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]
[1]Carnegie Mellon University, [2]Independent, [3]All Hands AI
{adityabs, gneubig}@cs.cmu.edu

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. Similarly, specialized AI agents with task-specific tools or architectures often fail to generalize beyond their intended scope. In this work, we ask: *can agents achieve generalizability across diverse domains with a small, but well-chosen set of general tools?* We propose OpenHands-Versa, a single-agent system with a modest number of general tools like code execution, search engine, web browser and multimodal file viewer, for three practical domains: software engineering, deep research, and web browsing. Notably, OpenHands-Versa demonstrates superior or competitive performance over task-specific specialized agents on three challenging benchmarks: SWE-Bench Multimodal, GAIA, and The Agent Company, with absolute improvements in success rate of **9.1**, **1.3**, and **9.1** points, respectively. Thus, our *single-agent* system can achieve strong generalization indicating that specialist agents for these domains provide no practical benefit. Furthermore, we find that specialist multi-agent systems do not generalize beyond their intended scope. These findings establish OpenHands-Versa as a strong baseline for future research.[1]

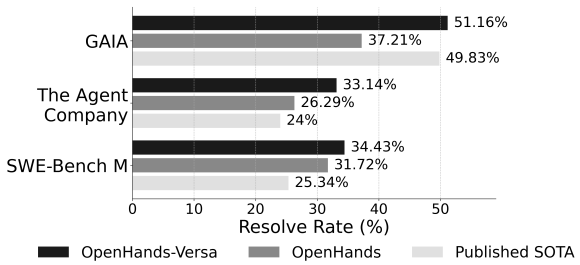

Figure 1: **OpenHands-Versa generalizes across three important domains: SWE, Deep Research and Web Browsing.** OpenHands-Versa achieves superior or competitive performance over SOTA specialist agents and OpenHands for all three benchmarks, implying that our approach achieves generalizability without hurting domain-specific performance. 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. Three prominent domains that have received significant attention from academia and industry are: *software engineering* (OpenAI, 2025c; Anthropic, 2025; Wang et al., 2024a), *deep research* (OpenAI, 2025a; ManusAI, 2025; Fourney et al., 2024), and *web browsing* (Müller and Žunič, 2024; Perplexity AI, 2025; OpenAI, 2025b; Wang et al., 2024b). For instance, agents have demonstrated strong software engineering (SWE) abilities, fixing up to two-thirds of issues in open-source Python repositories from SWE-Bench (Jimenez et al., 2024) and around one-third of issues in front-end libraries from SWE-Bench Multimodal (Yang et al., 2025). Furthermore, agents have demonstrated impressive web navigation capabilities, completing over half of the tasks from WebArena (Zhou et al., 2023) and more than one-third of the tasks from VisualWebArena (Koh et al., 2024). Agents have also achieved strong performance for deep research tasks, solving over half of the tasks from GAIA (Mialon et al., 2023). Finally, agents are also effective as digital co-workers, solving one-fourth of tasks in The Agent Company (Xu et al., 2024) where they must navigate company websites and communicate with colleagues.

However, until this point, the strongest published agents in each of these three domains 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,

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

Findings of the Association for Computational Linguistics: EACL 2026, pages 6052–6069
March 24-29, 2026

SWE-Agent (Yang et al., 2024a) and Agentless (Xia et al., 2024) have achieved state-of-the-art performance on SWE-Bench. Still, they cannot typically browse the web, use search engines, or process multimodal files, making them unsuitable for deep research and web browsing tasks. On the other hand, deep research agents like OpenDeepResearch (Roucher et al., 2025) and OWL (Hu et al., 2025) are not suitable for SWE tasks as they cannot make iterative file edits, execute bash commands, or code in languages other than Python. Similarly, 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 at all, making them unusable for SWE tasks.

To implement an agent that performs well across software engineering, deep research, *and* web browsing, we argue that the majority of tasks in these domains can be tackled by an agent that has these 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), 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 file content.

Therefore, we propose OpenHands-Versa, built using the popular OpenHands framework (Wang et al., 2024a) for coding agents, by imbuing OpenHands with the ability to perform visual browsing, accessing information from the web through search APIs, and processing multimodal file content. This simple strategy is surprisingly effective – we show that our single agent system achieves strong results (Figure 1), rivaling or exceeding the state-of-the-art published systems, on three challenging benchmarks: GAIA for deep research, The Agent Company for workflow automation in a simulated software company, involving web browsing, coding, and file processing, and SWE-Bench Multimodal for frontend-focused software engineering. OpenHands-Versa achieves absolute improvements of **9.1**, **1.3**, and **9.1** points in success rate over best-performing previously published results on GAIA, SWE-Bench Multimodal, and The Agent Company, respectively. Notably, we also find that current state-of-the-art multi-agent systems do not generalize beyond their intended scope.

We analyze the tool-use patterns of OpenHands-Versa and provide insights into why our 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 OpenHands-Versa's trajectories and provide concrete insights for future improvements. Finally, we perform ablation studies to analyse the effect of each tool on downstream performance. To the best of our knowledge, ours is the *first* work to empirically demonstrate that a small yet well-chosen set of general-purpose tools can enable strong generalization across software engineering, deep research, and web browsing domains.

2 Methodology

2.1 Preliminaries

We build OpenHands-Versa on top of the OpenHands (Wang et al., 2024a) framework because of its strong software engineering performance and a built-in harness for evaluating agents on numerous benchmarks, allowing for easier extensibility.[2] Furthermore, OpenHands offers a flexible event stream architecture, a sandboxed runtime, 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 relies on the BrowserGym framework (de Chezelles et al., 2025) for 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.).

Additional discussion is provided in §C.

2.2 Ingredients of Our Agent

Since OpenHands is primarily a software engineering agent, 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 these capabilities while inheriting the coding capabilities from OpenHands.

[2]We use OpenHands v0.28.1.

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

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 and Žunič, 2024)	⊗	📷	⊗	⊗	👤
OpenDeepResearch (Roucher et al., 2025)	</>	▶	🔍	≡	👥
OWL-roleplaying (Hu et al., 2025)	</>	📷	🔍	≡	👥
Magnetic One (Fourney et al., 2024)	</>	📷	🔍	≡	👥
OpenHands-Versa (Ours)	</> ✎	📷	🔍	≡	👤

Multimodal Web Browsing: The browsing tool in OpenHands relies only on text-based observations that represent a webpage using its accessibility tree (AXTree)[4], and misses crucial visual cues from the frontend. Thus, we use the Set-of-Marks prompting method (Yang et al., 2023), which takes 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., see §A) We include the full AXTree of the webpage that gives context beyond the viewport but truncate to the visible viewport if it does not fit in the LLM's context window.

We also incorporate context condensation into the browser tool. OpenHands uses an event stream architecture wherein the LLM is given 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 to improve information access: *web search* and *multimodal file processing*. First, synthesizing information from the web using search engines is a crucial ability. While agents can also access 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 address this by allowing the agent to search the web using a search API. This has the added advantage of reduced costs over browsing the web and allows the use of search APIs specificially designed for agents. We use Tavily API (Tavily-AI, 2024) for our experiments; but also support Exa (Exa, 2023) and Brave (Inc, 2023).

Second, many tasks require the agent to process multimodal data from 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, .js extensions). We enhance the file viewing ability 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 task 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., 2024a), 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 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 progress and create a plan for subsequent task execution.

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

2.3 Comparison with Specialist Agents

Next, we compare OpenHands-Versa with existing agents that excel in specific domains and benchmarks. For SWE, we consider OpenHands, SWE-Agent (Yang et al., 2024a) and Agentless (Xia et al., 2024). For web navigation, we consider BrowserGym GenericAgent (de Chezelles et al., 2025) and Browser-use (Müller and Žunič, 2024). Finally, for deep research, we consider Magentic-One (Fourney et al., 2024), OpenDeepResearch (Roucher et al., 2025), and OWL-roleplaying (Hu et al., 2025). Note that we only compare agents with open-source implementations, since this requires knowledge of their internal design.

The comparison of these agents with OpenHands-Versa is mainly along the 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 SWE tasks, agents must be able 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 lack all these capabilites. 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 for code editing. Also OWL-roleplaying and OpenDeepResearch do not have a shell. These multi-agent systems can only execute stand-alone Python programs and do not support other programming languages, making them unsuitable for SWE tasks. Agents like SWE-Agent, OpenHands, and OpenHands-Versa support all the above code-related capabilities. While Agentless does not have a shell, it supports the execution of selected tests in the repository to validate candidate patches using a human-defined workflow.

Web Browsing: For web navigation, agents should be able to browse webpages, execute interactive actions (like “click", “type"), and have strong multimodal processing abilities 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 have any web browsing abilities. OpenDeepResearch has a very limited browsing ability, wherein it can only open and scroll through webpages, and cannot execute any other actions like “click” or “type”. OpenHands supports interactive browsing actions, but it performs text-only browsing whereas all other deep research agents perform mulimodal web browsing leveraging 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. Except for multi-agent systems for deep research, no other agent supports search APIs.

Information Access via Multimodal File Processing: Agents must be able to process various files like PDFs, presentation slides, and spreadsheets. While this can be done using code or shell, it is prone to bugs and may require multiple attempts to successfully parse the file. Supporting file viewing as a tool is a more robust approach since the agent can access the file content using 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 restricted to plain-text files. All other agents have specific tool(s) for processing multimodal files.

3 Experimental Setup

Next, we describe our experimental setup, outlining the evaluation benchmarks (§3.1), baseline agents (§3.2), the choice of LLMs (§3.3), and evaluation metrics (§3.4).

3.1 Evaluation Benchmarks

We evaluate OpenHands-Versa on 3 benchmarks that cover a diverse range of capabilities—which can be roughly gleaned from Figure 2. §B shows qualitative examples tasks from the benchmarks.

SWE-Bench Multimodal (SWE-Bench M) (Yang et al., 2025) : This benchmark evaluates the software engineering abilities of agents where they must fix GitHub issues in front-end, user-facing libraries. It covers 17 JavaScript repositories for various use-cases like web development, syntax highlighting, and

Table 2: **OpenHands-Versa achieves *generalization* across diverse domains and outperforms SoTA specialist agents on GAIA, SWE-Bench M, and The Agent Company.** Highest metrics are **bold-faced** and 2^{nd} highest metrics are underlined. We consider 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%	-	-	-
Magentic-UI (Mozannar et al., 2025)	o4-mini	42.19%	-	-	-
TapeAgents (Bahdanau et al., 2024a)	GPT-4o	33.20%	-	-	-
	Claude-3.7-Sonnet	47.18%	-	-	-
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-3.5-Sonnet	-	27.66%	-	-
	Claude-3.7-Sonnet	**51.16%**	31.33%	30.86%	40.18%
	Claude-Sonnet-4	**51.16%**	**34.43%**	**33.14%**	**43.19%**

data visualization. Many tasks also have images and videos describing the issue and have URLs to online integrated development environments (IDEs) containing code snippets for reproducing the issue.

GAIA (Mialon et al., 2023): This benchmark evaluates the deep research abilities of agents using tasks that require browsing the open web, performing web search, coding, reasoning, and processing multimodal files. Unlike SWE-Bench M, where coding tasks involve editing *existing* code files from a repository, coding tasks in GAIA are limited to writing stand-alone programs from scratch.

The Agent Company (Xu et al., 2024): This benchmark evaluates the workflow automation abilities of agents as digital workers in a simulated software company. It covers tasks across software development, project management, data science, finance, etc., and uses four self-hosted web environments: GitLab for hosting code repositories, OwnCloud for cloud-based file-sharing, Plane for project management, and RocketChat for communicating with NPCs (simulated colleagues).

3.2 Baseline Agents

For each benchmark, we compare to previously published open-source agent frameworks that demonstrate strong performance on that task.

For **SWE-Bench Multimodal**, we compare with Agentless-Lite (Dunn, 2025), and SWE-Agent (Yang et al., 2024a) along with its Multimodal and Javascript variants proposed by Yang et al. (2025). For **GAIA**, we consider Magentic-One (Fourney et al., 2024), OpenDeepResearch (Roucher et al., 2025), TapeAgents (Bahdanau et al., 2024b), and Magentic-UI (Mozannar et al., 2025). For **The Agent Company**, we consider OWL-roleplaying (Hu et al., 2025) and OpenHands v0.14.2, which is the version used in the original paper. For all benchmarks, we evaluate OpenHands v0.28.1—the agent on top of which OpenHands-Versa is built—to understand the effect of our approach on the agent's generalizability. Note that, most baseline agents report performance on only one of the benchmarks, and their architecture typically does not support

evaluation on the others (§2.3). Refer to §H for additional discussion.

3.3 Choice of LLM Backbones

For all benchmarks, we use Claude-3.7-Sonnet, as the backbone LLM of OpenHands-Versa and OpenHands v0.28.1 to directly compare the two agent architectures using the same LLM. We also evaluate OpenHands-Versa with Claude-3.5-Sonnet on SWE-Bench M to compare with Agentless and SWE-Agent variants using the same LLM. Finally, we also evaluate OpenHands-Versa with Claude-Sonnet-4 on all benchmarks to study the effect of using stronger LLMs. Further experimental details are provided in §D.

3.4 Evaluation metrics

For SWE-Bench M and GAIA, we report resolve rate, i.e. % of tasks completed successfully. Since The Agent Company uses checkpoint-based evaluation, we report: **Full completion score** (% of tasks where all checkpoints were completed) and **Partial completion score** (that provides partial credit for completing checkpoints). We refer the reader to Xu et al. (2024) for more details. We report **pass@1** scores on the *test* split of each benchmark. Also, we report inference costs in §E.

4 Results

We present our experimental results in Table 2 and highlight the key takeaways below.

OpenHands-Versa *generalizes* to diverse domains: Notably, OpenHands-Versa is the *only* agentic system that generalizes to SWE, deep research, and web browsing domains. It achieves state-of-the-art or competitive performance on all three benchmarks with a resolve rate of **51.16%** on GAIA and **34.43%** on SWE-Bench M, and **33.14%** full completion score on The Agent Company. For GAIA, OpenHands-Versa outperforms complex, multi-agent systems that use specially designed agents for distinct skills/sub-tasks, with each agent using a separate LLM. For The Agent Company, OpenHands-Versa has an absolute improvement of **6.8** points in the full completion score and **6.7** points in the partial completion score over the best-performing baseline. For SWE-Bench M, OpenHands-Versa achieves absolute gains in resolve rate of **9.1** points over Agentless-Lite and **22.2** points over SWE-Agent and its variants.

Specialist agents offer no practical advantage over OpenHands-Versa: Specialist systems for these benchmarks have no benefit over our simple, single-agent system, and domain-specific over-engineering often harms generalizability. For instance, while OWL-roleplaying has strong performance on GAIA[5], our evaluation shows that it fails to generalize to The Agent Company with a poor **4%** full completion and **11.0%** partial completion score. On SWE-Bench M (using Claude-3.5-Sonnet), OpenHands-Versa outperforms specialist systems without task-specific optimizations like JavaScript linters in SWE-Agent variants or use of embedding models for RAG-based localization in Agentless-Lite. Finally, OpenHands-Versa significantly outperforms OpenHands when using the same LLM. These findings validate that our approach to improve generalization across diverse domains causes no regression in domain-specific performance. With Claude-3.7-Sonnet, OpenHands-Versa surpasses OpenHands on GAIA with an absolute improvement of **13.9** points in the resolve rate. For The Agent Company, OpenHands-Versa achieves absolute gains of **4.6** points in the full completion score and **3.8** points in the partial completion score over OpenHands. Also there is no regression in the coding abilities of OpenHands-Versa inherited from OpenHands as it achieves a nearly equal resolve rate on SWE-Bench M with an absolute difference of only 0.39 points.

We include additional discussion comparing different agent scaffolds with OpenHands-Versa when using an identical/comparable LLM backbone in Appendix J. We also include additional experimental results with GPT-5 in Appendix F.

5 Analysis

5.1 Tool Use Patterns across Benchmarks

Figure 2 shows the distribution of the relative tool use frequencies (as a % of total tool calls) for the 3 benchmarks, when using Claude-3.7-Sonnet. Here are the key takeaways:

OpenHands-Versa uses appropriate tools that *intuitively* align with task requirements: For GAIA, the agent primarily uses the browser and search tools, consistent with the need to synthesize web-based information, with limited use of file-editing tools while frequently executing standalone Python code via IPython. It makes creative use of the bash tool to install packages and download files with `wget`. In The Agent Company, tool use

[5] OWL-roleplaying reports results only on GAIA validation set (58.18%), 6 points **lower** than OpenHands-Versa (§G)

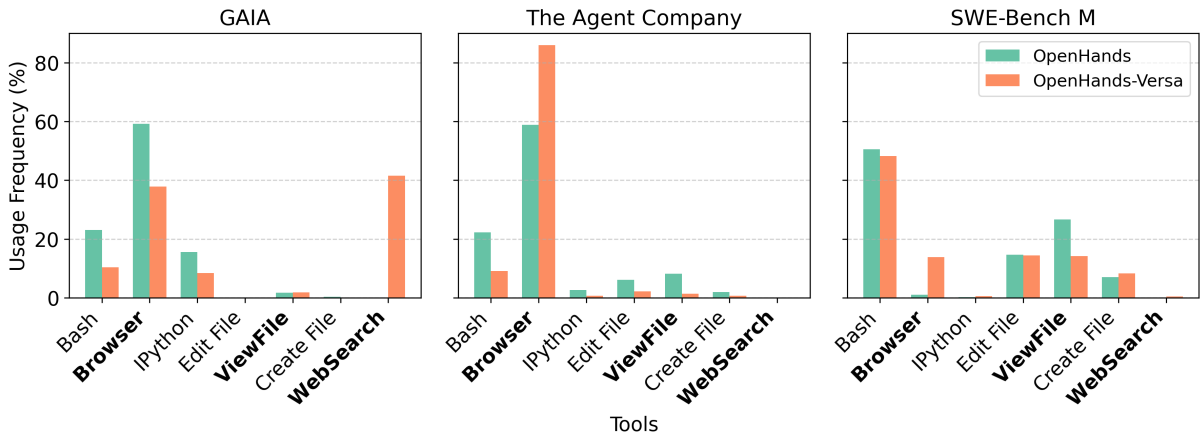

Figure 2: Tool-use distribution of OpenHands and OpenHands-Versa. OpenHands-Versa adapts its tool usage to different domains without any benchmark-specific optimizations and has better domain-aware tool-usage OpenHands. Tools with **bold-faced** names have been modified/created by our work.

Table 3: Effect of individual components on performance and per-instance cost (in US$) for GAIA.

OpenHands-Versa Variant	Resolve Rate (%)	Avg. Cost ($)
w/o search API	46.67	1.41
w/o Multimodal File Viewer	51.52	0.62
w/o Browser enhancements	58.18	**0.29**
Full tools	**64.24**	0.58

is dominated by the browser, as the benchmark's focus is on navigating company websites, while the search engine and IPython tools are rarely used since URLs of company websites are known to the agent and tasks require editing code in company codebases. In SWE-Bench M, frequent use of bash, edit_file, and view_file tools aligns with standard SWE workflow, and the browser is used to visually verify bug fixes by rendering HTML files.

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 tool, due to which it frequently navigates to hallucinated or invalid URLs. OpenHands-Versa uses the search tool to retrieve relevant webpages and then chooses a URL based on the search snippets, resulting in targeted navigation. For SWE-Bench M, 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 lacks due to text-only browsing. For The Agent Company, both agents display similar tool-use behavior, which is expected as we mainly improve browsing observations and not its action space.

5.2 Ablation of Tools

We measure the individual contributions of each new design component (§2.2) to agent performance (Table 3). Since the tools added/modified by our work are used most frequently in GAIA (§5.1), we perform this analysis using GAIA validation set. We remove each tool/feature individually – search API, multimodal file viewer, and browsing enhancements (i.e., without screenshots and context condensation). Table 3 reports performance and average costs of agent variants using Claude-Sonnet-4. Interestingly, removing each new component degrades performance, with the largest drop of **17** points when removing the search tool. Furthermore, OpenHands-Versa achieves the 2^{nd} lowest cost among all variants, higher only than the variant without browsing enhancements as processing images is much costlier than text in LLM APIs.

5.3 Error Analysis

To identify directions for future improvement, we manually analyze OpenHands-Versa 's trajectories across the 3 benchmarks (with claude-3.7 sonnet as the LLM). For GAIA and SWE-Bench M, we analyse all the unresolved instances from the validation set. For The Agent Company, we focus on a subset of 60 tasks that OpenHands-Versa could not solve or underperformed as compared to OpenHands. Table 4 shows the relative frequency of the key error behaviors and Table 7 shows qualitative examples of the observed behaviours.

For GAIA, the most common errors include flawed reasoning/task understanding and failure in searching the correct factual information due to

Table 4: Relative frequency (in %) of various error behaviours of OpenHands-Versa for each benchmark.

Error Behaviour	Frequency
GAIA	
Incorrect task approach (flawed understanding, wrong assumptions, reasoning errors)	50.0%
Cannot find correct info (misleading search results, browsing errors, CAPTCHAs etc.)	26.8%
Output formatting issues (wrong units, rounding errors, etc.)	9.8%
Fails to correctly process videos	8.5%
Retrieves info outside task-specific timeframe (uses latest data instead)	4.9%
SWE-Bench Multimodal	
Ignores or fails to run existing regression tests in the repository	35.7%
Other failures (step limit exceeded, wrong files localized, logical flaws in code)	27.1%
Doesn't write tests to reproduce bug, misses edge cases, or fails to run these tests	18.6%
Does not open the URL to online IDE containing code to reproduce the issue	18.6%
The Agent Company	
Stuck in loop when browsing due to pop-ups, failure to locate/download files, etc.	58.5%
Flawed task understanding causes premature task termination and/or partial completion	24.6%
Writes buggy code when solving SWE tasks	9.2%
Social communication errors (e.g. ignores NPC's instructions, messages wrong NPC)	7.7%

browsing errors, CAPTCHAs and over-reliance on the potentially misleading snippets from the webpage given by the search API. For SWE-Bench M, other than expected errors for SWE tasks like incorrect file localization and flawed code fixes, the agent often does not comprehensively test its code, using regression and reproduction tests, and prematurely terminates assuming that it has fixed the bug as its non-exhaustive tests pass. For The Agent Company, the dominant errors happen during browsing (especially when using OwnCloud) since the frequently gets stuck in loops. The agent often prematurely exits without completing all the checkpoints for more complex tasks, struggles with social communication (with NPCs via RocketChat), and struggles with more complex coding tasks.

5.4 Effect of Search APIs

As the search tool strongly affects performance (§5.2) and accounts for 40% of tool calls on GAIA (§5.1), we study how different search APIs impact results. We evaluate OpenHands-Versa on the GAIA validation split using Brave, Exa, and Tavily APIs with Claude-3.7-Sonnet. The choice of seach API notably influences performance, yielding resolve rates of **56.96%**, **58.18%**, and **64.24%**, respectively, with a **7.28** point gain when switching from Brave to Tavily. Interestingly, OpenHands-Versa often relies on search snippets to synthesize information from the web. Brave extracts these snippets from webpage text, while Exa and Tavily use higher-quality LLM-generated summaries, often removing the need to open webpages in the browser. However, these snippets also occasionally introduce hallucinations. Tavily partially mitigates these by offering an LLM-generated answer per query, synthesized from all the retrieved results, which tends to be more accurate than individual page summaries. This is also a potential reason of higher resolve rate when using Tavily.

6 Conclusion

In this work, we propose OpenHands-Versa– the *first* single-agent system that *generalizes* to three practical domains: SWE, deep research, and web browsing. Notably, this is validated by its strong performance on three challenging benchmarks: GAIA, SWE-Bench M, and The Agent Company. Our experimental results highlight that generalizability can be achieved using an intuitive agent design without over-engineered domain-specific optimizations, and specialist agents for these domains offer no practical advantage over our agent. Thus, even if none of the individual components in OpenHands-Versa is revolutionary in itself, designing an agent that generalizes across diverse domains without harming domain-specific performance is a *challenging* problem, and highlights our novel research contributions. Furthermore, our analysis provides intuitive understanding about the tool-use behaviours and the impact of different tools on performance. Our analysis of the diverse error behaviours provides concrete directions for future improvement. In conclusion, OpenHands-Versa serves as a strong baseline for future research on improving generalizability of LLM agents.

Limitations

We describe the limitations of our proposed approach. Firstly, while SWE, deep research, and web browsing cover a vast majority of use-cases of LLM agents, there also exist other tasks which we do not consider like database management, perception tasks like robotics and playing video games, legal domain, video processing, interaction with GUI-based desktop computers like those in OS-World (Xie et al., 2024) etc. Note that this limitation is even more pronounced for the baseline agents which cannot solve any task beyond a single domain, and our work makes significant progress in this direction of making LLM agents more generalizable.

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. Also, 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. High inference costs also prevent us from reporting averages (and standard deviations) over multiple runs and reliably computing statistical significance of results. However, our main claim is not just about SoTA performance, but about an approach for designing agents that generalize to diverse domains.

In addition, our ablation studies do not formally prove the minimality of our toolset, but they do offer concrete empirical evidence (Table 3) to show that removing even a single tool from our agent's toolset significantly degrades performance on GAIA by 17.6% without search tool, by 12.7% without multimodal file viewer, and by 6.1% without browser tool enhancements. These findings offer an intuitive justification that an even smaller toolset obtained by removing more tools will harm (or at least not improve) agent's performance. We also do not conduct ablation analyses for other benchmarks as our agent most frequently uses tools added/modified by our work for GAIA compared to other benchmarks (Figure 2) and also because of their high experimental costs (Table 6). On the other hand, while adding more tools may further improve performance of OpenHands-Versa, we focus on adding tools necessary to develop an agent with the three core capabilities defined in Section 1 - coding, multimodal browsing, and information access. We do not explore adding more tools as our experimental results are sufficient to answer our core research problem, and even with a simple design, our agent generalizes across three diverse domains.

Furthermore, our task planning mechanism is too simple, its limitations and scalability for more complex, long-horizon planning scenarios are not extensively studied.

Finally, while we hypothesize that adding these capabilities to other specialized agents would improve their generalizability, we do not have empirical evidence for this since modifying other agent systems requires significant implementation efforts and is beyond the scope of current work. Most specialist agents are engineered such that it is difficult to add additional tools as described below. Firstly, the existing tools in most specialist agents are overly constrained and need more improvement - e.g., limited coding abilities (for eg. OWL-roleplaying can code only in Python and use only a fixed set of pre-installed packages), limited action space in the browser (OpenDeepResearch supports only "goto" and "click" actions), etc. Secondly, they also lack essential features like sandboxed execution environments (e.g. via Docker containers) which is especially crucial for SWE tasks and for preventing any potential harms on the user's machine. Most importantly, specialist agents depend heavily on benchmark-specific prompt engineering and restrictive pre-defined workflows which we must avoid to ensure broad applicability to diverse domains.

References

Anthropic. 2025. Claude code. https://www.claude.com/product/claude-code. Accessed: 2025-10-01.

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

Dzmitry Bahdanau, Nicolas Gontier, Gabriel Huang, Ehsan Kamalloo, Rafael Pardinas, Alex Piché, Torsten Scholak, Oleh Shliazhko, Jordan Prince

Tremblay, Karam Ghanem, and 1 others. 2024a. Tapeagents: a holistic framework for agent development and optimization. *arXiv preprint arXiv:2412.08445*.

Dzmitry Bahdanau, Nicolas Gontier, Gabriel Huang, Ehsan Kamalloo, Rafael Pardinas, Alex Piché, Torsten Scholak, Oleh Shliazhko, Jordan Prince Tremblay, Karam Ghanem, Soham Parikh, Mitul Tiwari, and Quaizar Vohra. 2024b. Tapeagents: a holistic framework for agent development and optimization. *Preprint*, arXiv:2412.08445.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. 2025. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*. Expert Certification.

Dunn(2025). Agentless-lite.

Exa. 2023. Exa search api.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, and 1 others. 2024. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Ping Luo, and Guohao Li. 2025. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation.

Brave Software Inc. 2023. Brave search api.

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

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*.

ManusAI. 2025. Manusai. `https://manus.im/`. Accessed: 2025-10-01.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Hussein Mozannar, Gagan Bansal, Cheng Tan, Adam Fourney, Victor Dibia, Jingya Chen, Jack Gerrits, Tyler Payne, Matheus Kunzler Maldaner, Madeleine Grunde-McLaughlin, and 1 others. 2025. Magentic-ui: Towards human-in-the-loop agentic systems. *arXiv preprint arXiv:2507.22358*.

Magnus Müller and Gregor Žunič. 2024. Browser use: Enable ai to control your browser.

OpenAI. 2025a. Introducing deep research. `https://openai.com/index/introducing-deep-research/`. Accessed: 2025-10-01.

OpenAI. 2025b. Introducing operator. `https://openai.com/index/introducing-operator/`. Accessed: 2025-10-01.

OpenAI. 2025c. Openai codex. `https://openai.com/codex/`. Accessed: 2025-10-01.

Perplexity AI. 2025. Perplexity comet. `https://www.perplexity.ai/comet`. Accessed: 2025-10-01.

Aymeric Roucher, Albert Villanova del Moral, Merve Noyan, Thomas Wolf, and Clémentine Fourrier. 2025. Open-source deepresearch – freeing our search agents.

Tavily-AI. 2024. Tavily search api.

Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. 2003. 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*, pages 294–311. Springer.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, and 1 others. 2024a. Openhands: An open platform for ai software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024b. Agent workflow memory. *arXiv preprint arXiv:2409.07429*.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2024. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Preprint*, arXiv:2404.07972.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, and 2 others. 2024. Theagentcompany: Benchmarking llm agents on consequential real world tasks. *Preprint*, arXiv:2412.14161.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *Preprint*, arXiv:2310.11441.

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

John Yang, Carlos E Jimenez, Alex L Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R Narasimhan, Diyi Yang, Sida Wang, and Ofir Press. 2025. SWE-bench multimodal: Do AI systems generalize to visual software domains? In *The Thirteenth International Conference on Learning Representations*.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2024b. Agentoccam: A simple yet strong baseline for llm-based web agents. *arXiv preprint arXiv:2410.13825*.

Daoguang Zan, Zhirong Huang, Wei Liu, Hanwu Chen, Linhao Zhang, Shulin Xin, Lu Chen, Qi Liu, Xiaojian Zhong, Aoyan Li, Siyao Liu, Yongsheng Xiao, Liangqiang Chen, Yuyu Zhang, Jing Su, Tianyu Liu, Rui Long, Kai Shen, and Liang Xiang. 2025. Multi-swe-bench: A multilingual benchmark for issue resolving. *Preprint*, arXiv:2504.02605.

Ruichen Zhang, Mufan Qiu, Zhen Tan, Mohan Zhang, Vincent Lu, Jie Peng, Kaidi Xu, Leandro Z Agudelo, Peter Qian, and Tianlong Chen. 2025. Symbiotic cooperation for web agents: Harnessing complementary strengths of large and small llms. *arXiv preprint arXiv:2502.07942*.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. 2024. Autocoderover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1592–1604.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Webpage Screenshot with Set-of-Marks Annotation

Figure 3 shows an example screenshot of a webpage with set-of-marks annotation. Note that all the interactable elements on the webpage (for e.g., buttons, dropdowns, hyperlinks, text boxes, etc.) are annotated with a bounding box and a unique identifier (i.e. browsergym-id) which is used by the action space of the browser tool.

B Example Tasks from Evaluation Benchmarks

Table 5 demonstrates some example tasks for the 3 evaluation benchmarks used in this work: GAIA (Mialon et al., 2023), The Agent Company (Xu et al., 2024), and SWE-Bench Multimodal (Yang et al., 2025). Clearly, the tasks in these benchmarks are heavily complex in nature and the agent must have numerous capabilities to solve each task. Furthermore, Mialon et al. (2023) performs manual annotation of capabilities required to solve tasks in GAIA and reports that **29%** tasks need multimodal reasoning and **27%** tasks require diverse file reading abilities. Similarly, for SWE-Bench Multimodal, every task involves at least one visual asset (images, videos, or GIFs) in the issue description or unit tests. Finally, The Agent Company evaluates the agent's ability to solve tasks encountered by employees across diverse domains like finance, human resources, project management, software development, admin, data science, etc. and require several capabilities like web browsing, coding, multimodal file processing, and social communication (with colleagues).

C Rationale behind Choice of OpenHands

In this section, we describe our rationale behind the choice of OpenHands (Wang et al., 2024a) as the starting point for designing OpenHands-Versa. While we can ideally develop on top of existing deep research agents or browsing agents, we choose software engineering agents as a starting point because of significant implementation efforts required to develop a coding agent — sandboxed execution (for e.g. using Docker) for all tool calls, a bash terminal that supports execution of Unix-style commands, IPython server for Jupyter notebook-style execution, file processing abilities (for editing, viewing, and creating files), etc.

Table 5: Example tasks for each of the three benchmarks used in this work.

Benchmark	Task Description (irrelevant details truncated)	Capabilities/Tools required
GAIA	In the YouTube 360 VR video from March 2018 narrated by the voice actor of Lord of the Rings' Gollum, what number was mentioned by the narrator directly after dinosaurs were first shown in the video?	Web Search, Web browsing, Multimodal processing
	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	Create a logo picture for the Sotopia code repository on GitLab. The logo should feature a big "S" letter in the middle. Then, send the picture to Jessica Lee on Rocketchat. After getting her approval of the picture, you should change it as the profile picture of the Sotopia code repository.	Web browsing, Multimodal file processing (Image Generation)
	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
	The rightmost data point gets cut off for line chart ## Expected Behavior Line chart will be displayed in full. ## Current Behavior The data point to the far right gets cropped off. Image URL: https://user-images.githubusercontent.com/... ## Steps to Reproduce (for bugs) This is a working example of this bug: https://codepen.io/LeoU/pen/gVLybO ...	Coding, Web browsing, Multimodal file processing

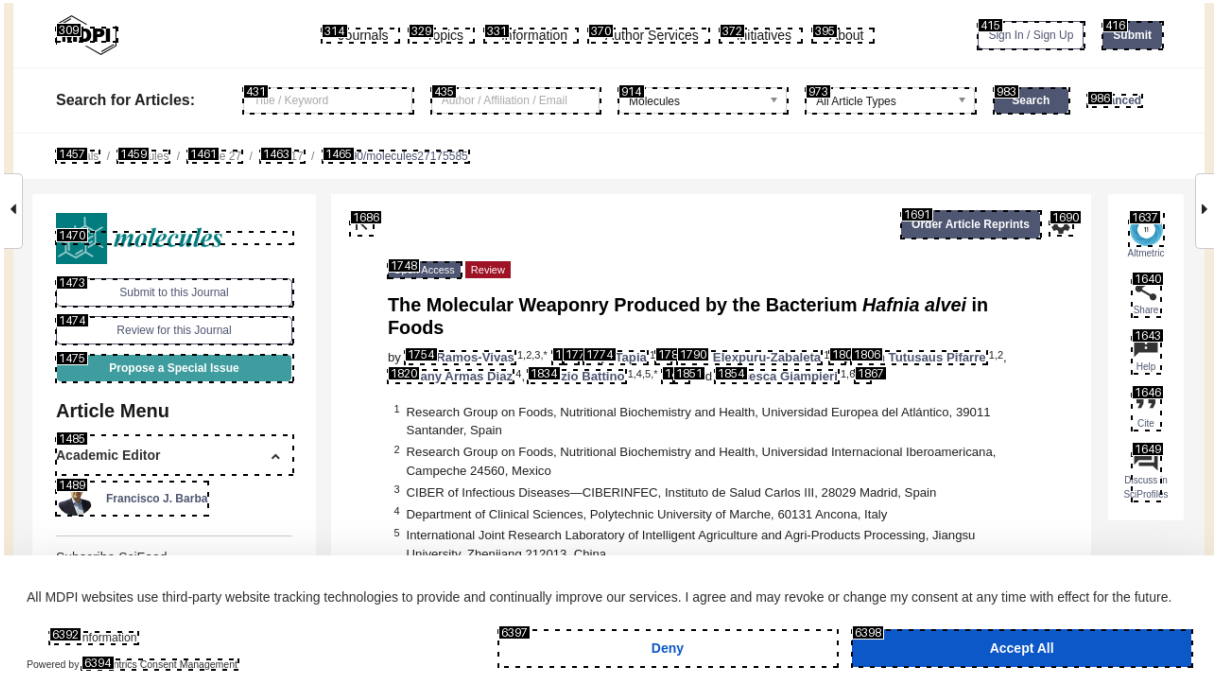

Figure 3: Example screenshot of a webpage with set-of-marks annotation

Among the software engineering agents, we considered OpenHands (Wang et al., 2024a), Agentless (Xia et al., 2024), SWE-Agent (Yang et al., 2024a), and AutoCodeRover (Zhang et al., 2024). With Claude-3.5 Sonnet, OpenHands has a higher resolve rate of 53% as compared to SWE-Agent (33.6%), AutoCodeRover (46.2%) and Agentless-1.5 (50.8%) on SWE-Bench Verified. Furthermore, among all the coding agents, only OpenHands supports a text-based browser tool which further simplifies implementation. In addition, OpenHands has a comprehensive evaluation suite which supports **23** evaluation benchmarks. Unlike complex multi-agent systems like Magentic-One (Fourney et al., 2024) or pipeline-based workflows like Agentless (Xia et al., 2024) that are over-engineered for a particular domain, OpenHands has a relatively simple and intuitive single-agent design where the LLM has access to relevant tools. This allows us to systematically study our main research question about how we can design agents that *generalize* across diverse task domains.

D 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

[6] https://archive.org

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 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 The Agent Company takes ≈50 hours.

E Inference Costs

In Table 6, 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 (relatively small) 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 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). Notably, when using claude-3.5 sonnet, cost of using OpenHands-Versa is less than half the cost of using SWE-Agent Multimodal, with **2.5**x higher success rate. On the other hand, OpenHands-Versa is significantly more expensive than OpenHands v0.28.1 for SWE-Bench M – 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 Additional Experimental Results with GPT-5

After the EACL submission deadline, we also evaluate OpenHands-Versa with the recently released frontier LLM GPT-5. Note that unlike different variants of Claude-Sonnet, GPT-5 is a reasoning model that generates *long* reasoning traces at each step which are *omitted* from the prompt for subsequent steps, following the recommended method for using reasoning LLMs in agents. This is different from older models like Claude-Sonnet and GPT-4o that would generate shorter thoughts at each step which were included in the prompt for subsequent steps.

As shown in Table 8, GPT-5 significantly outperforms all Claude models on GAIA test set with **67.11%** resolve rate and an absolute improvement of **15.95%** over Claude-Sonnet-4. On the other hand, it achieves nearly equal or comparable performance as that of Claude models on SWE-Bench Multimodal, outperforming Claude-3.7-Sonnet 2.52% absolute improvement and slightly under-performing over Claude-Sonnet-4 by 0.58%.

Interestingly, GPT-5 significantly underperforms both Claude-Sonnet-4 and Claude-3.7-Sonnet on The Agent Company with an absolute decrease of **6.85%** in full completion score and **4.63%** in partial completion score over Claude-Sonnet-4. We suspect that this maybe due to a combination of our context condensation strategy in the browser tool (where we retain observations only from the latest browsing action) and the long-horizon nature of tasks in The Agent Company - many tasks often include multiple sub-tasks that require the agent to gather necessary information to complete the task correctly (for e.g. asking for the location of a file from an NPC, and then doing some other task using this file).

While previous LLMs often logged the infor-

Table 6: Comparison of average inference cost per task instance (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	-	2.07	-
	claude-3.5 sonnet	-	1.52	-
SWE-Agent JS (Yang et al., 2025)	gpt-4o	-	0.99	-
	claude-3.5 sonnet	-	3.11	-
SWE-Agent Multimodal (Yang et al., 2025)	gpt-4o	-	2.94	-
	claude-3.5 sonnet	-	3.11	-
OpenHands v0.14.2 (Wang et al., 2024a)	gpt-4o	-	-	1.29
	gemini-2.0 flash	-	-	0.58
	claude-3.5 sonnet	-	-	6.34
OpenHands v0.28.1 (Wang et al., 2024a)	claude-3.7 sonnet	0.70	0.76	4.05
OpenHands-Versa	claude-3.5 sonnet	-	1.49	-
	claude-3.7 sonnet	0.87	1.95	3.70
	claude-sonnet-4	0.80	1.79	1.63

Table 7: Qualtiative examples demonstrating 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.

Table 8: Comparison of OpenHands-Versa across different choices of LLM backbones on the three benchmarks

Model(s)	GAIA	SWE-bench M	The Agent Company	
			Full	Partial
Claude-3.5-Sonnet	-	27.66%	-	-
Claude-3.7-Sonnet	51.16%	31.33%	30.86%	40.18%
Claude-Sonnet-4	51.16%	34.43%	33.14%	43.19%
GPT-5	67.11%	33.85%	26.29%	38.56%

mation/findings from the intermediate sub-tasks in their shorter thoughts at each step before calling necessary tools, which were available in the prompt for the subsequent steps, this is not the case for GPT-5 which would often log the intermediate findings in its reasoning traces which would not be included in the agent's prompt. As a result, the agent would often repeat the same sub-task repeatedly and exhaust its maximum step count before completing the task. We leave it to future work to explore better strategies to mitigate this issue, like including summarized reasoning traces in the prompt.

G Performance on GAIA validation set

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.

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).

Potential leakage of answers in GAIA validation set: Unlike the test split, where the ground truth answers are hidden, the answers for the tasks in the validation split are available online (via HuggingFace). Thus, the agent can potentially *cheat* and lookup answers from the web. However, our manual inspection found only one instance where this may have potentially occurred. In nearly all tasks, the agent behaves reliably and does not benefit from leaked information.

H Why Baseline Agents Cannot be Used for Other Benchmarks?

Agentless-lite and all SWE-agent variants cannot browse the open web, use search engines, or process multimodal file content, which are crucial for GAIA and The Agent Company. Baseline agents for GAIA cannot typically write code in languages other than Python (like JavaScript), making them unsuitable for SWE-Bench M. Unlike SWE-Bench, where all issues require only editing Python files, **28%** of SWE-Bench M instances require the agent to edit files across 2 or more programming languages (Yang et al., 2025).

I Description of 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 Open-DeepResearch. 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.

J Comparison of Agent Performance across backbone LLMs

In Table 2, we compare performance of multiple agent scaffolds across different choices of LLMs. While it is prohibitively expensive to evaluate all the baseline agents with the same LLM backbone as that of OpenHands-Versa, each of the three benchmarks includes one or more baselines which use the same or a comparable LLM as explained below.

SWE-Bench Multimodal: Since all the baseline agents except OpenHands achieve the best performance with Claude-3.5-Sonnet, we evaluate OpenHands-Versa with the same LLM on this benchmark. Notably, it outperforms all SWE-Agent variants with **15.47%** absolute improvement and achieves slightly better performance than Agentless-Lite with **2.32%** absolute gain. Also, OpenHands-Versa and OpenHands achieve nearly equal resolve rate on this benchmark with Claude-3.7-Sonnet, further highlighting that OpenHands-Versa retains the coding abilities inherited from OpenHands.

The Agent Company: Since OpenHands is the best performing agent baseline on this benchmark, we compare OpenHands-Versa and OpenHands with Claude-3.7-Sonnet. Notably, OpenHands-Versa outperforms OpenHands with an absolute improvement of **4.57%** in full completion rate and **3.77%** in partial completion rate.

GAIA: For GAIA, OpenHands-Versa outperforms TapeAgents by **3.98%** when using the same LLM (Claude-3.7-Sonnet). Furthermore, Open-

DeepResearch use a comparable LLM backbone (o1) and Magentic-One also uses GPT-4o and o1 for this benchmark. Notably, with Claude-3.7-Sonnet, OpenHands-Versa achieves an absolute gain of **19.29%** over Magentic-One and 1.33% over OpenDeepResearch.

The above findings highlight that even when using an identical/comparable LLM backbone, OpenHands-Versa achieves superior or competitive performance over specialist LLM agents using an identical **unified** scaffold.

K License of Artifacts

We use publicly available open-source datasets and agent implementations in our work which are developed by prior research and released under permissible open-source licenses (e.g. MIT, Apache 2.0, etc.). Our use of these artifacts is for research purposes and is consistent with the license agreements.

L Data statistics

SWE-Bench M contains 517 instances in the test split and 102 instances in the dev split. GAIA contains 165 instances in the dev split and 301 instances in the test split. The Agent Company contains 175 instances in the test split.

M Use of LLM-based Assistants

We use LLMs only to rephrase/refine the content of the paper and to debug some of the scripts used to make plots.

N Potential Risks

AI agents have shown promise in addressing complex tasks, but still face significant limitations when given long-horizon, real-world tasks. 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 agent capabilities, particularly in the domain of generalizable agents. However, these advancements bring important societal considerations. As AI agents become more sophisticated, potential risks emerge, including:

- Misuse of agents for illegal or harmful activities like sending phishing emails, posting harmful content on social media, spreading misinformation etc.
- Potential for causing large-scale unemployment and labor market disruption as agents become capable of performing tasks currently done by humans.
- Challenges for governance and policy creation as new risks and capabilities emerge with further development in capabilities of AI Agents.

Future work should focus not only on enhancing agent capabilities but also on developing appropriate safeguards, ethical frameworks, and policies to guide implementation in real-world contexts.