Coding Agents with Multimodal Browsing are Generalist Problem Solvers

Aditya Bharat Soni [1] **Boxuan Li** [2] **Xingyao Wang** [3] **Valerie Chen** [1] **Graham Neubig** [1 3]

Abstract

Modern human labor is characterized by specialization; we train for years and develop particular tools that allow us to perform well across a variety of tasks. In addition, AI agents have been specialized for domains such as software engineering, web navigation, and workflow automation. However, this results in agents that are good for one thing and fail to generalize beyond their intended scope because agent developers provide a highly specialized set of tools or make architectural decisions optimized for a specific use case or benchmark. In this work, we ask the question: *what is the minimal set of general tools that can be used to achieve high performance across a diverse set of tasks?* Our answer is **OpenHands-Versa**, a generalist agent built with a modest number of general tools: code editing and execution, web search, multimodal web browsing and file access. Unlike existing multi-agent systems that fail to generalize, OpenHands-Versa is a single-agent system that demonstrates superior or competitive performance over leading specialized agents across three diverse and challenging benchmarks: SWE-Bench Multimodal (Yang et al., 2025), GAIA (Mialon et al., 2023), and The Agent Company (Xu et al., 2024), outperforming the best-performing previously published results with absolute improvements in success rate of **9.1**, **1.3**, and **9.1** points respectively. These results demonstrate the feasibility of developing a generalist agent to solve diverse tasks and establish OpenHands-Versa as a strong baseline for future research.

[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, USA [2]Independent [3]All Hands AI, USA. Correspondence to: Aditya Bharat Soni <adityabs@cs.cmu.edu>, Graham Neubig <gneubig@cs.cmu.edu>.

Workshop on Computer-use Agents @ ICML 2025, Vancouver, Canada. *Copyright 2025 by the author(s).*

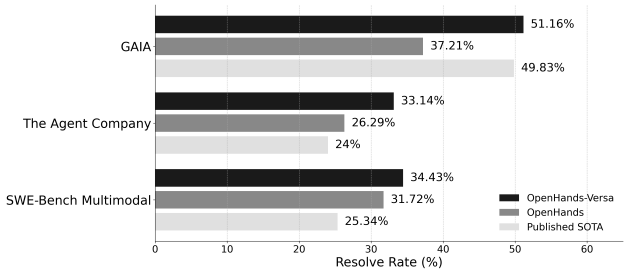

Figure 1. Comparison of OpenHands-Versa with previously published SOTA agents and OpenHands across GAIA, SWE-Bench Multimodal and The Agent Company. OpenHands-Versa outperforms the SOTA specialist agents for all three benchmarks. Notably, OpenHands-Versa improves browsing and information access abilities of OpenHands, while maintaining its software engineering capabilities. We focus on comparing to prior agents with reproducible code and results (more details in Table 2).

1. Introduction

AI agents powered by Large Language Models hold great promise to accelerate or automate a wide variety of practical tasks. For instance, agents have demonstrated strong software engineering capabilities and have been able to fix as many as two-thirds of issues in open-source Python repositories from SWE-Bench (Jimenez et al., 2024) and around one-third of issues in Javascript-based front-end libraries from SWE-Bench Multimodal (Yang et al., 2025). In addition, agents have shown impressive web navigation capabilities, and can complete over half of the tasks from WebArena (Zhou et al., 2023) and over one-third of the tasks from VisualWebArena (Koh et al., 2024). Agents have also exhibited strong performance as general assistants, solving over half of the tasks from GAIA (Mialon et al., 2023) that require various capabilities like gathering and synthesizing information from the web and processing multimodal data from diverse files. Finally, agents have also proven effective as digital workers, solving one-fourth of tasks in The Agent Company (Xu et al., 2024) that require the agent to navigate company-internal websites and communicate with co-workers.

However, until this point, the strongest published agents in each domain have typically been explicitly optimized to perform well on a relatively narrow set of tasks and benchmarks,

which we refer to as *specialist agents*. For example, agents such as Agentless (Xia et al., 2024) and SWE-Agent (Yang et al., 2024a) have achieved state-of-the-art performance on SWE-Bench Python programming problems. Still, they cannot typically gather information from the web or communicate with co-workers via web-based chat platforms, which would result in poor performance on GAIA and The Agent Company respectively. In contrast, strong web navigation agents like AgentSymbiotic (Zhang et al., 2025), AgentOccam (Yang et al., 2024b), and Agent Workflow Memory (Wang et al., 2024b) cannot write, debug, or execute code, so an agent with strong performance on WebArena may not be proficient on software engineering problems (e.g., SWE-Bench). More concretely, we refer to such agents as *specialist agents* that either lack one or more capabilities, such as code execution, browsing, file viewing, search APIs, and have been primarily evaluated on a narrow category of tasks (i.e., one of browsing, coding, general assistance benchmarks).

Does this mean that we are destined for a world where each user must interact with a broad variety of agents, each specialized for a single task? In this paper, we argue that the answer to this is *no*. How could this be the case? We argue that a great majority of tasks can be tackled by agents that have the below three capabilities.

- **Coding:** The ability to write, debug, and execute code, including the use of libraries that are available to programmers.
- **Multimodal Web Browsing:** The ability to browse the web, perform interactive actions (e.g., click, type) on webpages, and process vision and text modalities from webpages.
- **Information Access:** The ability to search information on the web, typically using search APIs, and process multimodal content from various files such as PDFs, spreadsheets, code files, etc.

To implement such an agent, we propose **OpenHands-Versa**, built using the popular OpenHands framework (Wang et al., 2024a) for coding agents, imbuing it with the ability to perform visual browsing, accessing information from the web through search APIs, and processing multimodal content from diverse files.

This simple strategy is surprisingly effective—we show a single agent can achieve strong results, rivaling or exceeding the state-of-the-art published systems, on three diverse and practical benchmarks: GAIA for general assistance and information access (Mialon et al., 2023), The Agent Company for evaluating agents as digital co-workers in a company (Xu et al., 2024), and SWE-Bench Multimodal for frontend-focused software engineering (Yang et al., 2025), as shown in Figure 1. Notably, OpenHands-Versa achieves state-of-the-art performance on all three benchmarks with absolute improvement of **9.1**, **1.3**, and **9.1** points in success rate over best-performing previously published results on SWE-Bench Multimodal, GAIA and The Agent Company respectively. Furthermore, OpenHands-Versa improves the browsing and information access abilities of OpenHands while retaining its coding capabilities. Furthermore, we also find that current state-of-the-art multi-agent systems fail to generalize beyond their intended scope.

We also study the tool-use patterns of OpenHands-Versa and provide insights into why this simple strategy works so well. We find that OpenHands-Versa uses appropriate tools that align well with task requirements and has better domain-aware tool-selection than OpenHands. We also perform extensive analysis of the results and find some highly complex tasks that can be solved by OpenHands-Versa, while also pointing out error behaviors of our agent that can be addressed by future work. Finally all our code and experimental scripts are open-source for future development[1].

2. Towards a Generalist Agent

2.1. Preliminaries

To implement a generalist agent, we choose to build OpenHands-Versa on top of the OpenHands (Wang et al., 2024a) framework.[2] OpenHands offers a flexible event stream architecture, a sandboxed runtime, a built-in evaluation harness for evaluating agents on numerous benchmarks, and the following tools:

1. A bash shell that connects to the operating system environment and supports the execution of Unix-style commands.
2. Interactive python code execution via a Jupyter IPython server.
3. A text-based browsing tool that uses a Chromium browser based on Playwright[3] and uses the BrowserGym framework (de Chezelles et al., 2025) to implement its action space.
4. A file-processing tool for creating, viewing and editing plain-text files (e.g., files with extensions like .py, .txt, .cpp, .js, .json etc.).

2.2. Ingredients of Our Agent

Since OpenHands has primarily been an agent developed for software engineering, with strong **coding abilities** and support for multiple programming languages (Zan et al., 2025), it lacks other capabilities like multimodal web browsing and information access. We augment OpenHands with

[1] OpenHands-Versa is available open-source at: https://github.com/adityasoni9998/OpenHands-Versa

[2] We use OpenHands v0.28.1.

[3] https://playwright.dev/python/

these capabilities while inheriting the coding capabilities from OpenHands.

Multimodal Web Browsing: The browsing tool in OpenHands relies solely on text-based observations that represent web pages using its accessibility tree (AXTree)[4], and misses critical visual cues from the frontend. Instead, we adopt the Set-of-Marks prompting method (Yang et al., 2023), which captures a screenshot of the current viewport (the visible area of a webpage in the browser window), overlays bounding boxes on interactable elements (e.g., buttons, links, text boxes), and labels them with unique alphanumeric browsergym-ids (de Chezelles et al., 2025) (e.g., refer to Appendix A). Note that this is similar to BrowserGym's GenericAgent (de Chezelles et al., 2025). We also include the full AXTree to provide context beyond the viewport but truncate to the current viewport if the AXTree is too large for the LLM's context window.

We also incorporate context condensation into the browser tool. OpenHands uses an event stream architecture wherein the backbone LLM is provided with action-observation pairs from all the previous steps at the next execution step. Since each browser observation consists of a webpage screenshot and a possibly large AXTree, this approach results in high inference costs, large observations that do not fit in the LLM's context window, and increases the agent's runtime due to slower LLM inference. To address this, we implement a browsing condenser that retains only the k most recent browsing observations and masks out each older browsing observation with a fixed placeholder message.

Information Access: We discuss two ways we improve information access: *web search* and *multimodal file processing*. First, synthesizing information from the web using search engines is crucial. While agents can achieve this by opening search engines like Google or Bing in the browser, in practice, we observe that the agent is frequently blocked by CAPTCHAs (von Ahn et al., 2003). We mitigate this by allowing the agent to perform a web search using a search API. This has the added advantage of reduced costs over browsing the web when searching for factual information and allows the use of specialized search APIs designed for agents. We use the Tavily API (Tavily-AI) for most of our experiments; however, OpenHands-Versa also supports the use of Exa (Exa) and Brave (Inc) APIs.

Second, several tasks require the agent to access information about multimodal data from various files such as PDFs, audio files, presentation slides, etc. However, OpenHands has a limited file viewing support restricted to files that can be opened in text editors (like those with .txt, .py, .json, .js extensions). We enhance the existing file viewing functionality of OpenHands by integrating Markdown converters, similar to those used by the FileSurfer in Magentic-One (Fourney et al., 2024), to transform various files into a unified Markdown representation.

Task Planning: Task planning is crucial for multi-step execution, where agents must decompose the end-goal into multiple sub-tasks and organize their actions into a logical sequence. Prior approaches include developing an orchestrator or a planning agent (Fourney et al., 2024; Bahdanau et al., 2024), a Think tool (Anthropic, 2025) that the agent can flexibly invoke to log its plan, and using Chain-of-Thought prompting (de Chezelles et al., 2025). We rather use a simple approach of appending a fixed planning prompt to the agent's event stream after every τ steps asking the agent to summarize its current progress and create a plan for the subsequent task execution.

2.3. Comparison with Existing Agents

Next, we compare OpenHands-Versa with existing agents that excel in specific domains and benchmarks. For coding agents, we consider SWE-Agent (Yang et al., 2024a) and Agentless (Xia et al., 2024). BrowserGym GenericAgent (de Chezelles et al., 2025) and Browser-use (Müller & Žunič, 2024) are well-suited for web navigation. Multi-agent frameworks like Magentic-One (Fourney et al., 2024), OpenDeepResearch (Roucher et al., 2025), and OWL-roleplaying (Hu et al., 2025) excel at general-purpose assistance by synthesizing information from the web and processing various files. We also include OpenHands in this comparison. Note that we only consider agents with open-source implementations, since this comparison requires understanding of their internal design.

The comparison of these agents with OpenHands-Versa is mainly along three core capabilities defined in §1. Also, we examine whether the system adopts a multi-agent framework with specialized agents for distinct skills (such as web navigation, coding, and planning), or a single-agent framework, where one agent utilizes all available tools to complete the task. Table 1 captures nuanced differences between these agents, which are described below.

Coding: For software engineering (SWE) tasks, agents must have the ability to write and execute code, debug code by editing files, and use a shell to run tests, install packages, and navigate the repository. Browser-use and BrowserGym GenericAgent are designed exclusively for web navigation and lack all code-related abilites. Multi-agent systems like OWL-roleplaying, OpenDeepResearch, and Magentic-One support a subset of the these abilities but they lack support for editing files, implying that the agent has to regenerate the entire code from scratch when making any changes to existing files. Also OWL-roleplaying and OpenDeepResearch do not have access to a shell. These multi-agent systems

[4] https://developer.mozilla.org/en-US/docs/Glossary/Accessibility_tree

Table 1. Comparison of different agents based on their tool-use capabilities. Definitions for the symbols - ⟨/⟩ : supports executing code, ✎ : supports editing operations, ➢ : uses textual browsing, 📷 : uses visual browsing, 🔍 : supports API-based search, ≡ : supports viewing multimodal file content, 🗎 : supports viewing only plain-text files, ⊗ : capability not supported, 👤 : uses a single agent framework, 👥 : uses a multi-agent framework.

Method	Coding	Browsing	Search	File Viewing	Agents
SWE-Agent (Yang et al., 2024a)	⟨/⟩ ✎	⊗	⊗	🗎	👤
Agentless (Xia et al., 2024)	✎	⊗	⊗	🗎	👤
OpenHands (Wang et al., 2024a)	⟨/⟩ ✎	➢	⊗	🗎	👤
BrowserGym GenericAgent (de Chezelles et al., 2025)	⊗	📷	⊗	⊗	👤
Browser-use (Müller & Žunič, 2024)	⊗	📷	⊗	⊗	👤
OpenDeepResearch (Roucher et al., 2025)	⟨/⟩	➢	🔍	≡	👥
OWL-roleplaying (Hu et al., 2025)	⟨/⟩	📷	🔍	≡	👥
Magnetic One (Fourney et al., 2024)	⟨/⟩	📷	🔍	≡	👥
OpenHands-Versa (Ours)	⟨/⟩ ✎	📷	🔍	≡	👤

mainly support the execution of stand-alone Python programs, making them unsuitable for SWE tasks and coding in other programming languages. Agents like SWE-Agent, OpenHands, and OpenHands-Versa support all the above code-related capabilities and are well-suited for SWE tasks. Although Agentless does not have a bash terminal, it supports the execution of selected tests in the repository to validate candidate patches within a human-defined workflow.

Web Browsing: Agents should be able to browse the web and execute interactive actions on websites to handle tasks such as filling online forms, ordering items from e-commerce websites and reading software documentation. Agents must also have a strong multimodal processing ability to comprehend the webpage content by jointly interpreting the visual layout (i.e., the rendered UI elements) and the webpage text. SWE-agent and Agentless do not support browsing, making them unsuitable for many practical tasks. OpenDeepResearch has a very limited browsing ability, wherein it can only open and scroll through webpages, without the ability to execute any other actions like "click" or "type". OpenHands supports interactive browsing actions, but it performs text-only browsing whereas all other agents perform mulimodal web browsing using visual context from webpage screenshots.

Information Access via Web Search: Agents must be able to query search engines using keywords to retrieve relevant URLs, synthesize factual information, and access up-to-date content. Search APIs provide a more robust mechanism for supporting this functionality and mitigate issues caused by access restrictions like CAPTCHAs. Despite this tool being useful for several real-world tasks, most existing agents do not have this ability except multi-agent systems that demonstrate strong performance on the GAIA benchmark (Mialon et al., 2023). This provides further evidence that many agent designs are over-tailored to specific domains or benchmarks.

Information Access via Multimodal File Processing: Agents must be able to view the content of various file types such as PDFs, presentation slides, spreadsheets etc. Although the agent can also read certain files using code or shell, this approach is prone to bugs and may require multiple attempts for successfully parsing the file. Supporting file viewing as a tool is a more robust approach since the agent can access content of various files through a single tool call. Web agents like Browser-Use and BrowserGym Generic Agent do not support file viewing. SWE-Agent, Agentless, and OpenHands have limited file-viewing support wherein the agent can only read plain-text files. All other agents have specific tool(s) that allow the agent to process multimodal file content.

3. Experimental Setup

In this section, we describe our experimental setup to demonstrate the effectiveness of OpenHands-Versa. We overview our choice of evaluation benchmarks and the corresponding evaluation metrics in §3.1, and discuss our baselines in §3.2.

3.1. Evaluation Benchmarks

We evaluate OpenHands-Versa on three benchmarks that cover a diverse range of capabilities and agent use cases—which can be roughly gleaned from Figure 2. We provide some example tasks for each benchmark in §D.

SWE-Bench Multimodal (SWE-Bench M) (Yang et al., 2025) : This benchmark evaluates the ability of agents to fix software issues in GitHub repositories of front-end libraries. The benchmark requires the agent to solve GitHub issues from 17 popular JavaScript code repositories for various use-cases like web development, syntax highlighting, and data visualization. Furthermore, several tasks also have visual assets (images and videos) describing the issue and links to online integrated development environments (IDEs) containing code snippets for reproducing the issue, requiring

agents to process multi-modal data to visually analyze the issue. Unlike SWE-Bench (Yang et al., 2025), where all the reference solutions only require editing Python files, more than 28% of SWE-Bench M instances require editing files across two or more programming languages.

GAIA (Mialon et al., 2023): This benchmark evaluates AI agents as general-purpose assistants using tasks that require browsing the open web, performing web search, coding, reasoning, and processing multimodal content from audios, spreadsheets, and PDFs. While the coding tasks in SWE-Bench M require agents to fix issues by editing *existing* code files, the coding tasks in GAIA generally require the agent to write and execute stand-alone programs from scratch.

The Agent Company (Xu et al., 2024): This benchmark evaluates the ability of agents as digital co-workers in a simulated software company using a reproducible, self-hosted environment. It covers tasks across software development, project management, data science, financial analysis, etc. It uses four self-hosted websites: GitLab for hosting code repositories and documentation, OwnCloud for cloud-based file-sharing, Plane for issue tracking and project management, and RocketChat for communicating with simulated co-workers. To solve the tasks in this benchmark, the agent must be able to browse websites, write code, communicate with simulated colleagues, and read, write and edit various files.

3.2. Baseline Agents

For each benchmark, we compare to the best-performing open source agent frameworks (from the benchmark's leaderboard) that have reproducibility guidelines [5].

For **SWE-Bench Multimodal**, we choose Agentless-Lite (Dunn, 2025), and SWE-Agent (Yang et al., 2024a) along with its Multimodal and Javascript variants proposed along with this benchmark. For **GAIA**, we consider Magentic-One (Fourney et al., 2024) and OpenDeepResearch (Roucher et al., 2025). For **The Agent Company**, we consider OWL-roleplaying and OpenHands v0.14.2, which is the version used in the original paper. Across all benchmarks, we evaluate OpenHands v0.28.1—the agent on top of which OpenHands-Versa is built—to understand the effect of our modifications.

Most baseline agents report performance on only one of the benchmarks, and their architecture typically does not support evaluation on the others (as discussed in § 2.3). Agentless-lite and all SWE-agent variants cannot browse, use search engines, or process multimodal files, which are crucial for GAIA and The Agent Company. Multi-agent baselines for GAIA cannot typically write code in languages other than Python (like JavaScript), making them unsuitable for SWE-Bench M(§2.2).

We used `claude-3-7-sonnet-20250219` as the backbone LLM of OpenHands-Versa and OpenHands v0.28.1 to ensure a direct comparison between the two agent architectures using the same LLM. We also evaluate OpenHands-Versa with the recently released `claude-sonnet-4-20250514` model. Further experimental details are provided in §B..

3.3. Evaluation metrics

For all the benchmarks, we use the evaluation metrics proposed by the authors of the corresponding benchmarks. We use resolve rate for GAIA and SWE-bench M – the % of tasks completed successfully by the agent. The Agent Company has checkpoint-based evaluation wherein two metrics are computed: **Full completion score** (the % of tasks for which all checkpoints were resolved) and **Partial completion score** (also provides partial credit for successful checkpoints in partially completed tasks). We refer the reader to The Agent Company (Xu et al., 2024) for more details. We use the test split for all three benchmarks (The Agent Company does not have a validation split). Furthermore, we report performance on GAIA validation split in §C. Finally, we report **pass @ 1** metrics for all benchmarks.

4. Main Results

We present our experimental results in Table 2 and highlight the key takeaways below. Furthermore, we report inference costs of various agents in §E

OpenHands-Versa outperforms or matches existing agents for all three benchmarks: Notably, OpenHands-Versa achieves state-of-the-art or close to state-of-the-art performance on all three benchmarks with **51.16%** resolve rate on GAIA, **34.43%** resolve rate on SWE-Bench M, and **33.14%** full completion score on The Agent Company with claude-sonnet-4 as the backbone LLM. OpenHands-Versa outperforms the strongest baseline for GAIA with an absolute improvement of **1.33** points. On GAIA, OpenHands-Versa outperforms complex, multi-agent systems which use specially designed agents for distinct skills/sub-tasks, with each agent using a separate LLM. In addition, OpenHands-Versa achieves state-of-the-art performance on The Agent Company with an absolute improvement of **6.9** points in the full completion score and **6.8** points in the partial completion score over the best-performing baseline. On SWE-Bench M, OpenHands-Versa demonstrates an absolute improvement in resolve rate of **9.09** points over Agentless-Lite and more than **22** points over SWE-Agent and its variants. Notably,

[5] On the GAIA leaderboard there are other methods with no code or technical details published with scores of up to 80%. We focus on methods that have available details and reproducible code bases.

Table 2. Comparison of agent performance across GAIA, SWE-bench M, and The Agent Company. Highest metrics for each benchmark are **bold-faced** and second highest metrics are underlined. When available, we report the metrics directly as mentioned by the baseline agents on the respective benchmark leaderboards. We restrict our comparison to agents with open-source implementations and method description.

Agent	Model(s)	GAIA	SWE-bench M	The Agent Company	
				Full	Partial
Magentic-One (Fourney et al., 2024)	gpt-4o, o1	37.87%	-	-	-
OpenDeepResearch (Roucher et al., 2025)	o1	49.83%	-	-	-
SWE-Agent (Yang et al., 2024a)	gpt-4o	-	11.99%	-	-
	claude-3.5 sonnet	-	12.19%	-	-
SWE-Agent JS (Yang et al., 2025)	gpt-4o	-	9.28%	-	-
	claude-3.5 sonnet	-	11.99%	-	-
SWE-Agent Multimodal (Yang et al., 2025)	gpt-4o	-	12.19%	-	-
	claude-3.5 sonnet	-	11.41%	-	-
Agentless-Lite (Dunn, 2025)	claude-3.5 sonnet	-	25.34%	-	-
OWL-roleplaying (Hu et al., 2025)	gpt-4o, o3-mini	-	-	4.00%	11.04%
OpenHands v0.14.2 (Wang et al., 2024a)	gpt-4o	-	-	8.60%	16.70%
	gemini-2.0 flash	-	-	11.40%	19.00%
	claude-3.5 sonnet	-	-	24.00%	34.40%
OpenHands v0.28.1 (Wang et al., 2024a)	claude-3.7 sonnet	37.21%	31.72%	26.29%	36.41%
OpenHands-Versa	claude-3.7 sonnet	**51.16%**	31.33%	30.86%	40.18%
	claude-sonnet-4	**51.16%**	**34.43%**	**33.14%**	**43.19%**

these gains are achieved without specific optimizations for SWE-Bench M, such as the JavaScript linter in SWE-Agent JS and SWE-Agent Multimodal.

OpenHands-Versa has stronger browsing and information access capabilities than OpenHands, while retaining its coding capabilities: While attempting to improve the browsing and information access capabilities in OpenHands-Versa (§2.2), it is also crucial to ensure that our changes do not cause regression in the coding abilities inherited from OpenHands. This is concretely validated by comparing the results of the two agents on the three evaluation benchmarks. When using the same backbone LLM (claude-3.7 sonnet), OpenHands-Versa significantly outperforms OpenHands on GAIA with an absolute improvement of **13.9** points in the resolve rate. Furthermore, for The Agent Company, OpenHands-Versa achieves an absolute improvement of **4.6** points in the full completion score and **3.8** points in the partial completion score over OpenHands. In addition, OpenHands-Versa achieves a nearly equal resolve rate on SWE-Bench M as that of OpenHands, with an absolute difference of only 0.39 points.

Multi-agent systems with strong performance on GAIA fail to generalize: OWL-roleplaying is a complex multi-agent system with separate agents for browsing, planning, web search etc. It is one of the top performing agents on GAIA validation set with a 58.18% resolve rate, but does not report performance on GAIA test set. However it fails to generalize to The Agent Company with a poor full completion score of **4%** and partial completion score of **11.0%**. OWL-roleplaying significantly underperforms OpenHands-Versa, with an absolute decrease in **29.1** points in the full completion score and **32.2** points in the partial completion score.

5. What Went Right and What Went Wrong?

In this section, we provide fine-grained analyses to understand the agent behaviour for different tasks. Since we observe that OpenHands-Versa outperforms or matches OpenHands on the three benchmarks, with a minimal set of

Figure 2. Distribution of the different tools used by OpenHands and OpenHands-Versa. OpenHands-Versa adapts its tool usage to different benchmarks without any benchmark-specific optimizations and OpenHands-Versa has better domain-aware usage of its tools as compared to OpenHands. Tools with **bold-faced** names have been modified/created by our work.

changes, we first compare their tool use patterns (§5.1). Next, we perform a comprehensive error analysis of our agent, to understand its limitations and provide insights for future improvement (§5.2). Finally, we also discuss the effect of the search API on downstream agent performance for GAIA (§5.3).

5.1. Tool Use Patterns across Benchmarks

To better understand the behavior of OpenHands-Versa compared to the original OpenHands (when using the same LLM claude-3.7 sonnet), we plot the distribution of the relative tool use frequencies (as a percentage of total tool calls made by the agent) across all tasks, for all the 3 benchmarks in Figure 2. This figure presents some interesting insights into the behavior of OpenHands-Versa and OpenHands, which we describe below:

OpenHands-Versa uses appropriate tools that align well with task requirements: Our analysis shows that OpenHands-Versa generally selects tools that intuitively align with various tasks. For GAIA, the agent primarily uses the browser and search engine, consistent with the need to synthesize web-based information, and makes limited use of file-editing tools while frequently executing standalone Python code via IPython. It uses the bash tool in creative ways. For example, to install packages and download files with `wget` – highlighting flexible problem-solving. In The Agent Company, tool usage is dominated by the browser, which reflects the benchmark's focus on navigating internal websites, with little reliance on the search engine or IPython since URLs of company websites are known to the agent and tasks involve editing code from repositories. For SWE-Bench M, the agent frequently uses bash, edit_file, and view_file tools, in line with practical software engineering workflows, and leverages the browser to visually verify its changes by rendering HTML files, demonstrating a nuanced understanding of front-end development practices.

OpenHands-Versa has better domain-aware tool-selection than OpenHands: For GAIA, OpenHands relies more heavily on the browser in the absence of a search engine, due to which it frequently navigates to hallucinated or invalid URLs. OpenHands-Versa first uses the search engine to retrieve relevant links and then chooses a URL based on the retrieved snippets, resulting in more targeted navigation. For SWE-Bench Multimodal, while overall tool usage patterns are similar, OpenHands-Versa makes more frequent use of the browser for visual verification of front-end changes, a capability that OpenHands cannot exploit due to text-only browsing. For The Agent Company, both agents display similar tool-use behavior, which is expected since changes to the browser tool in OpenHands-Versa primarily improve browsing observations rather than changing or expanding its action space.

5.2. Error Analysis

Next, we manually analyse the trajectories of OpenHands-Versa, describe its error behaviors, and provide some examples in Table 3.

For GAIA, we use the validation split since the ground truth is not available for the test set. We find that OpenHands-Versa is sometimes over-reliant on the retrieved summaries/snippets from the webpage given by the search API and uses factually incorrect information. We also find that the agent cannot access some websites due to various security measures like CAPTCHAs. For tasks in the SWE-Bench M, we find that the agent frequently struggles at creating comprehensive tests to verify its code, and prematurely exits, assuming that its code is correct since its non-exhaustive tests pass. Sometimes, the agent does not execute tests given in the repository to verify if its changes did not break existing functionality. For The Agent Company, we find that the agent generally struggles when interacting with OwnCloud, and frequently gets stuck in loops. Furthermore, we find that the agent sometimes prematurely exits without satisfying all the task requirements for the more complex tasks.

5.3. Effect of Search API on GAIA

Since 40% of the tool calls made by OpenHands-Versa for GAIA are to query the search engine with minimal use of this tool for other 2 benchmarks, we study the effect of choosing different search APIs on the downstream performance for this benchmark. We evaluated OpenHands-Versa on the GAIA validation split using three search APIs: Brave, Exa, and Tavily and use `claude-3-7-sonnet-20250219` for our experiments.

Although seemingly unimportant, the choice of search API significantly impacts downstream performance. We observe considerable variations in the resolve rate with **56.96%**

Table 3. Example tasks for some observed error behaviors of OpenHands-Versa.

Benchmark	**Task Description (irrelevant details truncated)**	**Observed behaviour**
GAIA	The Latin root of the Yola word "gimlie" shares a spelling with a Spanish word. What is the Google translation of the source title for the 1994 example sentence for that word in the Collins Spanish-to-English dictionary online?	Agent cannot access Collins dictionary website due to CAPTCHAs.
	In April of 1977, who was the Prime Minister of the first place mentioned by name in the Book of Esther?	Agent relies on incorrect search engine summary when searching for the first place given in the book.
The Agent Company	We are collecting employees' preferences on drinks. Please navigate to ownCloud and find drinks_survey.pdf and tell 3 most popular drinks to HR manager via RocketChat.	Agent gets stuck in a loop and fails to find the file on ownCloud.
	In Plane there open issues in the JanusGraph project. I want you to add all "In Progress" issues to Gitlab.	Agent fails to copy all issues and exits after partial completion of task.
SWE-Bench M	Happiness Support card needs preventWidows treatment in WordPress. Steps to reproduce ... What I expected ... What happened instead ...	Agent does not write tests to verify its fix and does not follow steps to reproduce the bug.
	WebGL: render buffers are not always created correctly. The issue is that when creating a retained-mode geometry...	Agent does not execute existing tests in the repository due to which its changes fail the Pass-to-Pass tests.

when using Brave, **58.18%** when using Exa and **64.24%** when using Tavily APIs. Notably, switching from Brave to Tavily results in an absolute improvement of **7.28** points in resolve rate. Our analysis shows that the agent relies on search snippets to decide which webpages to open for obtaining information. Brave extracts these snippets from raw webpage text, while Exa and Tavily provide higher-quality LLM-generated summaries. These often eliminate the need to open webpages in the browser, reducing inference costs due to large browsing observations as compared to the compact search results. However, reliance on these summaries occasionally introduces hallucinations when they contain inaccuracies. Tavily partially mitigates these by offering an LLM-generated answer per query, synthesized from all retrieved results, which tends to be more accurate than individual page summaries. This is also one of the primary reasons why using Tavily has a higher resolve rate than other APIs.

6. Conclusion

In this work, we propose OpenHands-Versa– a simple and flexible agent that demonstrates strong performance across three benchmarks – GAIA, SWE-Bench M and The Agent Company. Our experimental results demonstrate the effectiveness of OpenHands-Versa in tasks across various domains and highlight that generalizability can be achieved using a simple and intuitive agent design without developing specialized agent implementations over-optimized for a particular domain. More concretely, these results indicate that generalist agents can be designed by simply providing the necessary tools to the backbone LLM and leaving it for the LLM to autonomously decide how to use these tools to solve the task. Our results also demonstrate why existing agents fail to generalize beyond their target domain. We elaborate on the limitations of our approach in §F. In conclusion, OpenHands-Versa will serve as a strong baseline for future research on generalist agents.

Impact Statement

AI agents have shown promise in addressing complex tasks, but still face significant limitations when confronted with real-world challenges. Our research advances the field by enhancing the generalizability of these systems and improving their performance across diverse practical applications. The work establishes a robust foundation for future developments in AI agents. However, these advancements bring important societal considerations. As AI agents become more sophisticated, potential risks emerge, including misuse for illegal activities, labor market disruption as agents become capable of performing complex tasks, and questions of governance to ensure responsible deployment. Future work should focus not only on enhancing agent capabilities but also on developing appropriate safeguards and ethical frameworks to guide real-world deployments.

References

Anthropic(2025). The "think" tool: Enabling claude to stop and think in complex tool use situations. URL https://www.anthropic.com/engineering/claude-think-tool. A new tool that improves Claude's complex problem-solving performance.

Bahdanau, D., Gontier, N., Huang, G., Kamalloo, E., Pardinas, R., Piché, A., Scholak, T., Shliazhko, O., Tremblay, J. P., Ghanem, K., et al. Tapeagents: a holistic framework for agent development and optimization. *arXiv preprint arXiv:2412.08445*, 2024.

de Chezelles, T. L. S., Gasse, M., Lacoste, A., Caccia, M., Drouin, A., Boisvert, L., Thakkar, M., Marty, T., Assouel, R., Shayegan, S. O., Jang, L. K., Lù, X. H., Yoran, O., Kong, D., Xu, F. F., Reddy, S., Neubig, G., Cappart, Q., Salakhutdinov, R., and Chapados, N. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

Dunn(2025). Agentless-lite. URL https://github.com/sorendunn/Agentless-Lite.

Exa. Exa search api. URL https://exa.ai/exa-api.

Fourney, A., Bansal, G., Mozannar, H., Tan, C., Salinas, E., Niedtner, F., Proebsting, G., Bassman, G., Gerrits, J., Alber, J., et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

Hu, M., Zhou, Y., Fan, W., Nie, Y., Xia, B., Sun, T., Ye, Z., Jin, Z., Li, Y., Zhang, Z., Wang, Y., Ye, Q., Luo, P., and Li, G. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation, 2025. URL https://github.com/camel-ai/owl.

Inc, B. S. Brave search api. URL https://brave.com/search/api/.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. R. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Koh, J. Y., Lo, R., Jang, L., Duvvur, V., Lim, M. C., Huang, P.-Y., Neubig, G., Zhou, S., Salakhutdinov, R., and Fried, D. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.

Mialon, G., Fourrier, C., Wolf, T., LeCun, Y., and Scialom, T. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

Müller, M. and Žunič, G. Browser use: Enable ai to control your browser, 2024. URL https://github.com/browser-use/browser-use.

Roucher, A., del Moral, A. V., Noyan, M., Wolf, T., and Fourrier, C. Open-source deepresearch – freeing our search agents, 2025. URL https://huggingface.co/blog/open-deep-research.

Tavily-AI. Tavily search api. URL https://tavily.com/.

von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. Captcha: Using hard ai problems for security. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pp. 294–311. Springer, 2003. doi: 10.1007/3-540-39200-9_18. URL https://iacr.org/archive/eurocrypt2003/26560294/26560294.pdf.

Wang, X., Li, B., Song, Y., Xu, F. F., Tang, X., Zhuge, M., Pan, J., Song, Y., Li, B., Singh, J., et al. Openhands: An open platform for ai software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2024a.

Wang, Z. Z., Mao, J., Fried, D., and Neubig, G. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024b.

Xia, C. S., Deng, Y., Dunn, S., and Zhang, L. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint*, 2024.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

Xu, F. F., Song, Y., Li, B., Tang, Y., Jain, K., Bao, M., Wang, Z. Z., Zhou, X., Guo, Z., Cao, M., Yang, M., Lu, H. Y., Martin, A., Su, Z., Maben, L., Mehta, R., Chi, W., Jang, L., Xie, Y., Zhou, S., and Neubig, G. Theagentcompany: Benchmarking llm agents on consequential real world tasks, 2024. URL https://arxiv.org/abs/2412.14161.

Yang, J., Zhang, H., Li, F., Zou, X., Li, C., and Gao, J. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023. URL https://arxiv.org/abs/2310.11441.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K. R., and Press, O. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://arxiv.org/abs/2405.15793.

Yang, J., Jimenez, C. E., Zhang, A. L., Lieret, K., Yang, J., Wu, X., Press, O., Muennighoff, N., Synnaeve, G., Narasimhan, K. R., Yang, D., Wang, S., and Press, O. SWE-bench multimodal: Do AI systems generalize to visual software domains? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=riTiq3i21b.

Yang, K., Liu, Y., Chaudhary, S., Fakoor, R., Chaudhari, P., Karypis, G., and Rangwala, H. Agentoccam: A simple yet strong baseline for llm-based web agents. *arXiv preprint arXiv:2410.13825*, 2024b.

Zan, D., Huang, Z., Liu, W., Chen, H., Zhang, L., Xin, S., Chen, L., Liu, Q., Zhong, X., Li, A., Liu, S., Xiao, Y., Chen, L., Zhang, Y., Su, J., Liu, T., Long, R., Shen, K., and Xiang, L. Multi-swe-bench: A multilingual benchmark for issue resolving, 2025. URL https://arxiv.org/abs/2504.02605.

Zhang, R., Qiu, M., Tan, Z., Zhang, M., Lu, V., Peng, J., Xu, K., Agudelo, L. Z., Qian, P., and Chen, T. Symbiotic cooperation for web agents: Harnessing complementary strengths of large and small llms. *arXiv preprint arXiv:2502.07942*, 2025.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://webarena.dev.

A. Webpage Screenshot with Set-of-Marks Annotation

Figure 3 is an example screenshot of a webpage with all the interactable elements annotated with bounding boxes and their corresponding browergym-ids.

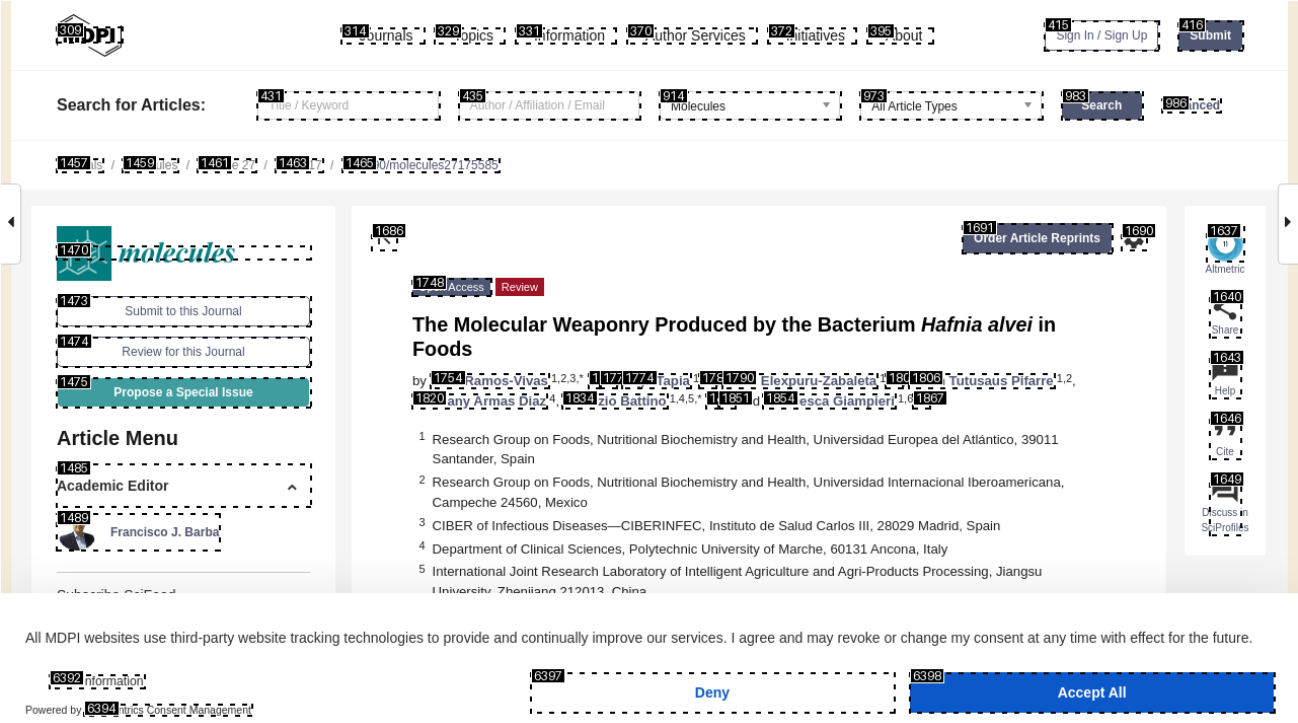

Figure 3. Example screenshot of a webpage with set-of-marks annotation

B. Experimental Setup

In this section, we provide more details about our experimental setup.

First, we discuss the exact configuration used for OpenHands-Versa. For browsing condensation (§2.2), we set the context window (k) to 1 implying that we only retain the most recent browsing observation in the event stream. For planning, we set the planning interval (τ) to 10, which implies that we append the planning prompt (§2.2) to the event stream after every 10 steps. Notably, we use identical agent implementation for all the three benchmarks as opposed to other agents that selectively choose only relevant tools for different benchmarks, develop specialized tools to improve performance on a specific benchmark, or use benchmark-specific or domain-specific system prompts that will not generalize to all scenarios. For example, OWL-roleplaying provides tools to search Wikipedia and the Wayback Machine[6], which are particularly useful for GAIA since many tasks require the agent to search for factual information from Wikipedia and some tasks refer to websites that are no longer publicly available, requiring the agent to access them via the Wayback Machine.

We set the temperature of the backbone LLM to 0 for all our experiments. We limit the maximum number of steps allowed for the agent to 100 for SWE-Bench M and The Agent Company, and to 60 for GAIA. Since GAIA requires the final answer given by the agent to exactly match with the ground truth answer, we extract the final answer of the agent using an LLM (particularly `claude-3-7-sonnet-20250219`) giving it the task description and the final thought of the agent. This also helps with some output formatting errors. For example, the agent may write the answer numerically (for eg. 500), whereas the task asks the agent to write it in text (i.e five hundred).

All our experiments are run using CPU-only, cloud-based machines (AWS EC2 instances – t3.2xlarge specification with

[6] `https://archive.org`

32GB RAM, 8 vCPUs, and 512GB disk space). However, they can also run on local computers and do not require any GPU resources. The total runtime for evaluating OpenHands-Versa and OpenHands is ≈24 hours for GAIA, ≈54 hours for The Agent Company, and ≈12 hours for SWE-Bench M. Also, evaluating OWL on TAC takes ≈50 hours.

B.1. Baseline Agents

Next, we provide more details about the baseline agents used in our work.

Magentic-One(Fourney et al., 2024) is a generalist multi-agent system that uses an LLM-based Orchestrator Agent responsible for planning, tracking progress, and querying other agents/tools for different sub-tasks. Orchestrator can issue commands to WebSurfer, Coder, FileSurfer and ComputerTerminal. WebSurfer is an LLM-based agent responsible for web browsing and searching the web using Bing. Coder is an LLM-based agent that can write a new stand-alone Python program for each request and it should regenerate the entire code from scratch when debugging the code it previously wrote. The Orchestrator can read various files using the FileSurfer that converts different files in a unified Markdown format. Finally, the Orchestrator can run Unix-style commands in a shell using the ComputerTerminal tool. This system does not have native support to create or edit files and write code in other programming languages.

OpenDeepResearch (Roucher et al., 2025) is a multi-agent system similar to Magentic-One. Its CodeAgent can write and execute stand-alone Python programs, read different files similar to FileSurfer in Magentic-One, ask questions about files, videos, and images to an LLM-based file viewer, and delegate browsing tasks to a separate browsing agent. The browsing agent uses a text-only browser to view webpages. It can only scroll on the webpage and search for text on a webpage, but cannot perform other actions like click, type, hover, etc. The browsing agent has tools to search the web using APIs and search the Wayback machine for archived webpages. The CodeAgent and the browsing agent each have their own planner agents that analyze their progress after every few steps and create a step-by-step plan. The CodeAgent is restricted to a fixed set of pre-installed libraries/packages that it can use. There is no native support for using a bash shell, writing and editing files, and executing code in other programming languages.

OWL-roleplaying (Hu et al., 2025) is a multi-agent system similar to Magentic-One and OpenDeepResearch. It has a user agent that assists with the task, creates plans, and issues commands to the assistant agent. The assistant agent is responsible for solving the task and has access to the various tools to extract content from different files, query LLMs to analyze images, videos and audios, execute stand-alone Python code, use an LLM-based search tool for searching the web using multiple search APIs, the WayBack machine, and Wikipedia, and delegate its browsing tasks to a separate browsing agent. The browsing agent has its own planner agent, uses visual browsing to browse the web, and can execute interactive actions on webpages. It has no native support for writing and editing files, using a bash shell, and executing code in other programming languages. Similar to OpenDeepResearch, it has a restrictive design wherein the agent can only use a fixed set of pre-installed libraries/packages for its Python programs.

SWE-Agent (Yang et al., 2024a) is a software engineering agent that has access to a bash terminal, an agent-computer interface for reading, writing and editing code files, and a specialized Python-specific linter that checks if the edits made by the agent are syntactically correct. It cannot browse webpages, search the web, or read multimodal file content.

SWE-Agent JS and SWE-Agent Multimodal (Yang et al., 2025) are extensions of SWE-Agent for the SWE-Bench M benchmark. SWE-Agent JS adds support for detecting errors in Javascript code edits made by the agent. SWE-Agent Multimodal is built on top of SWE-agent JS, and has the ability to serve local HTML code in a visual web browser, and open images. This allows the agent to visually reproduce image-based issues and visually verify its fixes. Just like SWE-Agent, none of these variants have the ability of browse public webpages, use search engines, or process multimodal file content.

Agentless-Lite (Dunn, 2025) is a lightweight version of Agentless (Xia et al., 2024) that first uses RAG-based localization to retrieve the top 5 files that are relevant to the issue. Next it queries an LLM with these files to generate a patch. While it achieves impressive results with this simple method, its design is very limited. It does not support code execution, bash shell, multimodal file processing, web browsing, or using search engines.

C. Performance on GAIA Validation Split

We also evaluate OpenHands-Versa on the validation split of GAIA. Just like all other experiments, we consider agents with open-source implementation which have reproducibility guidelines and provide details about the exact configuration used by their agent.

Table 4. Example tasks for each of the three benchmarks used in this work.

Benchmark	Task Description (irrelevant details truncated)	Capabilities/Tools required
GAIA	What animals that were mentioned in both Ilias Lagkouvardos's and Olga Tapia's papers on the alvei species of the genus named for Copenhagen outside the bibliographies were also present in the 2021 article cited on the alvei species' Wikipedia page about a multicenter, randomized, double-blind study?	Web search, Web Browsing, Multimodal file processing
	The attached image contains a Python script. Run the Python code against an array of strings, listed below. The output of the Python script will be a URL containing C++ source code. Compile and run this C++ code against the array [35, 12, 8, 99, 21, 5] and return the sum of the third and fifth integers in the sorted list. arr = ['_alg', ..., 'ht']	Code execution, Multimodal file processing, Web Browsing.
The Agent Company	We are collecting employees' preferences on drinks. Please navigate to ownCloud and find drinks_survey.pdf and tell 3 most popular drinks to HR manager via RocketChat.	Web browsing, Multimodal file processing
	On our office cloud at http://the-agent-company.com:8092/, find the July-Sep 2024 financial report for our company, and create a SQLite database with two tables that appropriately populates the data in the report	Web browsing, Code Execution, Multimodal file processing
SWE-Bench M	KML Symbol Align/Placement/Size. There is a bug with the anchor point for some symbols [Right Image] ... I've attached a screen clipping from Google Earth to show how it is supposed to look	Coding, Multimodal file processing (images and code files)
	Bracket highlighted with different color in class inheritance context. - Reproduced in JSFiddle: https://jsfiddle.net/kkangmj/e7h48w36/7/ (Image) ...	Coding, Web Browsing, Multimodal file processing

Using the agent configuration described in §3 and claude-3.7-sonnet as the backbone LLM, OpenHands-Versa achieves a resolve rate of **64.24%** on GAIA validation split. Notably, OpenHands-Versa outperforms top-performing, specialist, multi-agent systems – Magentic-One (46.06% resolve rate), OpenDeepResearch (55.15% resolve rate) and OWL-roleplaying (58.18% resolve rate).

D. Task Examples

In this section, we provide some examples of tasks from each of the three benchmarks – GAIA (Mialon et al., 2023), SWE-Bench M (Yang et al., 2025), and The Agent Company (Xu et al., 2024). Table 4 shows some example tasks along with the tools or capabilities required to solve each of these tasks. Clearly, these examples qualitatively demonstrate that an agent must be proficient in several capabilities to perform well on all three benchmarks. Furthermore, they also help us understand why other agents will not be able to solve tasks from other benchmarks. In the absence of browsing, Agentless-Lite and SWE-Agent cannot solve any of the given examples for GAIA and The Agent Company. In the absence of Javascript code execution, none of the multi-agent systems can solve example tasks given for SWE-Bench M.

E. Inference Costs

In Table 5, we report the total inference cost of the agents on the 3 benchmarks used in this work. Some of the baseline agents do not report costs and are discarded from the table. This cost does not include cost for retries of failed/crashed instances, search API costs and cost of using LLMs during evaluation/metric computation in The Agent Company. For OpenHands and OpenHands-Versa, we report the actual dollar costs with prompt caching for SWE-Bench M and GAIA. However, for The Agent Company, our baselines report costs without prompt caching and simply use token counts to

Table 5. Comparison of total inference cost (in US$) across GAIA, SWE-bench M, and The Agent Company. When available, we report the costs values directly as reported by the baseline agents. Some of the baseline agents in Table 2 do not report costs and are discarded from the table. For The Agent Company, we follow the benchmark authors and calculate costs from the token counts and do not consider prompt caching.

Agent	Model(s)	GAIA	SWE-bench M	The Agent Company
SWE-Agent (Yang et al., 2024a)	gpt-4o	-	1070.19	-
	claude-3.5 sonnet	-	785.84	-
SWE-Agent JS (Yang et al., 2025)	gpt-4o	-	511.83	-
	claude-3.5 sonnet	-	1607.87	-
SWE-Agent Multimodal (Yang et al., 2025)	gpt-4o	-	1519.98	-
	claude-3.5 sonnet	-	1607.87	-
OpenHands v0.14.2 (Wang et al., 2024a)	gpt-4o	-	-	225.75
	gemini-2.0 flash	-	-	101.5
	claude-3.5 sonnet	-	-	1109.5
OpenHands v0.28.1 (Wang et al., 2024a)	claude-3.7 sonnet	210.56	381.50	708.75
OpenHands-Versa	claude-3.7 sonnet	261.52	1010.45	647.50
	claude-sonnet-4	241.83	925.04	285.25

compute costs. To allow direct comparison, we also follow this approach when reporting costs of OpenHands v0.28.1 and OpenHands-Versa on The Agent Company. For all baseline agents, we directly report the costs as reported by the agent authors.

For GAIA, both OpenHands v0.28.1 and OpenHands-Versa have nearly equal costs, with OpenHands-Versa being slightly more expensive. For The Agent Company, OpenHands-Versa with Claude-4-Sonnet is significantly less expensive than most of the baseline agents while offering the strong performance. OpenHands-Versa is slightly cheaper than OpenHands when using claude-3.7-sonnet. For SWE-Bench M, OpenHands-Versa is significantly cheaper than SWE-Agent Multimodal (that has ability to browse local webpages which is not present in other variants of SWE-Agent). On the other hand, OpenHands-Versa is significantly more expensive than OpenHands v0.28.1 – one of the reasons for this is that OpenHands-Versa uses the browser to *visually* verify its changes which OpenHands simply cannot do. While this capability is crucial for front-end software development, it significantly increases the costs due to presence of images and potentially large AXTrees in browsing observations. Finally, Claude-4-Sonnet is more cost efficient than Claude-3.7-Sonnet for all three benchmarks with OpenHands-Versa as the agent scaffold.

F. Limitations of Our Approach

In this section, we describe the limitations of our proposed approach. Firstly, we do not consider tasks that involve interaction with GUI-based desktop computers like those in OSWorld (Xie et al., 2024). OpenHands-Versa and OpenHands both have access to a headless/non-GUI based operating system via the shell. Secondly, OpenHands-Versa has limited video processing abilities and cannot view local video files using the file viewer tool. Potential mitigations for this could be to use an LLM-based file summarizer/video summarizer. Thirdly, like most other agent frameworks, our work also primarily relies on closed-source LLMs. While it is feasible to use open-source LLMs for OpenHands-Versa, we observe that most AI agents perform very poorly when open-source LLMs are used. Finally, due to high cost of using proprietary LLMs, we are unable to evaluate every baseline agent on all the 3 benchmarks, but limit ourselves to strong baseline agents in order to empirically validate our hypothesis.