad range of agent based learning tasks.

F.3 Model Layer

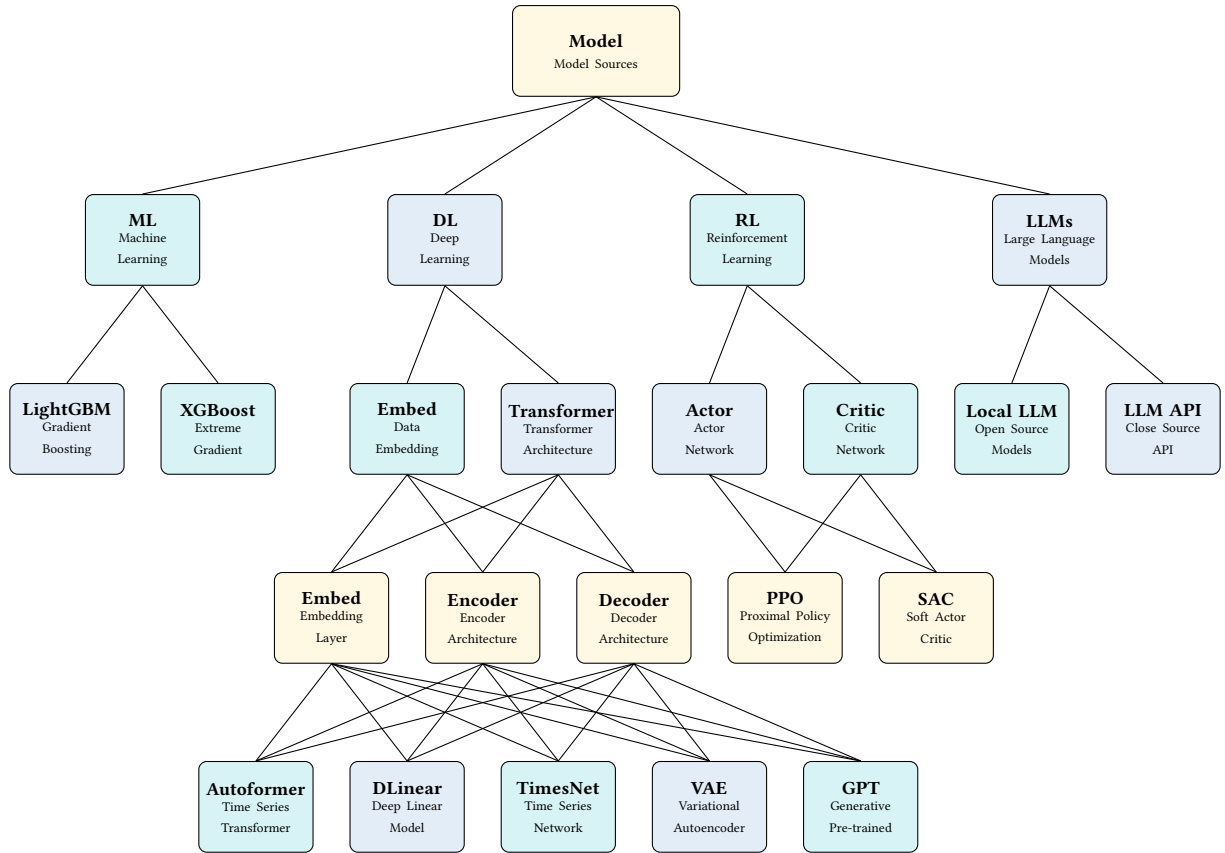


Figure 13: Model Layer Architecture of FinWorld

The model layer of FinWorld comprises a set of specialized yet interoperable modules that support a unified definition, management, and invocation of heterogeneous modeling paradigms across the platform. This layer abstracts both construction and orchestration for traditional machine learning, deep neural architectures, and LLMs, exposing standardized interfaces for training, inference, and downstream integration in financial AI workflows. By decoupling model specification from execution and deployment, it enables consistent behavior across tasks, reproducible configurations, and smooth interchangeability of components.

Machine Learning Models. This module provides a declarative specification for classical models under a uniform schema. Supported classes include linear and logistic regression, decision trees, random forests, and gradient-boosting ensembles (e.g., XGBoost and LightGBM).

Each model is described by a consistent input/output contract and an explicit task head (regression, binary classification, or multiclass classification). Structural attributes, such as regularization choices, tree growth policy, and ensemble topology, are expressed as first-class parameters, enabling transparent comparison across alternatives and systematic hyperparameter control. The uniform abstraction ensures that training and scoring pipelines, metrics, and model persistence follow the same conventions regardless of the underlying estimator.

Deep Learning Models. Neural architectures are organized into composable *components*, *layers*, and *models* with stable interfaces. Components include data embedding for financial time series [55], patch embedding [9], positional encodings, attention mechanisms, transformer blocks, and common activation functions. Layers assemble these components into an embedding layer, an encoder, and a decoder that can be mixed and matched without changing external contracts. Networks such as Autoformer, VAE, and GPT-style decoders are realized by assembling the layers under one specification that remains independent of training and inference. This separation supports rapid iteration on architecture variants while preserving consistent preprocessing, batching, and evaluation semantics.

Reinforcement Learning Models. This module defines RL network structures using an actor-critic abstraction. Policy and value networks are composed from the Deep Learning components, with backbones chosen from MLP, LSTM/GRU, or Transformer families. Specifications cover discrete or continuous action heads and value heads, allow shared or separate encoders, and optionally incorporate memory. Financial constraints are captured as structural hooks (transaction costs, slippage and market impact, risk limits, and trading calendars) so that environment dynamics and trading/portfolio objectives can be expressed alongside the network definition. The result is a consistent interface for simulation and live inference that aligns with the rest of the platform’s training and serving patterns.

LLMs Models. This module centralizes access to proprietary and open-source LLMs behind a unified interface. It supports seamless switching among commercial providers (e.g., GPT-4.1, Claude-4-Sonnet, Gemini-2.5-Pro) and efficient local models (e.g., Qwen2.5, Qwen3), with standardized controls for context length, sampling behavior, and cost/latency tracking. Built-in capabilities include function calling, tool use, and retrieval-augmented generation tailored to financial documents. The module integrates naturally with agents and downstream models, enabling document understanding, information extraction, and instruction following under the same invocation patterns as other FinWorld models. This symmetry reduces integration overhead and promotes consistent monitoring and evaluation across modalities.

F.4 Training Layer

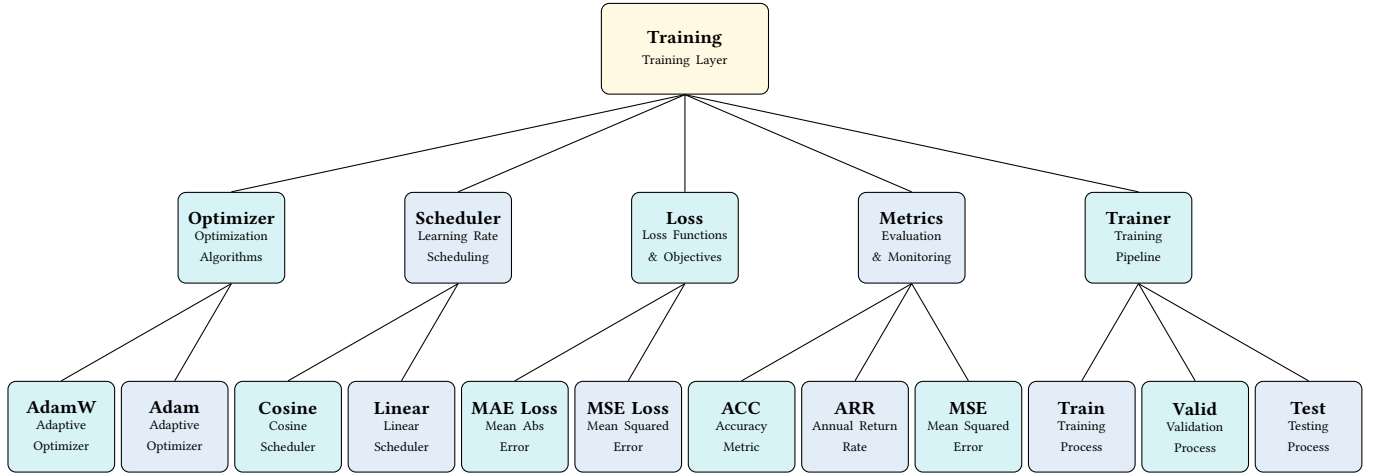


Figure 14: Training Layer Architecture of FinWorld

The training layer in FinWorld offers a modular scaffold that abstracts every element required to optimise all method pipelines for financial applications. A uniform interface guarantees experiment reproducibility, smooth scaling from single GPU notebooks to distributed clusters, and quick transfer of best practices across tasks such as time series forecasting, trading, portfolio management, and LLM applications.

Optimizer. A rich catalogue of first order methods, ranging from classic stochastic gradient descent (SGD) to adaptive variants such as Adam and AdamW, lets practitioners match optimisation dynamics to task characteristics. A common wrapper normalises hyperparameter signatures and supports gradient centralisation, decoupled weight decay, and mixed precision updates, ensuring robust convergence on noisy and non stationary market data.

Loss. A flexible factory produces objective functions that cover regression losses (mean squared error and mean absolute error), classification losses for event prediction, and reinforcement learning surrogates for policy and value objectives. Composite losses can be declared with a single line, enabling multi task learning or the joint optimisation of risk adjusted return and prediction accuracy.

Scheduler. Static strategies (step, cosine, linear) and adaptive schemes (warm up then decay, reduce on plateau) are available for both learning rates and regularisation coefficients. Schedulers are time aware, resuming the exact trajectory when checkpoints are reloaded.

Metrics. In addition to generic accuracy and error scores, the library ships with finance specific diagnostics such as annual return, Sharpe ratio, maximum drawdown, and tail risk measures. Each metric is logged at configurable intervals and can trigger early stopping or hyper parameter sweeps, enabling data driven model selection.

Trainer. Acting as the orchestrator, the trainer pipes data loading, forward and backward passes, gradient clipping, metric evaluation, checkpointing, and experiment logging (TensorBoard or WandB). Specialised variants provide task specific logic, for example the forecasting trainer, trading trainer, portfolio trainer, and large language model trainer. Clear callback hooks let researchers inject custom steps, such as on the fly data augmentation or bespoke risk constraints, without changing the core loop.

Together, these components form a coherent architecture that accelerates experimentation, strengthens reproducibility, and lowers the barrier to launching state of the art AI solutions in the demanding environment of financial markets.

F.5 Evaluation Layer

The evaluation layer of FinWorld is designed to provide a unified, extensible, and highly modular evaluation system for financial AI tasks. At its core, the evaluation layer organizes all assessment methods into two main branches: metrics and plot, each responsible for different but complementary aspects of model evaluation.

Metrics. The metrics defined in the training layer are reused in the evaluation implementation.

Plot. The plot module provides advanced visualization tools that enable intuitive analysis and diagnosis of model performance. Supported plot types include K-line (candlestick) charts, cumulative return curves, compass plots, star plots, and sunburst diagrams. This modular design ensures that both quantitative and qualitative evaluation needs are met in a systematic and reproducible manner.

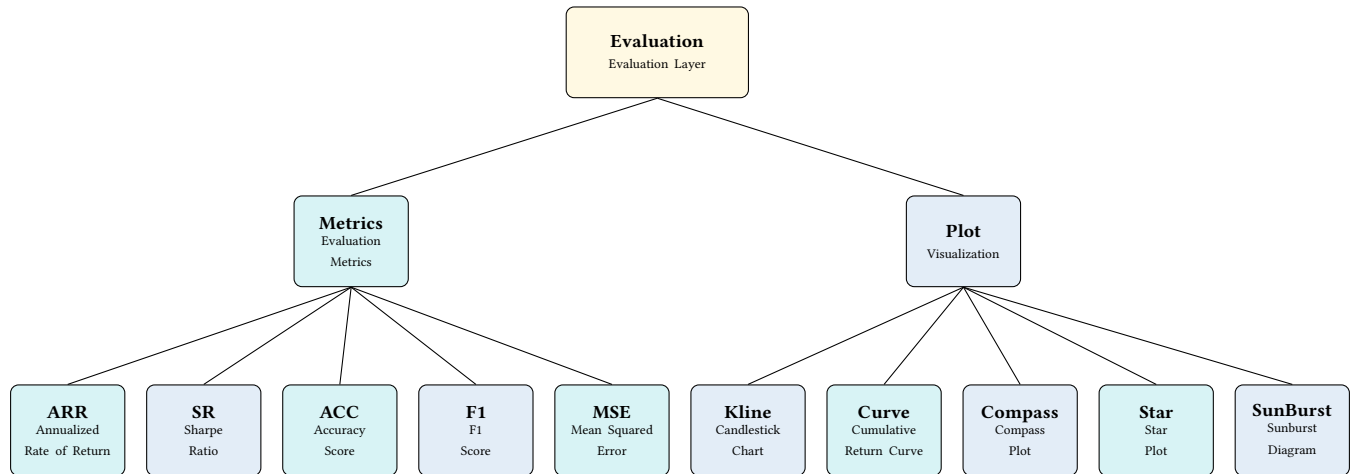


Figure 15: Evaluation Layer Architecture of FinWorld

This adaptive evaluation design simplifies the extension and customization of assessment routines, while ensuring consistent and standardized evaluation across the platform. By streamlining model assessment for diverse financial tasks, it enables systematic comparison, rapid diagnosis of strengths and weaknesses, and iterative improvement of financial AI methods. In our implementation, the evaluation layer is integrated into the trainer’s validation and test stages, allowing metric and plotting methods to be shared with the training phase.

F.6 Task Layer

At its core, the task layer is responsible for the systematic definition, abstraction, and encapsulation of financial AI task types. It systematically supports a wide spectrum of financial AI tasks, providing unified abstractions and modular interfaces that facilitate integration with upstream data modules and downstream modeling components. This layer centers on several core financial tasks, including time series forecasting, algorithmic trading, portfolio management, and LLMs applications, which are formally defined in Section 3.

Each task is specified by configurable input/output schemas and standardized evaluation protocols, supporting both established benchmarks and user-customized scenarios. The unified task architecture enables rapid prototyping, cross-task generalization, and reproducible research across financial AI applications. Additionally, the layer provides a flexible framework for deploying LLM-based agents in asynchronous, single-agent, or multi-agent configurations, allowing users to compose and customize financial agents for diverse research and application needs.

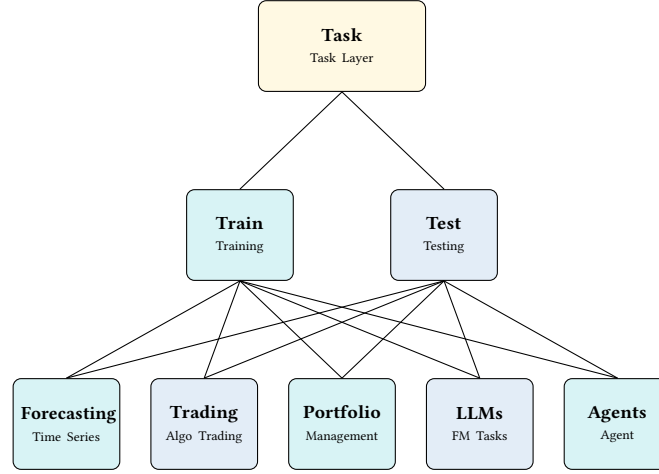


Figure 16: Task Layer Architecture of FinWorld

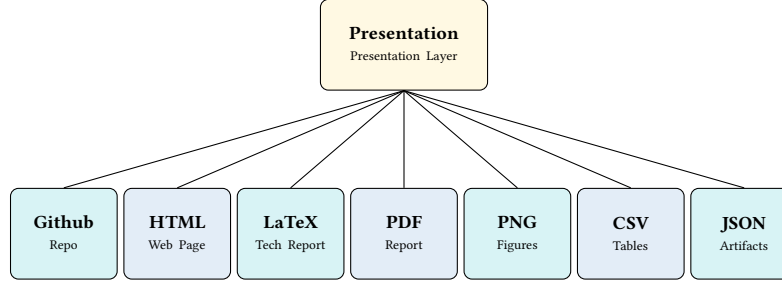


Figure 17: Presentation Layer Architecture of FinWorld

F.7 Presentation Layer

The presentation layer in FinWorld is designed for automated and multi-modal dissemination and documentation of experimental results. At the center of this layer is a dedicated presentation agent, which coordinates the aggregation of evaluation outputs, the automatic generation of technical reports, and the creation of interactive web pages for result interpretation and sharing. Experimental findings, along with key visualizations and benchmark summaries, are systematically compiled into structured documents and published to collaborative platforms such as GitHub and GitHub Pages (github.io), ensuring transparent sharing and long-term accessibility. Seamless integration with experiment tracking tools like Wandb also enables real-time visualization and comparison of metrics throughout the research lifecycle.

This automated presentation pipeline is highly configurable. Users can customize report templates, select specific evaluation results to highlight, and choose the output format that best fits their audience, including detailed PDF or LaTeX technical reports, interactive dashboards, and web-based summaries. The layer supports multi-channel publishing, allowing results to be distributed simultaneously across research repositories, organizational knowledge bases, and public web platforms. This significantly expands the reach and impact of the research. By centralizing the documentation and presentation process, the layer ensures that all experimental records, plots, and analyses are consistently organized and easily retrievable, supporting systematic archiving and knowledge transfer.

The presentation layer also emphasizes both human and machine interpretability. Generated reports include not only visual and tabular summaries, but also machine-readable artifacts such as JSON or CSV exports, which can be used for downstream analysis or automated benchmarking. Support for version control and changelog generation makes it straightforward to track the evolution of results over time, aiding reproducibility and collaborative development. By combining automation, configurability, and multi-modal output, the presentation layer streamlines communication, supports peer review and compliance, and enhances the visibility, transparency, and long-term value of financial AI research within the platform.