TimeSeriesGym: A Scalable Benchmark for (Time Series) Machine Learning Engineering Agents

Large Language Model (LLM) agents have shown promise in reducing the manual effort in machine learning (ML) engineering, but existing benchmarks have limitations: they source tasks primarily from well-structured problems like Kaggle challenges which do not fully reflect real-world complexity, focus on outcome-based metrics (e.g., success rate) that obscure individual ML skills, and lack scalability due to manual task curation.

To address these limitations, we introduce *TimeSeriesGym*, a comprehensive, scalable, agent-agnostic benchmarking framework for evaluating LLM agents on time series ML engineering tasks. We focus on time series as it is one of the most common data modalities in ML practice and offers a lens into how LLMs handle structured data through agents. The current version of *TimeSeriesGym* features 34 challenges from 23 data sources across 8 problem types (e.g., forecasting, classification, understanding) and 15+ domains (e.g., healthcare, finance, epidemiology). Challenges are derived from Kaggle competitions (n=12) and popular benchmarks or research code repositories (n=14), each designed to evaluate critical ML engineering skills in (1) *Data Handling:* missing data, labeling tools, multi-source integration, (2) *Modeling:* model development, hyperparameter tuning, model selection, research code migration and improvement, and (3) *Benchmarking:* rigorous evaluation on standard benchmarks. To improve accessibility, we also provide *TimeSeriesGym-Lite*, a subset of six challenges that efficiently assess these core skills while maintaining diversity across domains and problems.

TimeSeriesGym enables scalable generation of new challenges by providing detailed documentation and specialized tools (e.g., simulating missing data). It also offers multimodal, skill-based, and holistic evaluation. We design challenges that isolate and test specific skills (e.g., handling missing data, code migration) and fine-grained evaluation tools that assess multiple dimensions of performance, including correctness, instruction following, and code quality. Our evaluation combines quantitative metrics (e.g., accuracy), programmatic analysis (e.g., code inspection), and qualitative evaluation (LLM-as-a-judge), balancing the reliability of objective metrics with the flexibility of subjective assessment to provide actionable feedback.

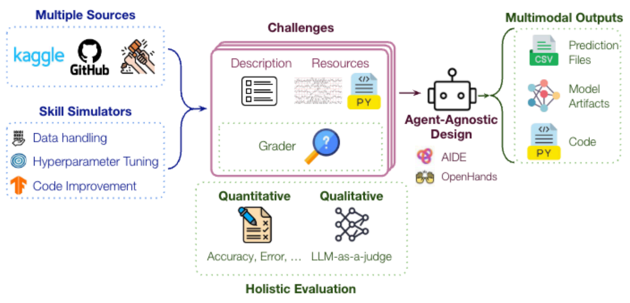

Fig. 1: Overview of *TimeSeriesGym*.

Lite	Model	Resources (hours / steps)	Valid Submission (%)	Reasonable Submission (%)
	`OpenHands`			
✓	+ `gpt-4.1-2025-04-14`	4 / 50	44.4 ± 19.3	11.1 ± 9.6
	`AIDE`			
✗	+ `gpt-4.1-2025-04-14`	4 / 50	57.3 ± 7.9	12.5 ± 0
✓	+ `gpt-4.1-2025-04-14`	4 / 50	66.7 ± 16.7	27.8 ± 9.6
	+ `o3-2025-04-16`		**94.4 ± 9.6**	33.3 ± 0.0
	+ `claude-3-7-sonnet-20250219`		50.0 ± 16.7	38.9 ± 19.3
Effect of Scaling Resources				
✓	+ `gpt-4.1-2025-04-14`	4 / 50	66.7 ± 16.7	27.8 ± 9.6
		8 / 100	72.2 ± 9.6	22.2 ± 9.6
		12 / 150	61.1 ± 9.6	**50.0 ± 0.0**

Tab. 1: Main Results. Each experiment was run with 3 random seeds (shown as mean +/- standard deviation).

Tab. 1 presents the main results of agent runs on *TimeSeriesGym*, comparing scaffold types (OpenHands [1] vs. AIDE [2]), model choices (GPT-4.1, o3, Claude 3.7), and resource allocations (4/50 to 12/150 hours/steps). We report two metrics: (1) *valid*–submissions that yield non-null scores, and (2) *reasonable*–submissions with a genuine attempt at producing a valid solution. Our findings show that: (1) AIDE outperforms OpenHands as a scaffold, (2) the reasoning model o3 achieves the highest valid submission rate (94.4%), (3) Claude 3.7 produces the most reasonable submissions (38.9%), and (4) agents do not necessarily improve with more time. Overall, our findings suggest that while frontier LLMs combined with AIDE scaffolding can achieve moderate to high success rates in producing valid submissions, they still struggle to solve time series ML engineering tasks, particularly those that go beyond standard Kaggle-style challenges and involve multi-file code repositories.

[1] Wang et al. Openhands: An open platform for ai software developers as generalist agents. ICLR (2024).
[2] Jiang et al. Aide: Ai-driven exploration in the space of code. arXiv preprint arXiv:2502.13138 (2025).