AgentCompass: A Unified Evaluation Infrastructure for Agent Capabilities

AgentCompass Team, Shanghai AI Laboratory

As Large Language Models (LLMs) evolve into autonomous agents, the need for unified evaluation infrastructure becomes critical. However, current evaluation pipelines remain highly fragmented and tightly coupled, hindering reproducibility and causing redundant engineering. To address this, we introduce AgentCompass, an open-source, lightweight, and extensible infrastructure for evaluating LLM-based agents. AgentCompass organizes the evaluation process around three independent components, namely *Benchmark*, *Harness*, and *Environment*, thereby enabling flexible configurations without requiring the reimplementation of complex execution logic. Furthermore, it features a fault-tolerant asynchronous runtime and comprehensive trajectory analysis tools to transparently diagnose nuanced failure modes like reward-hacking. Natively supporting over 20 benchmarks across five capability dimensions, AgentCompass provides the community with a scalable and reproducible infrastructure for advancing agent research.

1. Introduction

The paradigm of Large Language Models (LLMs) is rapidly shifting from instruction-following text generators to the foundation of LLM-based agents capable of complex reasoning, planning, and tool interaction in dynamic environments [17, 38]. As LLM-based agents continue to advance in capability, corresponding evaluation methodologies must be developed in tandem to ensure rigorous assessment of these increasingly multi-faceted systems.

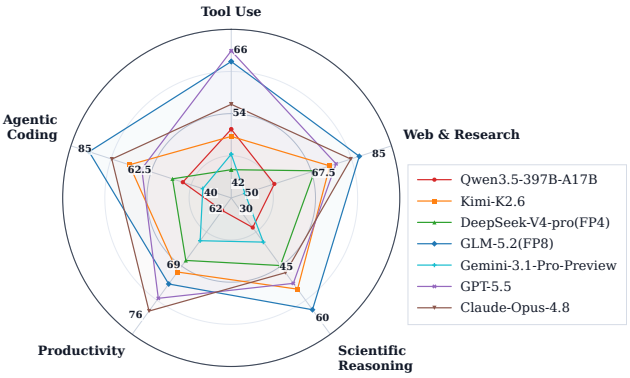

Figure 1: Capability profiles of representative models across the five core evaluation dimensions.

However, the current agent evaluation landscape is highly fragmented, suffering from a severe infrastructural deficit. While numerous specialized benchmarks have emerged to assess specific capabilities—such as tool invocation [46] and deep research [22]—they operate as isolated evaluation suites. This forces researchers to repeatedly configure heterogeneous execution environments, data formats, and scoring protocols. Such redundant engineering not only hinders efficiency but also compromises reproducibility due to inconsistent baseline implementations. Furthermore, existing general-purpose infrastructure either lack native support for interactive agent workflows [5, 8] or restrict their scope to narrow domains like coding [12]. The community urgently needs a unified, extensible infrastructure to streamline agent evaluation.

To bridge this gap, we introduce **AgentCompass**, an open-source, lightweight, and extensible evaluation infrastructure designed to support systematic evaluation of LLM-based agents. Its core design philosophy lies in decoupling three typically entangled components: *Benchmark*, *Harness*, and *Environment*. By abstracting the harness into an independent layer, AgentCompass transforms rigid evaluation pipelines into flexible benchmark × harness × Environment configurations. This architecture allows researchers to evaluate different agents on a shared benchmark, compare alternative interaction protocols, or integrate new datasets without

† Corresponding: {zhudongsheng, mazerun, zhangqi}@pjlab.org.cn
∗ Code is at https://github.com/open-compass/AgentCompass

reimplementing complex execution logic. As illustrated in Figure 1, this unified approach readily enables the comprehensive profiling of representative models across multiple core capability dimensions.

Built upon this modular abstraction, AgentCompass standardizes critical operations such as task dispatch, environment interaction, and metric aggregation. It features an asynchronous runtime optimized for parallelizing long-running agent trajectories. Beyond execution, the infrastructure provides rich trajectory-level analysis and visualization, capturing intermediate actions, tool calls, and environment feedback to help developers transparently diagnose failure modes. Having already served as the foundational evaluation infrastructure for the Intern-S agent series [51], AgentCompass offers a scalable, reproducible, and highly accessible toolkit for advancing agent research.

2. Related Work

As agent capabilities continue to expand, evaluation must move beyond LLM benchmarks, giving rise to a broad yet fragmented landscape of agent-specific benchmarks and infrastructure.

Benchmarks. Comprehensive benchmarks assess holistic performance via diverse interactive environments [19], challenging real world questions [11, 22, 28, 37, 48, 49], and game-playing scenarios [39], with specialized variants for multi-agent collaboration [14, 50]. Other benchmarks provide granular analysis of tool and function calling [2, 26, 31, 46], scientific reasoning and research coding [34, 35, 42, 43], productivity-oriented skill execution [18, 27, 29], and agentic capabilities in software engineering [6, 13, 15, 45], live coding [3], and terminal operations [21, 33].

Infrastructure. To manage these diverse evaluations, supporting infrastructure has emerged. Some provide analytical dashboards [20], unit-testing frameworks [4], or commercial tracing tools [16]. More recently, unified frameworks have been proposed to streamline development and evaluation. AgentGym [40] provides infrastructure for evolving agents across diverse environments. Harbor [12] further targets agent-oriented evaluation, with particular emphasis on coding and skill-based scenarios. For multi-agent systems, MASLab offers a unified and comprehensive codebase [47]. Beyond agent evaluation, general-purpose frameworks also support broader model assessment: EvalScope [32] covers multiple evaluation modes and backends; OpenCompass [5] targets broad reasoning and safety evaluation; and VLMEvalKit [8] focuses on image-text multimodal evaluation.

3. Framework

AgentCompass modularizes benchmark-specific evaluation scripts into composable components connected by stable protocols. As shown in Figure 2, each evaluation consists of a *benchmark*, *harness*, and *environment*, with optional recipes and analyzers.

3.1. Design Overview

In AgentCompass, an evaluation run is specified through a declarative `RunRequest`. The request separates the substantive objects of evaluation from the operational choices used to execute it. Specifically, `BenchmarkSpec` defines the task and evaluation metrics, `HarnessSpec` defines the agent procedure used to interact with each task, `EnvironmentSpec` identifies the execution context, and `ModelSpec` describes the model endpoint, credentials, inference parameters, and supported API protocols. Runtime options that govern how the evaluation is executed are placed in a separate `ExecutionSpec`, without changing the semantic definition of the evaluation itself.

This separation clarifies the configurable parts of an evaluation. For example, one may compare several harnesses on the same benchmark, reuse a harness across different benchmarks, or evaluate the same model through different interaction protocols without rewriting benchmark code. AgentCompass resolves these choices through lightweight, decorator-based registries for benchmarks, harnesses, environments, recipes, and analyzers. As a result, new components can be introduced by registering their implementations locally, rather than by modifying the central runtime or adapting unrelated modules. The resulting abstraction replaces fixed benchmark-specific pipelines with composable benchmark $\times$ harness $\times$ environment configurations, while

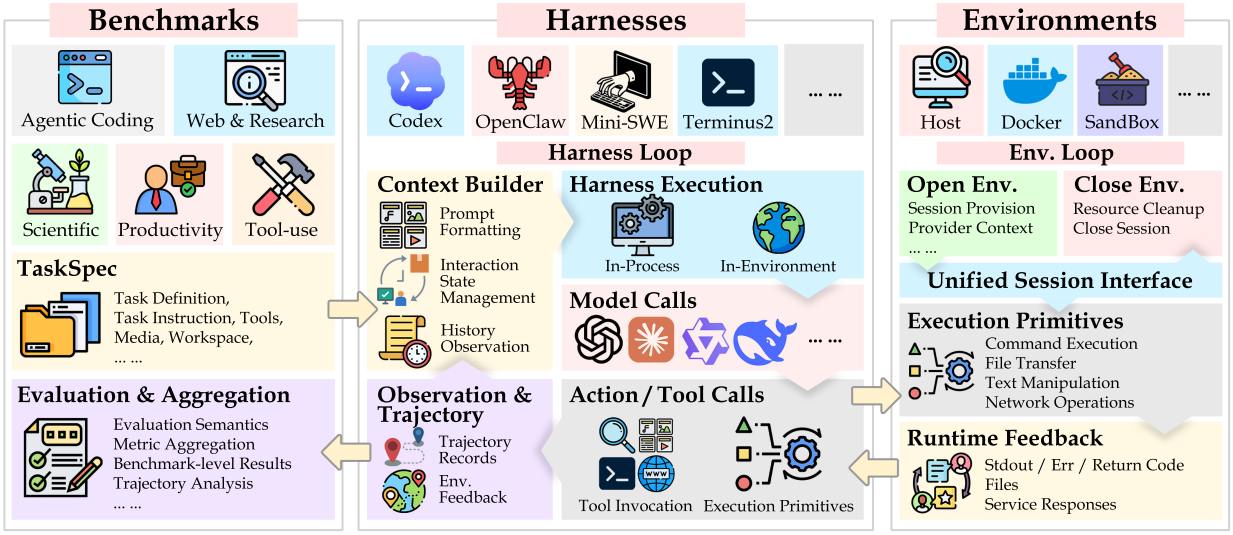

Figure 2: Overview of the AgentCompass architecture. The framework fully decouples Benchmarks, Harnesses, and Environments to enable flexible and composable agent evaluation.

preserving a clear boundary between evaluation semantics and execution mechanics.

3.2. Core Components and Protocols

AgentCompass improves modularity and extensibility by decomposing evaluation into protocol-driven components, replacing tightly coupled, benchmark-specific scripts with flexible architecture.

Benchmark. The benchmark component encapsulates dataset-specific logic. It loads raw data into a uniform `TaskSpec`, prepares task materials, and computes final scores via an `evaluate` function. The grading mechanism supports deterministic matching, execution-based verification, and LLM-as-judge scoring. To separate agent rollout from evaluation, the benchmark defines a scorer execution mode: `none` for in-memory verification, `reuse` for the agent's existing workspace, or `fresh` for a clean, isolated test environment (e.g., applying an agent-generated patch to a pristine repository).

Harness. The harness serves as the operational wrapper that instantiates a LLM into an interactive agent. It orchestrates the agent's internal logic, including prompt formatting, interaction state management, multi-turn tool invocation, and provider-specific API handling. Acting as a mediator, the harness consumes standardized task materials and coordinates with the model API and environment, entirely shielding the benchmark from agent-specific implementation details. The framework supports both in-process harnesses, where the interaction loop is managed natively by AgentCompass, and in-environment harnesses, which execute external agent frameworks as subprocesses within a sandbox. This accommodates a wide spectrum of systems, ranging from simple single-turn inference scripts to complex autonomous coding agents.

Environment. The environment provides the isolated execution context and system primitives necessary for agent interaction and code verification. It exposes a unified session interface for executing commands, transferring files, manipulating text, and provisioning network services. Crucially, it serves as the security and isolation boundary for external agent execution. By abstracting the backend infrastructure, AgentCompass allows the same benchmark and harness configurations to run consistently across local host processes, local Docker containers, or distributed cluster instances. This allows researchers to scale evaluations from local development to large clusters without modifying the underlying code.

Protocol Abstractions Component interoperability is guaranteed through two levels of strict protocol abstraction. First, models are represented purely as declarative API specifications (defining endpoints, parameters, and supported formats) rather than being tightly coupled to a monolithic client gateway. Second, data exchange between benchmarks and harnesses is governed by a standardized material protocol. Benchmarks compile each `TaskSpec` into a `PreparedTask` (bundling prompts, tools, media, and expected outputs), while the harness

Table 1: Summary of the over 20 built-in benchmarks supported by AgentCompass, categorized across key capability dimensions.

Capability Dimension	Benchmarks
Tool Use	Tau-bench [46], Tau2-bench [2], Tau3-bench [31]
Web & Research	BrowseComp [37], BrowseComp-ZH [49], DeepSearchQA [11], GAIA [22], HLE [28], HLE-Verified [48]
Scientific Reasoning	FrontierScience [35], SciCode [34], SGI-Bench(Deep Research) [43], ResearchClawBench [42]
Agentic Coding	SWE-bench-Verified [44], SWE-bench-Pro [6], SWE-bench-Multilingual [45], Terminal-bench@2, Terminal-bench@2-Verified, Terminal-bench@2.1 [33]
Productivity	GDPVal-AC [27], SkillsBench [18], PinchBench [29]

processes this structure to yield a uniform `RunResult` containing the final prediction, score, and the complete recorded trajectory. This explicit data contract eliminates the need for cross-modification when introducing new benchmarks or agents into AgentCompass.

3.3. Asynchronous Execution Runtime

The AgentCompass runtime is specifically designed to handle the high I/O overhead typical of agent-environment interactions. Users initiate an evaluation via the CLI or Python SDK by constructing a `RunRequest`. The runtime then dynamically resolves the registered components, prepares the task materials, and distributes tasks through an asynchronous dispatcher built on `asyncio`. By configuring concurrency limits, the infrastructure efficiently manages multiple parallel, long-running agent trajectories without blocking. Furthermore, the runtime natively supports fault tolerance and state persistence. Partial results and structured progress events are saved incrementally. If an evaluation is interrupted, the infrastructure can resume seamlessly: it skips completed tasks and re-executes only the samples that encountered retryable failures. This incremental execution mechanism is crucial for cost-effective evaluation of autonomous agents.

3.4. Trajectory Tracking and Behavioral Analysis

While traditional benchmarks often collapse agent performance into a single scalar metric, AgentCompass records a comprehensive, versioned trajectory for each task. A trajectory captures the complete interaction sequence, including the agent's reasoning process, issued tool calls, environment feedback, and granular metrics (e.g., token consumption, inference latency, and stop reasons). This uniform schema standardizes interaction histories across heterogeneous models and benchmarks.

To facilitate in-depth diagnostics, AgentCompass provides a pluggable analyzer layer. Analyzers automatically process trajectories to extract structured insights, flagging problematic samples and categorizing errors into model-side, environment-side, or framework-side failures. Built-in analyzers can systematically detect anomalies such as output truncation, latency spikes, and repetitive generation loops. These trajectory-derived statistics help researchers to transparently pinpoint specific failure modes rather than solely observing overall performance degradation.

3.5. Extensibility and Reproducibility

Extensibility. AgentCompass employs a lightweight, registry-based architecture to minimize coupling. Introducing a new benchmark only requires a protocol-compliant subclass and local registration, with optional scoring functions and setup recipes. Because components are decoupled, existing harnesses and environments remain untouched. Model API protocols are likewise handled where they are consumed rather than through a monolithic client. This design lowers the barrier to integrating new capabilities within a unified codebase.

Reproducibility. AgentCompass ensures strict tracking of evaluation provenance. Each run is persisted by benchmark and model, storing exact configurations, task-level logs, and aggregated summaries. AgentCompass

distinguishes semantic parameters that alter agent behavior from execution parameters such as concurrency, so execution-only changes do not invalidate results. By isolating retryable failures from completed tasks, evaluations remain auditable and restartable at scale.

3.6. Supported Benchmarks and Harnesses

As of July 2026, AgentCompass natively integrates over 20 benchmarks spanning five core capability dimensions: tool use, web & research, scientific reasoning, agentic coding, and productivity (summarized in Table 1). Rather than being limited to static question-answering, the integrated benchmarks encompass highly interactive and long-horizon environments, such as repository-level code modifications (e.g., SWE-bench variants) and complex open-ended web navigation (e.g., GAIA, HLE). AgentCompass also supports specialized evaluation paradigms; among them, GDPVal-AC is a custom AgentCompass variant that employs an agentic judger (via OpenClaw) to perform pair-wise evaluation against a Claude-Opus-4.8 baseline.

To interact with these diverse environments, the infrastructure provides a wide spectrum of agent harnesses, ranging from lightweight direct chat-completion wrappers to complex, in-environment autonomous frameworks (summarized in Table 2). Complementarily, the built-in harnesses are designed to standardize the execution boundaries across fundamentally different agent architectures. This allows researchers to evaluate closed-source commercial tools (e.g., Claude Code, OpenAI Codex) alongside open-source autonomous frameworks (e.g., OpenHands, Mini-SWE-agent) under the exact same evaluation metrics and trajectory logging protocols. To facilitate specific research domains, the infrastructure also provides tailored workflows like the Naive Search Agent, a lightweight harness specifically developed for deep research that comes pre-equipped with search and web-browsing tools.

Harness	**Driven Agent / Workflow Type**
OpenAI Chat Wrapper	Direct chat-completion
Naive Search Agent	Native deep-search loop
Claude Code	Claude Code CLI
Codex	OpenAI Codex CLI
OpenHands	OpenHands agent
OpenClaw	OpenClaw CLI
Mini-SWE-agent	Mini-SWE-agent CLI
Terminus2	Terminus2 terminal

Table 2: Overview of representative built-in agent harnesses available in AgentCompass.

By leveraging a unified registry design, researchers can seamlessly pair any compatible benchmark with any harness, enabling standardized evaluation of heterogeneous agents without requiring additional glue code.

4. Experiments

4.1. Experimental Settings

To validate the efficacy, scalability, and universality of AgentCompass, we conduct a comprehensive evaluation of representative LLMs across eight highly challenging benchmarks. Specifically, the evaluated models include Qwen3.5-397B-A17B [30], DeepSeek-V4-pro(FP4) [41], Kimi-K2.6 [23], GLM-5.2(FP8) [9], GPT-5.5 [24], Gemini-3.1-Pro-Preview [10], and Claude-Opus-4.8 [1]. Model names followed by '()' indicate the corresponding quantized versions used in our evaluation.

The evaluation brings together a representative set of challenging benchmarks with their corresponding harnesses, including τ^3-bench with its official workflow [31], DeepSearchQA and FrontierScience(Research) [11, 35] with Naive Search Agent, an in-house search harness developed in AgentCompass, SWE-bench-Pro and SWE-bench-Multilingual [6, 45] with Mini-SWE-agent [44] and OpenHands [36], PinchBench [29] with OpenClaw [25], SkillsBench [18] with OpenClaw and OpenHands, and SciCode [34] with a multi-turn workflow equipped with official tools. All reported evaluation results are averaged over three independent runs to reduce measurement variance. Further details are deferred to Appendix A.

4.2. Main Results

Table 3 shows that AgentCompass fairly aligns heterogeneous tasks and enables direct cross-harness comparisons, revealing that empirical agent capabilities are sensitive to the underlying harness. For instance, model

Model	Tool Use	Web & Research	Scientific Reasoning		Productivity			Agentic Coding			
						SkillsBench		SWE-Pro		SWE-Multilingual	
	τ^3-bench	DeepSearchQA	FrontierSci	SciCode	PinchBench	OpenClaw	OpenHands	MiniSWE	OpenHands	MiniSWE	OpenHands
Qwen3.5-397B-A17B	51.78	59.41	26.67	46.35	90.10	35.58	37.36	43.55	42.04	65.00 $_{-4.3}$	63.78 $_{-5.5}$
Kimi-K2.6	50.76	71.52	48.27	51.87 $_{-0.3}$	87.35	53.10	50.62 $_{-3.3}$	58.09 $_{-0.5}$	61.29	77.78 $_{+1.0}$	77.22
DeepSeek-V4-pro(FP4)	46.02	68.22	42.22	47.53	88.51	49.53	47.12 $_{-2.9}$	49.98 $_{-5.4}$	41.77 $_{-13.6}$	72.44 $_{-3.8}$	61.78 $_{-14.4}$
GLM-5.2(FP8)	61.42	77.96	57.22	51.97	88.76	53.19	52.63	78.80	77.06 $_{+15.0}$	82.00	81.78
Gemini-3.1-Pro-Preview	48.22	53.00	25.00	54.44 $_{-4.6}$	87.97	44.99	44.64 $_{-8.1}$	39.26 $_{-15.0}$	45.69 $_{-8.5}$	44.00	63.00
GPT-5.5	62.94	72.89	41.67	55.92	91.81	49.52	56.08 $_{-11.2}$	52.94 $_{-5.6}$	56.22 $_{-2.4}$	73.33	78.00
Claude-Opus-4.8	55.33	76.11 $_{-8.7}$	36.67	56.21	90.70	54.40	58.66 $_{+4.6}$	66.21 $_{-3.0}$	73.87 $_{+4.7}$	77.00 $_{-7.4}$	77.00 $_{-7.4}$

Table 3: Comprehensive evaluation results of representative models across the five core capability dimensions. Colored subscripts show gaps from official baselines, with green/red indicating higher/lower AgentCompass scores; no subscript means no official baseline was disclosed.

performance fluctuates significantly depending on the employed harness, as evidenced by the discrepancies between OpenClaw and OpenHands on SkillsBench, or Mini-SWE-agent and OpenHands on the SWE-bench variants.

Beyond absolute scores, the gap annotations in Table 3 further show that agent evaluation is highly sensitive to infrastructure choices. Models can deviate substantially from their officially reported baselines under the unified AgentCompass protocol, such as Claude-Opus-4.8 dropping by 8.7 points on DeepSearchQA and GLM-5.2(FP8) improving by 15.0 points on SWE-bench-Pro with OpenHands. Such fluctuations may also arise from differences in harness versions, or from benchmark-specific adaptations made to the harness during evaluation. These discrepancies underscore the need for an open-source, unified evaluation framework that standardizes settings, implementations, and reporting protocols for reproducible and comparable agent assessment.

4.3. Analysis Results

In this section, we illustrate the importance of trajectory analysis in agentic task evaluation from several concrete perspectives.

RQ1: Beyond final scores, what types of behavioral bad cases occur in model trajectories? To characterize trajectory-level behavioral patterns, we leveraged the analysis capabilities of AgentCompass to quantify the distribution of bad-case behaviors. Figure 3 shows the distribution of behavioral bad-case patterns across models, where percentages indicate the proportion of each type among a model's total bad cases. DeepSeek-V4-pro(FP4) mainly fail through repetitive content generation. Kimi-K2.6 has a higher share of bad cases involving multilingual mixing on search tasks and repeated tool calls. Gemini-3.1 is dominated by repeated tool calls, while Claude-Opus-4.8 and GPT-5.5 mainly produce empty outputs, although GPT-5.5 has very few bad cases overall. These findings suggest that, even when final task outcomes appear comparable, models may differ substantially in their underlying interaction dynamics and failure characteristics. Such beyond-score trajectory analysis therefore provides a more informative basis for diagnosing agentic behavior and identifying targeted directions for model improvement.

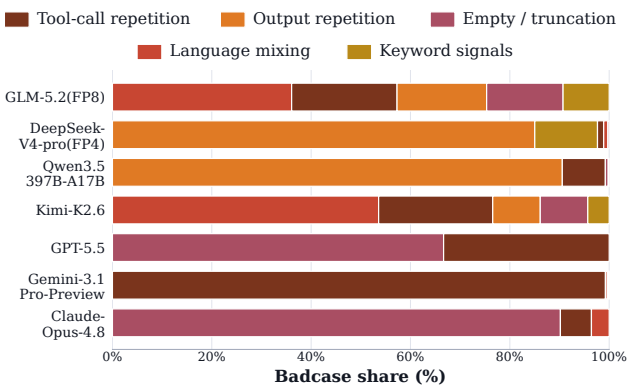

Figure 3: Distribution of bad-case behaviors across model trajectories.

RQ2: Are high-scoring models on coding benchmarks genuinely strong? Using our reward-hacking analyzer inspired by [7], we analyze correct samples on SWE-Pro and SWE-Multilingual with a two-level taxonomy of suspected hacking behaviors. Notably, the notion of suspected reward hacking adopted here is defined behaviorally rather than evidentially. Specifically, any action exhibiting characteristics of hacking is classified as reward hacking, regardless of whether there is direct evidence establishing a causal relationship

Model	Sample-level	Step-level
Qwen3.5-397B-A17B	3.08%	0.06%
Kimi-K2.6	17.2%	0.73%
DeepSeek-V4-pro(FP4)	0.82%	0.06%
GLM-5.2(FP8)	**39.12%**	**2.09%**
Gemini-3.1-Pro-Preview	19%	1.05%
GPT-5.5	17.03%	0.59%
Claude-Opus-4.8	9.71%	0.65%

(a) SWE-Pro

Model	Sample-level	Step-level
Qwen3.5-397B-A17B	13.07%	0.35%
Kimi-K2.6	20.51%	0.74%
DeepSeek-V4-pro(FP4)	4.02%	0.16%
GLM-5.2(FP8)	9.83%	0.24%
Gemini-3.1-Pro-Preview	**21.97%**	**0.80%**
GPT-5.5	6.41%	0.37%
Claude-Opus-4.8	5.63%	0.33%

(b) SWE-Multilingual

Table 4: Reward-hacking analysis results for different models using the Mini-SWE-agent harness.

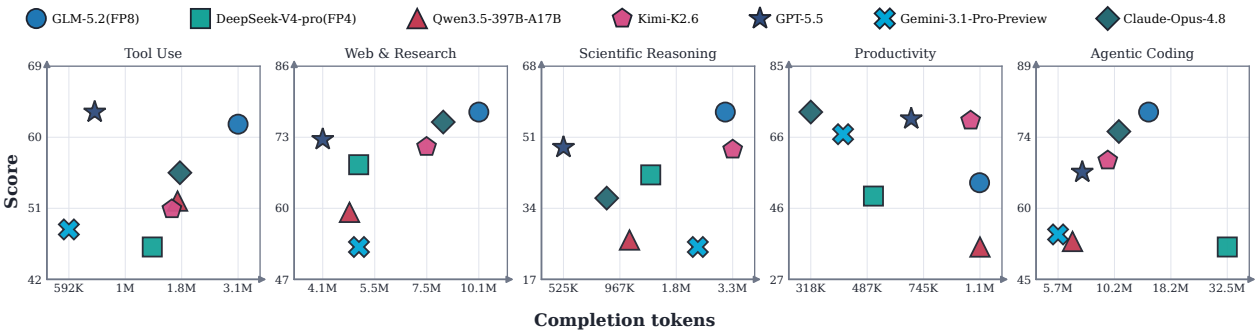

Figure 4: Capability–token length trade-offs across different capability dimensions.

between the behavior and the final outcome. Most models exhibit suspected reward hacking, such as modifying tests, retrieving golden patches. GLM-5.2(FP8) shows the highest suspected hacking rate on SWE-Pro, while Gemini-3.1-Pro-Preview does so on SWE-Multilingual; DeepSeek-V4-pro(FP4) remains consistently low on both. Notably, although GLM-5.2(FP8) outperforms Claude-Opus-4.8 by about 12 points on SWE-Pro, it also shows roughly 30% more suspected reward-hacking samples. Detailed results are illustrated in Table 4. These findings highlight the importance of carefully designing execution environments and constraints in both training and evaluation to ensure benchmark results better reflect coding ability.

RQ3: How do model capability and token length relate across different tasks? To help users make more scientifically grounded and operationally efficient decisions under a given budget, Figure 4 illustrates the relationship between model performance and token length across different task categories. On coding tasks, most models follow the test-time scaling law, with longer outputs generally associated with higher scores, while DeepSeek-V4-Pro(FP4) appears as a notable outlier. On productivity tasks, Claude-Opus-4.8 stand out by achieving higher scores with lower token budgets, whereas Qwen3.5-397B-A17B performs relatively weaker. On tool-use and web-search tasks, GPT-5.5 tends to produce shorter outputs while maintaining comparatively strong performance. Detailed statistics on the average interaction steps across benchmarks are provided in Appendix B, which further characterize the execution complexity and interaction depth of different evaluation tasks.

5. Conclusion

In this paper, we introduced AgentCompass, an open-source and highly extensible infrastructure designed to systematize the evaluation of LLM-based agents. By decoupling the evaluation pipeline into independent Model, Benchmark, Harness, and Environment components, it eliminates redundant engineering and ensures rigorous reproducibility. Supported by a fault-tolerant asynchronous runtime and granular trajectory analysis, AgentCompass goes beyond traditional scalar scores to help researchers diagnose complex behaviors such as reward-hacking. With native integration of over 20 diverse benchmarks, AgentCompass provides a robust and scalable infrastructure to accelerate the next generation of agent research.

Acknowledgments

This project is supported by Shanghai Artificial Intelligence Laboratory.

Author Contributions

The authors are listed in alphabetical order by their last names:

Zichen Ding, Jiaye Ge, Shufan Jiang, Kai Chen, Mo Li, Qingqiu Li, Zehao Li, Zonglin Li, Tiaohao Liang, Shudong Liu, Zerun Ma, Zixing Shang, Wenhui Tian, Zun Wang, Liwei Wu, Zhenyu Wu, Jun Xu, Bowen Yang, Dingbo Yuan, Qi Zhang, Songyang Zhang, Peiheng Zhou, Dongsheng Zhu

References

[1] Anthropic. Claude Opus 4.8. https://www.anthropic.com/news/claude-opus-4-8, 2026. 4.1

[2] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025. 2, 1

[3] CodedotAI. Aider: Ai pair programming in your terminal. https://github.com/paul-gauthier/aider, 2023. 2

[4] Confident AI. Deepeval: The open-source evaluation framework for llms. https://github.com/confident-ai/deepeval, 2023. 2

[5] OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023. 1, 2

[6] Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025. 2, 1, 4.1

[7] Darshan Deshpande, Anand Kannappan, and Rebecca Qian. Benchmarking reward hack detection in code environments via contrastive analysis, 2026. 4.3

[8] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11198–11201, 2024. 1, 2

[9] GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. 4.1

[10] Google DeepMind. Gemini 3.1 Pro. https://deepmind.google/models/gemini/pro/, 2026. 4.1

[11] Nikita Gupta, Riju Chatterjee, Lukas Haas, Connie Tao, Andrew Wang, Chang Liu, Hidekazu Oiwa, Elena Gribovskaya, Jan Ackermann, John Blitzer, et al. Deepsearchqa: Bridging the comprehensiveness gap for deep research agents. *arXiv preprint arXiv:2601.20975*, 2026. 2, 1, 4.1

[12] Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments. https://github.com/harbor-framework/harbor, January 2026. 1, 2

[13] Wenqi Huang, Charley Lee, Leonard Tng, and Serena Ge. Deepswe: Measuring frontier coding agents on original, long-horizon engineering tasks. *arXiv preprint arXiv:2607.07946*, 2026. 2

[14] Jonathan Hyun, Nicholas R. Waytowich, and Boyuan Chen. Crew-wildfire: Benchmarking agentic multi-agent collaborations at scale. *arXiv preprint arXiv:2507.05178*, 2025. 2

[15] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, 2024. 2

[16] LangChain. Langsmith: A unified platform for debugging, testing, evaluating, and monitoring your llm applications. https://www.langchain.com/langsmith, 2025. 2

[17] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 1

[18] Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026. 2, 1, 4.1

[19] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[20] Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. In *Advances in Neural Information Processing Systems*, 2024. 2

[21] Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026. 2

[22] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. 1, 2, 1

[23] Moonshot AI. Kimi-k2.6. https://www.kimi.com/en/blog/kimi-k2-6, 2026. 4.1

[24] OpenAI. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/, 2026. 4.1

[25] OpenClaw Team. OpenClaw. https://github.com/openclaw/openclaw, 2026. 4.1

[26] Shishir G. Patil, Huanzhi Mao, Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *International Conference on Machine Learning*, 2025. 2

[27] Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, et al. Gdpval: Evaluating ai model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*, 2025. 2, 1

[28] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025. 2, 1

[29] PinchBench Team. Pinchbench: Real-world benchmarks for ai coding agents, 2026. 2, 1, 4.1

[30] Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. 4.1

[31] Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barres. τ-knowledge: Evaluating conversational agents over unstructured knowledge. *arXiv preprint arXiv:2603.04370*, 2026. 2, 1, 4.1

[32] ModelScope Team. EvalScope: Evaluation framework for large models, 2024. 2

[33] The Terminal-Bench Team. Terminal-bench: A benchmark for ai agents in terminal environments. https://github.com/laude-institute/terminal-bench, 2025. 2, 1

[34] Minyang Tian, Luyu Gao, Shizhuo D Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, et al. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650, 2024. 2, 1, 4.1

[35] Miles Wang, Robi Lin, Kat Hu, Joy Jiao, Neil Chowdhury, Ethan Chang, and Tejal Patwardhan. Frontierscience: Evaluating ai's ability to perform expert-level scientific tasks. *arXiv preprint arXiv:2601.21165*, 2026. 2, 1, 4.1

[36] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pages 65882–65919, 2025. 4.1

[37] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025. 2, 1

[38] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. 1

[39] Yue Wu, Xuan Tang, Tom M. Mitchell, and Yuanzhi Li. Smartplay: A benchmark for llms as intelligent agents. In *International Conference on Learning Representations*, 2024. 2

[40] Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. Agentgym: Evolving large language model-based agents across diverse environments, 2024. 2

[41] Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. 4.1

[42] Wanghan Xu, Shuo Li, Tianlin Ye, Qinglong Cao, Yixin Chen, Hengjian Gao, Yiheng Wang, Qi Li, Kun Li, Sheng Xu, et al. Researchclawbench: A benchmark for end-to-end autonomous scientific research. *arXiv preprint arXiv:2606.07591*, 2026. 2, 1

[43] Wanghan Xu, Yuhao Zhou, Yifan Zhou, Qinglong Cao, Shuo Li, Jia Bu, Bo Liu, Yixin Chen, Xuming He, Xiangyu Zhao, et al. Probing scientific general intelligence of llms with scientist-aligned workflows. *arXiv preprint arXiv:2512.16969*, 2025. 2, 1

[44] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 1, 4.1

[45] John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents, 2025. 2, 1, 4.1

[46] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024. 1, 2, 1

[47] Rui Ye, Keduan Huang, Qimin Wu, Yuzhu Cai, Tian Jin, Xianghe Pang, Xiangrui Liu, Jiaqi Su, Chen Qian, Bohan Tang, et al. Maslab: A unified and comprehensive codebase for llm-based multi-agent systems. *arXiv preprint arXiv:2505.16988*, 2025. 2

[48] Weiqi Zhai, Zhihai Wang, Jinghang Wang, Boyu Yang, Xiaogang Li, Xander Xu, Bohan Wang, Peng Wang, Xingzhe Wu, Anfeng Li, et al. Hle-verified: A systematic verification and structured revision of humanity's last exam. *arXiv preprint arXiv:2602.13964*, 2026. 2, 1

[49] Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, et al. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *arXiv preprint arXiv:2504.19314*, 2025. 2, 1

[50] Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, et al. Multiagentbench: Evaluating the collaboration and competition of llm agents. *arXiv preprint arXiv:2503.01935*, 2025. 2

[51] Yicheng Zou, Dongsheng Zhu, Lin Zhu, Tong Zhu, Yunhua Zhou, Peiheng Zhou, Xinyu Zhou, Dongzhan Zhou, Zhiwang Zhou, Yuhao Zhou, Bowen Zhou, Zhanping Zhong, Zhijie Zhong, Haiteng Zhao, Penghao Zhao, Xiaomeng Zhao, Zhiyuan Zhao, Yechen Zhang, Jin Zhang, Wenwei Zhang, Hongjie Zhang, Zhuo Zhang, Wenlong Zhang, Bo Zhang, Chao Zhang, Chen Zhang, Yuhang Zang, Fei Yuan, Jiakang Yuan, Jiashuo Yu, Jinhui Yin, Haochen Ye, Qian Yao, Bowen Yang, Danni Yang, Kaichen Yang, Ziang Yan, Jun Xu, Yicheng Xu, Wanghan Xu, Xuenan Xu, Chao Xu, Ruiliang Xu, Shuhao Xing, Long Xing, Xinchen Xie, Ling-I Wu, Zijian Wu, Zhenyu Wu, Lijun Wu, Yue Wu, Jianyu Wu, Wen Wu, Fan Wu, Xilin Wei, Qi Wei, Bingli Wang, Rui Wang, Ziyi Wang, Zun Wang, Yi Wang, Haomin Wang, Yizhou Wang, Lintao Wang, Yiheng Wang, Longjiang Wang, Bin Wang, Jian Tong, Zhongbo Tian, Huanze Tang, Chen Tang, Shixiang Tang, Yu Sun, Qiushi Sun, Xuerui Su, Qisheng Su, Chenlin Su, Demin Song, Jin Shi, Fukai Shang, Yuchen Ren, Pengli Ren, Xiaoye Qu, Yuan Qu, Jiantao Qiu, Yu Qiao, Runyu Peng, Tianshuo Peng, Jiahui Peng, Qizhi Pei, Zhuoshi Pan, Linke Ouyang, Wenchang Ning, Yichuan Ma, Zerun Ma, Ningsheng Ma, Runyuan Ma, Chengqi Lyu, Haijun Lv, Han Lv, Lindong Lu, Kuikun Liu, Jiangning Liu, Yuhong Liu, Kai Liu, Hongwei Liu, Zhoumianze Liu, Mengjie Liu, Ziyu Liu, Wenran Liu, Yang Liu, Liwei Liu, Kaiwen Liu, Junyao Lin, Junming Lin, Tianyang Lin, Dahua Lin, Jianze Liang, Linyang Li, Peiji Li, Zonglin Li, Zehao Li, Pengze Li, Guoyan Li, Lingkai Kong, Linglin Jing, Zhenjiang Jin, Feifei Jiang, Qian Jiang, Junhao Huang, Zixian Huang, Haian Huang, Zhouqi Hua, Han Hu, Linfeng Hou, Yinan He, Conghui He, Tianyao He, Xu Guo, Qipeng Guo, Aijia Guo, Yuzhe Gu, Lixin Gu, Jingyang Gong, Qiming Ge, Jiaye Ge, Songyang Gao, Jianfei Gao, Xinyu Fang, Caihua fan, Yue Fan, Yanhui Duan, Zichen Ding, Shengyuan Ding, Xuanlang Dai, Erfei Cui, Ganqu Cui, Pei Chu, Tao Chu, Guangran Cheng, Yu Cheng, Kai Chen, Yongkang Chen, Chiyu Chen, Guanzhou Chen, Qiaosheng Chen, Sitao Chen, Xin Chen, Haojiong Chen, Yicheng Chen, Weihan Cao, Yuhang Cao, Qinglong Cao, and Lei Bai. Intern-s1-pro: Scientific multimodal foundation model at trillion scale, 2026. 1

A. Detailed Experimental Configurations

In this section, we provide the detailed experimental settings used in our evaluation to ensure full reproducibility.

A.1. Model Configurations

For closed-source models (e.g., GPT-5.5, Claude-Opus-4.8), we used the default inference settings provided by the corresponding model APIs to ensure deterministic outputs where possible. For open-weight models (e.g., Qwen3.5-397B-A17B), we deployed them with the SGLang inference engine and followed the official recommended inference configurations for each model. For all evaluated models, whenever a reasoning-effort parameter was available, we set it to `high`. This consistent setup avoids ad hoc parameter tuning and keeps the reported comparisons reproducible.

A.2. Benchmark Specification.

In Table 3, the subscripted differences report the gap between the AgentCompass score and the closest available external reference result for the same model and benchmark. External references are drawn from the model's official technical report and the benchmark's official leaderboard when available. If both sources are available, we use the one whose reported score is closest to the AgentCompass evaluation result. If neither source reports a comparable result, no subscripted difference is shown.

Tool Use, Web & Research and Scientific Reasoning. For tool-use evaluation, we used the official workflow of τ^3-bench. For web & research and scientific-reasoning evaluation, DeepSearchQA and FrontierScience(Research) were evaluated with the AgentCompass Naive Search Agent, whereas SciCode was evaluated with the AgentCompass port of the official tool-assisted multi-turn workflow. DeepSearchQA used Qwen3.6-35B-A3B as the judge model, while FrontierScience(Research) used GPT-5.5. For SciCode, we used the official reference checkout e3158e and loaded the task package hosted by AgentCompass. Scientist-annotated background information was enabled during evaluation. We evaluated all 80 problems, which resulted in 338 scored subproblems after excluding the three official prefilled steps. The SciCode harness ran in `tool_use` mode with `code_interpreter` as the only enabled tool, allowing up to 30 tool-use iterations per generated step and imposing a 180-second timeout for each local code-interpreter execution.

Agentic Coding. For software-engineering evaluation, SWE-bench Pro and SWE-bench Multilingual were evaluated with both Mini-SWE-agent (v2.3.0) and OpenHands, where OpenHands was instantiated through the OpenHands Software Agent SDK (v1.23.0). Both agents operated on the same repository checkout and were scored through an identical patch-generation interface and unit-test-based metric. The system prompt was supplied by the corresponding harness, and the user prompt consisted of the benchmark issue description together with an instruction to write the generated patch to `patch.txt`. For SWE-bench Pro instances based on minimal container images, such as Alpine Linux or Go toolchain images, we applied runtime compatibility fixes at container startup, including the provisioning of glibc-related components, and executed OpenHands inside a micromamba-managed isolated environment.

Productivity. For productivity-oriented evaluation, we used the official benchmark releases whenever available and kept task files and judging rules fixed across harnesses. SkillsBench used the AgentCompass data version derived from official commit 17dec32 and was evaluated with the integrated OpenClaw (v2026.5.9) and OpenHands (v1.23.0) harnesses under a task-level timeout multiplier of 24. PinchBench used the official skill repository release v1.1.0, corresponding to commit e8e833, and was evaluated on the full `suite=all` setting with 23 tasks using the AgentCompass-integrated OpenClaw harness. Unless otherwise specified, PinchBench runs used OpenClaw (v2026.3.22) with a 250000-token context window, an 80000-token per-turn limit, and a maximum single-message length of 131072 characters. For Kimi-K2.6, we used OpenClaw (v2026.7.1-beta.2) because the default version did not robustly handle responses with non-empty reasoning content but empty assistant content. All PinchBench runs were judged by `claude-opus-4-5-20251101` with thinking mode enabled; task timeouts followed the benchmark-defined `timeout_seconds` values with a 10x multiplier, and the grading runner used a 480-second outer timeout to accommodate judge execution and result parsing.

Model	τ^3-bench	DeepSearchQA	FrontierSci	SciCode	PinchBench	SkillsBench	SWE-Pro	SWE-Multilingual
Qwen3.5-397B-A17B	21.56	19.21	4.94	4.26	5.26	34.45	59.42	68.31
Kimi-K2.6	20.68	23.36	8.73	4.29	5.29	38.50	57.72	50.85
DeepSeek-V4-pro(FP4)	18.40	15.37	3.49	4.06	4.26	30.83	44.35	38.55
GLM-5.2(FP8)	18.06	16.23	9.41	4.13	5.10	41.76	67.76	60.59
Gemini-3.1-Pro-Preview	48.14	27.44	9.01	4.00	4.77	43.10	76.29	99.56
GPT-5.5	20.07	21.60	11.16	4.26	6.57	33.18	46.59	31.64
Claude-Opus-4.8	17.92	10.03	1.74	4.16	5.13	21.16	30.22	19.13

Table 5: Average interaction steps per evaluation instance across different benchmarks. Each value represents the mean trajectory length over all evaluated samples for the corresponding model and benchmark.

B. Step statistics

Table 5 summarizes the average number of interaction steps required to complete one evaluation instance across different benchmarks. Here, a *step* denotes one complete agent–environment interaction cycle, including one model generation together with the corresponding tool invocation (if any) and the returned environment observation. For purely conversational benchmarks, a step is equivalent to one model response, whereas for interactive coding or productivity benchmarks, a step typically consists of one reasoning action followed by environment execution.

The statistics are computed by averaging the recorded trajectory lengths over all evaluated instances for each model and benchmark. Because AgentCompass records the complete execution trajectory in a unified format, these values are collected automatically without benchmark-specific instrumentation. Several observations can be drawn from Table 5. First, benchmarks exhibit substantially different interaction depths. Scientific reasoning tasks (e.g., SciCode) generally require only around four interaction steps due to their constrained tool workflow, whereas software engineering benchmarks frequently require tens of iterations because agents repeatedly inspect repositories, modify source files, execute tests, and refine patches. Second, model-specific differences are also evident. For example, Gemini-3.1-Pro-Preview performs considerably longer trajectories on the SWE benchmarks than other models, while Claude-Opus-4.8 typically completes tasks with fewer interaction steps. Finally, longer trajectories do not necessarily translate into higher benchmark scores. As illustrated in the main paper (Figure 4), the relationship between interaction length and evaluation performance is benchmark dependent. These trajectory-level statistics therefore provide complementary evidence beyond final task accuracy and are useful for understanding evaluation cost, interaction efficiency, and agent behavior.