Polished title

OrVisZM3P Benchmark: Bridging MetaQuotes Language's (MQL5) API with Pub/Sub-scriber ZeroMQ for Microsecond Bid/Ask Orderflow Data Extraction from MetaTrader (MT5) and External Visualization Across Seven Programing lagiuages

OrVisZM3P Benchmark: Bridging MetaQuotes Language's (MQL5) API with Pub/Sub-scriber ZeroMQ for Real-time Microsecond Bid/Ask Orderflow Data Extraction from MetaTrader (MT5) and External Visualization Across Rust, Go, C++, Python, Java, C#, and NodeJS for Algorithmic Trading

Or — orderflows in bid/ask form
Vis – Visualization after microsecond extraction
Z - means ZeroMQ binding to bridge between the MQL5 API to another programing languages
M - means MetaTrader's (MT5) MetaQuotes Language (MQL5)
3 - three combination (eg, rust + zmq + mql5)
P - the programming languages (Rust, Go, C++, Python, Java, C#, and NodeJS)

Authors

Albeos, Rembrant Oyangoren
Independent Researcher

Draft. Overview Title

Binding MT5's MQL5's API: Exploiting Real-time Microsecond live trading data from MetaTrader (MT5) using ZeroMQ bound to MetaQuotes (MQL5) Expert Advisor, bridging 3 combinations to programming languages of either such as Rust, Go, C++, Python, Java, C#, NodeJS to extract and visualize developing Bid/Ask Orderflows in an external UI chart application.

MT5Flow

Exploiting Real-time Microsecond Bid/Ask Orderflows data from MetaTrader 5 Using Rust, ZeroMQ, and MetaQuotes Language (MQL5)

At the time of writing, there is no publicly known approach using Rust, ZeroMQ, and MetaQuotes Language (MQL5) combination to exploit real-time microsecond bid/ask orderflows data from MetaTrader 5, before this implementation. This study will show a new approach to extract and visualize the developing microsecond bid/ask formation with a low-latency phase. We used MT5's API to fetch real-time live trading data, bound to ZeroMQ & Expert Advisor MQL5 integration, and Rust as an external processor to generate fast-phase developing bid-ask orderflow visualization through a simple chart.

At the time of writing, it's really hard to find benchmark research papers since no one is interested in binding MT5's API using a ZMQ bridge to other programming languages such as Rust, Go, C++, and Python (creating a combination of 3). We search across the internet using 9 AI leading platforms, namely Grok, Gemini, Claude, ChatGPT, MSCopilot, Perplexity, Baidu (Ernie Bot), DeepSeek, and Qwen (1, 2, 3, 4, 5, 6, 7, 8, 9). Some of the AI's responses end up hallucinating, suggesting a combination of 2, and not the strictly 3, but later been clarified and confirmed that there are no public projects yet to implement before this. We also searched across four scholarly libraries, such as Arxiv, Google Scholar, SSRN, and Baidu 学术 (10, 11, 12, 13). The lack of literature suggests that this study's approach (specifically using MT5's MQL5 API to fetch live-trading datafeed from its connected Broker) is bound to ZMQ, bringing the other programming languages are not that popular and not well explored.

Detailed ZMQ (The DLLs require that you have the latest Visual C++ runtime (2015).) binding to MQL - https://github.com/dingmaotu/mql-zmq (14)
ZMQ binding to Rust - https://github.com/zeromq/zmq.rs (15)
ZMQ binding between MQL and JavaScript - https://github.com/EricSchles/bindings-mql4-5 (16)

We used this repository to train (be-aware) locally our Gemini Pro inside Google Antigravity to analyze and have a benchmark of what has been proven to work so far. This approach is similar to training LLMs using open-source repositories to generate source code. Such as Proposes *RepoMark*, a framework to audit whether a code repository has been used in training a code large language model, addressing transparency and license compliance concerns in training on open-source projects (17). Investigates whether a given model has *actually used* specific code from public repositories in its training data via membership inference, providing methods for detecting code inclusion (18). By being mindful of regulatory concerns (19) and following practices (20). With that being said, Gemini Pro already has access to the internet.

MetaTrader 5 (MT5)

MetaTrader 5 (MT5) is a multi-asset trading platform developed by MetaQuotes Software and launched in 2010. It is designed to facilitate trading in various markets, including Forex, stocks, futures, and CFDs. (21). It supports connecting to a regulated broker (22)

MetaQuotes Language 5 (MQL5) MQL5 Reference – How to use algorithmic/automated trading language for MetaTrader 5 (23)

MQL5 API Common APIs - MQL5 Programming for Traders (24)

Methodology
Combination of

A	Rust, ZeroMQ, and MQL5	RuZM
B	Go, ZeroMQ, and MQL5	GuZM
C	C++, ZeroMQ, and MQL5	C+ZM
D	Python, ZeroMQ, and MQL5	PyZM
E	Java, ZeroMQ, and MQL5	JaZM
F	C# , ZeroMQ, and MQL5	C#ZM
G	Node.js, ZeroMQ, and MQL5	NoZM

Make a software application for those combinations and make a simple bid/ask exploit. Download CSV files, run all software at once using the same ZMQ socket. Compare using a correlation matrix to see how well (in %) does the developing bid/ask formation correlate to each other.

Polished title

OrVisZM3P: Bridging MetaTrader's (MT5) MetaQuotes Language's (MQL5) API with ZeroMQ for Real-Time Microsecond Bid/Ask Orderflow Data Extraction and Visualization Across Rust, Go, C++, Python, Java, C#, and NodeJS

Or — orderflows in bid/ask form
Vis – Visualization after microsecond extraction
Z - means ZeroMQ binding to bridge between the MQL5 API to another programing languages
M - means MetaTrader's (MT5) MetaQuotes Language's (MQL5)
3 - three combination (eg. rust + zmq + mql5)
P - the programming languages (Rust, Go, C++, Python, Java, C#, and NodeJS)

API-access-database Projects

Both historical and recent practices fetch trading-data from various platforms to develop trading strategies. These platforms with accessible APIs are not designed for trading purposes but mainly to analyze portfolios and trading trends, namely *Yahoo Finance & Twitter*'s data is used for Long-Short Term Memory neural network (LSTM) modeled time series analysis (1), *Alpha Vantage* for Naive Bayes machine learning algorithm for predicting stock prices (2), *Finnhub* is used as part of a large-scale Financial News and Stock Price Integration Dataset (FNSPID) model for time series financial analysis (3), *Massive* (was Polygon.io) to aggregate news articles (4)*, IEX Cloud* to retrieve real-time stock data to predict stock prices (5), *SimFin's* balance sheet data for machine learning fundamental analysis to predict stock prices (6), *Quandl's* dataset as a part of alternative data and sentiment analysis (7). And consideration of the cryptocurrency

side, namely *CoinGecko's* databases (8), *CryptoCompare's* datasets for real-time data architecture (9), *BraveNewCoin's* daily prices datasets for analysis of the entry and exit dynamics of the cryptocurrency market (10), *Twelve Data's* datasets for quantitative analysis for Stocks and Cryptocurrencies (11), *Messari's* datasets for style investing (12), Glassnode's dataset for Bitcoin price direction prediction using on-chain data and feature selection (13). Some of these platforms do offer forex-based datasets, which are also used for studies, such as from *Yahoo Finance's* datasets for forecasting with feature-augmented multivariate LSTM models (14),

And many more platforms with API access for trading datasets. Although it is widely used for trading analysis and machine learning modeling, those platforms do not offer a trade-exectable APIs. Scholars are studying to create strategies, but they are not mainly for live deployment purposes, which is why we argue that's the reason why there models may fail in live environment since their datasets itself for modeling/practing doesnt have trade-exucutable API. If we

Financial Modeling without the aim to implement it on live environment sucks.

Trading with two completely different platforms, one connected with the dataset API and one with connected has trade-executin API, may cause mispriceing ot mis alignment of prices. It was forecast and perfectly fine to have a stoploss of 2R, but the broker had the ad ada dip a few ticks below what was in the chatting platform.

two arguments: first, They conduct model construction using datasets from platforms that don't have trade-executable API support, meaning they just model and never meant to make it implement on live. Second, Modeling using two completely different platforms, one for charting and one for trade execution, may cause a misprice alignment that may lead to unfortunate losses even if run in real-time. Was modeled perfectly, but failed with a few ticks on live.

The great example of this is TradingView same assestr

by also implementing TN the model in live environment, yhen we must use platforms that offers both API for datasets and API for trade-exucution. For instance, Metatrader5 (MT5)

There are many scholarly studies besides what was mentioned above, with the use of platforms to access financial datasets to model their trading strategies, but the use of those platforms is almost irrelevant for 'live-trading', at least for lower timeframe trading (we are not specifically referring to HFTs trading). Its magnificent to design these machine learning based technologies

but they often fail in live trading, simply because they analyze inside a dataset providers or chart platforms and if they place a trade from a completely different place, then there would be a mispricing alignment. Nnot only that,

Unlike Binance, which has an externally accessible API for trade execution, brokers such as vlavlalvalvllavlalvalvlav doesnt duppoert that kind of API. But Meratrader addressthge issue. Metra trader has limited use as its native language was MWLr built after cc++, as of now at the time of writing, MT5 API is not that popular in scholarly studies for rkerhk, some claimed that it's due to old-like version bit havent, this study will ima a unique application of exploiting data. Less popular is the use of the MetaTrader5 Python library for developing trading studies.

Does not API-access-database Projects

Meanwhile, some brokers do not offer public external APIs for feature-modeling or algorithmic trading and are only accessible to private entities, namely Exness's API, IC Markets's API, XM Group's API, FPMarkets API's, AvaTrade API's, and more. Although APIs can't be accessed publicly and individually, there are ways to extract data, feature modeling, and conduct algorithmic trading. MetaTrader 5 (MT5) is a third-party trading platform (15) developed by MetaQuotes Software, which recently won Prestigious Awards at Forex Expo Dubai 2025 (16). It is designed to connect users to their chosen forex broker and offers advanced trading tools and features, making it suitable for multi-asset trading across various instruments.

[1] Baku, Azerbaijan. (2023). Predicting Financial Market Trends using Time Series Analysis and Natural Language Processing. https://arxiv.org/abs/2309.00136

[2] Kunal Raut, Pinak Kasture, Chetan Gosavi, Tanmay Deshpande. (2022). Stock Market Prediction using Alpha Vantage API and Machine Learning Algorithm. https://www.irjet.net/archives/V9/i5/IRJET-V9I5162.pdf

[3] Zihan Dong, Xinyu Fan. (2024). FNSPID: A Comprehensive Financial News Dataset in Time Series. https://arxiv.org/html/2402.06698v1

[4] Marian Pompiliu Cristescu , Dumitru Alexandru Mara,* , Raluca Andreea Nerișanu , Lia Cornelia Culda and Ionela Maniu. (2023). Analyzing the Impact of Financial News Sentiments on Stock Prices—A Wavelet Correlation. https://doi.org/10.3390/math11234830

[5] Anurag Gupta, Dr. Vishal Shrivastava, Dr. Akhil Pandey, Er. Ram Babu Buri. (2025). Real-Time Stock Market Data Analysis and Visualization Using Power BI. https://ijrpr.com/uploads/V6ISSUE5/IJRPR45525.pdf

[6] AkshayKhanapuri, NarayanaDarapaneni, AnweshReddyPaduri. (2024). Utilizing Fundamental Analysis to Predict Stock Prices. https://doi.org/10.4108/airo.5140

[7] Kristian Bondo Hansen, Christian Borch. (2022). Alternative data and sentiment analysis: Prospecting non-standard data in machine learning-driven finance. https://doi.org/10.1177/20539517211070701

[8] David Vidal-Tomás. (2022) Which cryptocurrency data sources should scholars use? https://doi.org/10.1016/j.irfa.2022.102061

[9] Adrian Barradas, Acela Tejeda-Gil, Rosa-María Cantón-Croda. (2022) Real-Time Big Data Architecture for Processing Cryptocurrency and Social Media Data: A Clustering Approach Based on k-Means. https://doi.org/10.3390/a15050140

[10] David Vidal-Tomás. (2021).The entry and exit dynamics of the cryptocurrency market. https://doi.org/10.1016/j.ribaf.2021.101504

[11] Akshat Sharma; Ashtha Goyal; Durgaprasad Gangodkar; Yogesh Lohumi. Quantitative Analysis for Stocks and Cryptocurrencies using Python. (2024) https://doi.org/10.1109/ICEECT61758.2024.10739272

[12] Fatima Abd Rabbo, Mustafa Disli. (2025). Style investing and return comovement in the cryptocurrency market. https://doi.org/10.1016/j.ribaf.2025.102949

[13] Ritwik Dubey, David Enke. (2025) Bitcoin price direction prediction using on-chain data and feature selection. https://doi.org/10.1016/j.mlwa.2025.100674

[14] Enhancing forex market forecasting with feature-augmented multivariate LSTM models using real-time data. (2025) https://doi.org/10.1016/j.knosys.2025.114500

[15] MetaTrader5. (2026). A powerful platform for Forex and Exchange markets. https://www.metatrader5.com/

[16] MetaQuotes LLC. (2025). MetaQuotes Wins Two Prestigious Awards at Forex Expo Dubai 2025. https://www.metaquotes.net/en/company/news/5498

There is no problem with using platforms that offer an API to access financial datasets, which give data for modeling trading strategies. On the other hand, if we were planning to elevate the modeling approach, then we would also consider making it compatible in a live environment. Simply by using one platform that offers both an API for dataset access and an API for trade-execution. One example of that is in Binance (1). Some studies are dedicated to designing both financial modeling and live implementation using Binance APIs, such as a pipeline using PostgreSQL for historical data storage, Redis for real-time caching of Binance WebSocket streams, and the Binance REST API for trade execution (2), a backtesting and portfolio optimization to live trading results on Binance Futures with a pipeline consists of universe selection, alpha backtesting, volatility aware portfolio optimization, and dynamic drawdown-based risk management (3), FinRL is an open-source deep reinforcement learning framework that acts as a full pipeline from strategy design to simulated trading and can interface with live trading APIs for execution (4), and more. Despite that, Binance does not offer traditional forex trading. Instead, it provides a crypto-oriented approach to forex through several instruments. Binance allows trading in stablecoins such as USDT, BUSD, and USDC, which are pegged to the US dollar (5).

On the other hand, the

[1] Binance. (2026). Binance Spot API Docs. GitHub. https://github.com/binance/binance-spot-api-docs

[2] Elisa Beraudo, Yurii Oliinyk. (2024). The automatic cryptocurrency trading system uses a scalping strategy. https://doi.org/10.20535/2786-8729.5.2024.316563

[3] Thanh Nguyen. (2025). Talyxion: From Speculation to Optimization in Risk-Managed Crypto Portfolio Allocation. https://doi.org/10.48550/arXiv.2511.13239

[4] Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, Christina Dan Wang. (2022). FinRL: a deep reinforcement learning framework to automate trading in quantitative finance. https://doi.org/10.1145/3490354.3494366

[5] BinanceUS. (2026). Listings on Binance.US | Supported crypto, networks, and trading pairs. https://support.binance.us/en/articles/9842915-listings-on-binance-us-supported-crypto-networks-and-trading-pairs

Abstact draft

Abstract Draft

Algorithmic trading research studies always start and may use two types of Application Programming Interface API, one for accessing the datasets from the database, mainly used for feature engineering and chart analysis (API-AD), and one for programmatically executing trades (API-TE). Some financial trading-related platforms offer public APIs to access their database of vast financial data, but don't serve trading-activities. Some platforms offer trading activities, where participants can buy/long, sell/short, hold, hedge, or engage in any form of trading activities, but do not support external publicly available APIs. While some platforms offer both API-AD and API-TE, but limited to asset classes. Unlike what was said, the MetaTrader5 platform, connected to your preferred exchange/broker, offers both API-AD and API-AD and is able to perform feature-modeling and automated algorithmic trading inside the platform, but is limited to Expert Advisors (EAs) programmed on MetaQuotes Language (MQL5). The MetaQuotes LLC company officially published a Python package library integration to MT5; other than that, there are no publicly known available methods to internally extract information and externally project data from MT5. To address that gap, this study implemented a unique approach by combining MQL5 with different programming languages using publisher-subscriber ZeroMQ binding (eg., MQL5 + ZMQ + Rust {~*Go, e.C++, e.Python, Java, Javascript, C#, NodeJS*}). This benchmark-based study allows future quantitative researchers, feature engineers, and algorithmic traders to develop trading systems based on MT5 datafeed with their preferred programming languages, and not be limited to pure MQL5 EAs.

API-AD access database to get financial datasetsA
API-TE trade executions ,progmatically

SUM3API: Bridging Two APIs using ZeroMQ for data extraction from MetaTrader (MT5) and Visualization of bid/ask orderflow formation across

SUM3API: A nine three-pair programming-language API combination designed to internally extract and externally project microsecond-level bid/ask order-flow data from MetaTrader 5

				Order trade executions/operation capability					
	microsecond ordeflow formation			market orders *m* (instant fill)		limit orders *l* (pending)		stop orders *s* (pending)	
API combination	bid/ask	time	tick volume	m.buy	m.sell	l.buy	l.sell	s.buy	s. sell
Rust, ZMQ, MQL5	✓	✓	✓	✓	✓	✓	✓	✓	✓
Go, ZMQ, MQL5	✓	✓	✓	✓	✓	✓	✓	✓	✓
e.C++, ZMQ, MQL5	✓	✓	✓	✓	✓	✓	✓	✓	✓
e.Python, ZMQ, MQL5	✓	✓	✓	✓	✓	✓	✓	✓	✓
Java, ZMQ , MQL5	✓	✓	✓	✓	✓	✓	✓	✓	✓
Javascript, ZMQ, MQL5	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rust, ZMQ, MQL5	✓	✓	✓	✓	✓	✓	✓	✓	✓

Developing tick volume, and buttons for Marketorders (buy and sell), Limitorders (buy and sell), Stoporders (buy and sell).

This study will only use forex and Bitcoin

Add Tick Volume & Trading Order Buttons to MT5 Chart

Add tick volume display and working trading order buttons (Market, Limit, Stop orders with buy/sell functionality) to the egui application, with MQL5 backend changes for order execution.

User Review Required

> IMPORTANT
> **Bidirectional Communication**: This feature requires adding a second ZeroMQ socket pair:
>
> - Current: **PUB/SUB** (tick data from MT5 → Rust) - remains unchanged
> - New: **REQ/REP** (orders from Rust → MT5) on port `5556`
>
> This means the MT5 EA will need to handle **both** sockets simultaneously.

> WARNING
> **Order Execution Risk**: The order buttons will execute **REAL** trades on your MT5 account. Ensure you test on a demo account first.

Full end-to-end testing requires MetaTrader 5 running with the EA attached to a chart on a demo account.

IMPORTANT

Bidirectional Communication: This feature requires adding a second ZeroMQ socket pair:

Current: PUB/SUB (tick data from MT5 → Rust) - remains unchanged
New: REQ/REP (orders from Rust → MT5) on port 5556
This means the MT5 EA will need to handle both sockets simultaneously.

WARNING

Order Execution Risk: The order buttons will execute REAL trades on your MT5 account. Ensure you test on a demo account first.

apendix

I suggest that you check that the software codes if its working. You don't have to trust me fully on this. I am not a graduate of any software development programs; I do not have professional education in finance, mathematics, or anything found in this paper. I am a self-dependent researcher and algorithmic trader who uses Grok as a search engine and Google Antigravity Gemini Pro and ClaudeOpus4.5 for programming workloads. Although I claim that the concept was based on my raw thoughts, I can't deny that the mentioned AI models elevated the project and helped me able to present it to them, readers as comprehensively as possible. Thanks to the companies that run them.

My goal is not to showcase how hard for me to develop SUM3API, but to present a unique way to extract data and communicate with MT5. I aim to help future researchers to explore new approaches in financial trading, feature engineering, algorithmic trading, and more methodologies with the help of a simple benchmark: Bridging MetaQuotes Language's (MQL5) API with Pub/Sub-scriber ZeroMQ for Microsecond Bid/Ask Orderflow Data Extraction from MetaTrader (MT5) and External Visualization Across Seven Programming languages

Steps and configuration

open MT5 platform ->

go to '*Open Data Folder*' (ctrl+shift+d) ->

Go back to VSCode and relocate the files from the repository, and find similar folder paths and relocate to (MQL5\Experts\) for *ZmqPublisher.mq5* and (MQL5\Include\Zmq) *for Zmq.mqh* file. ->

Open MetaEditor5 platform ->

Open MetaEditor5 platform ->

Go to 'Navigator' (ctrl+d) -> Experts folder -> find *ZmqPublisher.mq5* and compile

Open MT5 platform and drop the expert advisor on the screen

Open your terminal and make sure you are at ruth path, just bash it {cd C:\Users\User\Desktop\VSCode\SUM3API\mt5-chart} in my case

Cargo build –release
Cargo run –release
cargo clean

Unlike the official bidding from MT5, you just open your preferred code editor (VSCode, for instance), make a python_file.py folder, and pip install MetaTrader5 (make sure you are using Python 3.13 version)

Run cargo clean, run carg0 buil –release infinitely until no error. The file lock errors are from Windows Defender scanning - running it multiple times usually works. Let me know the result!

I advice that using the raw terminal for running cargo commands (such as ‘cargo clean’, cargo build –release’ ‘cargo run –release’) is much better. Using Antigravity can't bypass my security/Antivirus system

Cancel executed orders option.

labels for marketorders limitorders stoporders (red for sell, blue for buy) and use breakline on chart.

[redacted]

Not that

The foundation of OCHL candlestick which consists of price and time is from bid/ask spread (x-axis as raw up-down tick count), and tick-volume calculation is based on that too.

Open xauusd chart then do step 1

backend database frontend. Mermaid diagrams of the whole system, (includes account info, historical data request, forward data request, trade controls, messenging and notifications, live tick-level bid/ask formation chart) . Detailed Communication Flow, Data Structures, Supported Actions (PUB → SUB), Supported Actions (REQ → REP). File structure. Common errors and debugging.

New Abstract

The MetaTrader 5 (MT5), connected to your preferred exchange or broker, supports automated algorithmic trading via Expert Advisors (EAs) written in MetaQuotes Language (MQL5). Also, MetaQuotes LLC provides an official Python integration package; there are limited publicly documented methods for internally extracting and externally projecting MT5 financial data, other than that. To address this gap, this study implements a unique approach that bridges MQL5 and Rust via ZeroMQ publisher–subscriber & request-reply bindings. Nothing is compared in this study, and this benchmark-based methodology enables quantitative researchers, feature engineers, and algorithmic traders to develop trading systems leveraging MT5 datafeeds using Rust, bypassing the limitations of pure MQL5 EAs. The methodology is proven through integrating it in one software application (simple trading termminal) demonstrating these low-latency functionality: includes real-time account info (balance, equity, free and used margin), historical data request (OCHL, raw tick), forward data request (live recorded), trade controls (buy and sell either Market, Limit, or Stop orders) with lotsizing and trade cancelation, messenging and notifications (debugging and recent calls), and a live microsecond raw tick-level bid/ask formation chart.

Intro

Research studies and methods substantiate API-based data requests as a common starting point in practical implementations, particularly for enabling scalable feature engineering and quantitative analysis in algorithmic trading (1, 2, 3, 4, 5, 6, 7). Acquiring datasets to model their own way of trading, analyzing portfolios, parsing event sentiments, and decoding trading narratives.

[1] David Jukl, Jan Lansky. (2025). Systematic Review on Algorithmic Trading. Acta Informatica Pragensia, 14(3), 506-534. https://doi.org/10.18267/j.aip.276

[2] Jifeng Li, Arnav Grover, Abraham Alpuerto, Yupeng Cao, Xiao-Yang Liu. (2025). Orchestration Framework for Financial Agents: From Algorithmic Trading to Agentic Trading. https://arxiv.org/html/2512.02227v1

[3] Krishnamurthy Nayak, Supreetha Balavalikar Shivaram, Sumukha K. Nayak. (2025). Machine Learning Framework for Algorithmic Trading. Comput. Sci. Math. Forum 2025, 12(1), 12; https://doi.org/10.3390/cmsf2025012012

[4] Mohit Tyagi, Nookala Venu. (2025). An RSI-Based Algorithmic Trading System Using Angel One Smart API: Design, Implementation, and Performance Evaluation. https://www.ijfmr.com/papers/2025/3/45227.pdf

[5] Adam Darmanin, Vince Vella. (2025). Language Model Guided Reinforcement Learning in Quantitative Trading. https://arxiv.org/html/2508.02366v2

[6] Keyi Wang, Nikolaus Holzer, Ziyi Xia, Yupeng Cao, Jiechao Gao, Anwar Walid, Kairong Xiao, Xiao-Yang Liu, Yanglet. (2023). FinRL Contests: Benchmarking Data-driven Financial Reinforcement Learning Agents. https://arxiv.org/pdf/2504.02281v3

[7] Wentao Zhang, Yilei Zhao, Chuqiao Zong. (2025). FinWorld: An All-in-One Open-Source Platform for End-to-End Financial AI Research and Deployment. https://arxiv.org/html/2508.02292v1