Sandboxed Coding Agents are Competitive Omni-modal Task Solvers

Dongping Chen[1*], Xuanao Huang[*], Zhihan Hu, Qingyuan Shi, Dianqi Li, Tianyi Zhou[2†]
[1]University of Maryland, [2]MBZUAI

Abstract

As multimodal LLMs increasingly target video and audio, it is often assumed that such tasks require native omnimodal models. We show that this is not always the case: coding agents with only text+image access and a sandboxed tool-use interface can match, and in several settings outperform, SOTA native omnimodal models and predefined multimodal agent scaffolds across multiple audio-video benchmarks. Our trajectory analysis suggests that their strength comes from writing code and orchestrating tools to extract relevant evidence from transcripts, frames, and other modality signals, thereby converting omnimodal tasks into retrieval and information-processing problems rather than ingesting entire media streams. We further characterize their limitations through a failure taxonomy and process-level trace analysis, and show that simple skill injection, including human-written and self-distilled skills, substantially improves performance. To explore open-source elicitation, we introduce CODE-O, a training recipe with the OMNICODING trajectory dataset and verifiable reward, and provide baselines on Qwen-3.5-9B and Qwen-3.6-27B. Finally, we argue that the next frontier is many-modality processing, and introduce TERMINALBENCH-O, a process-level benchmark for real-world omnimodal processing tasks. Code will be available at `https://github.com/Dongping-Chen/OmniCoding`.

1 Introduction

As AI agents move from screen-bound workflows toward the broader world around users, they are expected to spend increasing amounts of time participating in everyday life as general-purpose assistants. To do so, they must handle multi-modal information from the embodied environments, especially video and audio. This immediately raises a question: Is native video and audio understanding necessary to an agent that is expected to process video or audio in real-world tasks?

In practice, however, many commonly-adopted agents nowadays, such as OpenClaw [57] and Hermes [54], are still text-only or text+image systems, while extending them to handle video and audio natively requires substantial additional modeling and engineering effort. This creates a large gap between real-world agent deployment and the dominant paradigm for video-audio processing, namely native omni-modal models that directly ingest audio and video [62, 72]. At the same time, modern coding agents already make highly effective use of terminal environments for search, editing, execution, and long-horizon task solving [89, 1, 7]. This raises a straightforward question:

Can coding agents, which are not designed for native multimodal perception, solve video and audio tasks by exploiting existing tools like `ffmpeg`, ASR, or new tools they write in a sandbox?

We investigate this problem using the most native Codex and Claude Code harnesses, without any human-provided tools or skills. Text+image coding agents, namely GPT-5.4 `high`/`xhigh`

[*]Equal contribution. [†]Corresponding author: `david.tianyi.zhou@gmail.com`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

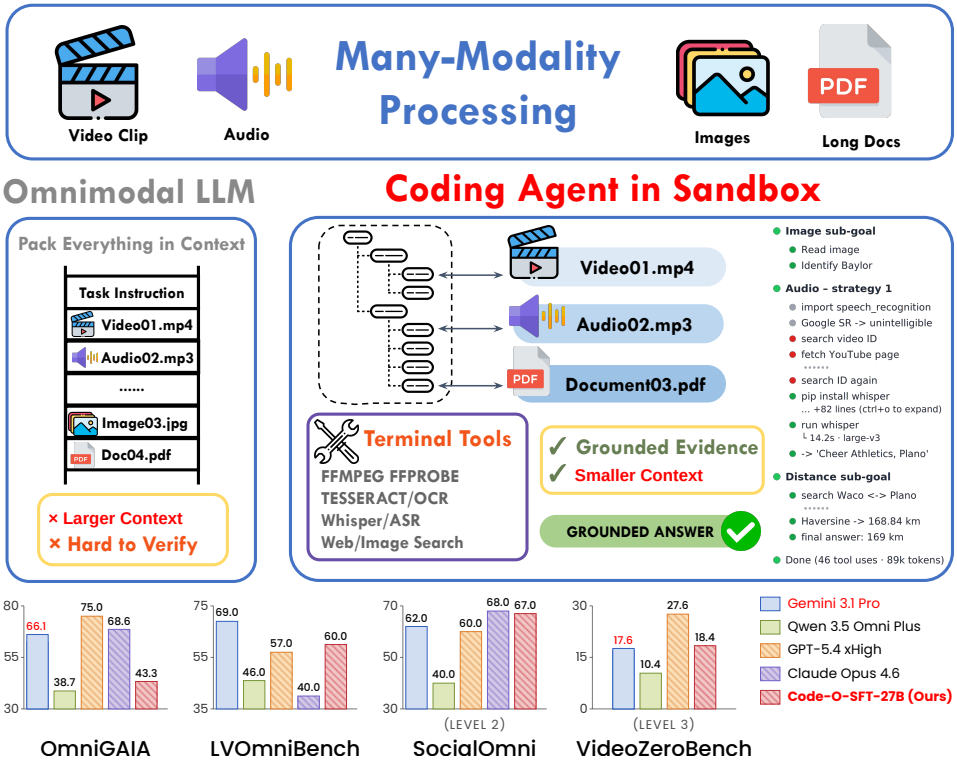

Figure 1: We discover that sandboxed coding agents are competitive omnimodal processors, achieving comparable performance and even surpassing native omnimodal models with shorter context.

and Claude Opus 4.6 [56, 2, 1], achieve strong performance on video-audio benchmarks such as OmniGAIA, SocialOmni, and LVOmniBench [40, 84, 71]. Codex with GPT-5.4 `xhigh` reaches 75.0% on OmniGAIA without directly perceiving raw video or audio, outperforming the strongest native omni-modal model in our study, Gemini 3.1 Pro [72], by 8.9 percentage points, while Claude Opus 4.6 exceeds it by 2.5 percentage points. Trajectory analysis shows that this advantage does not come from end-to-end perception. Instead, coding agents selectively decompose video and audio into transcripts, sampled frames, and metadata through staged tool use, reducing the context length relative to packing raw streams into model context.

To push coding agents on omni content further, we ask where they still fail and whether such failures can be repaired without retraining the base model. We sample 200 trajectories across models and benchmarks and propose an **Omni Terminal Failure Taxonomy** together with a process-level evaluation, which reveals six recurring failure modes ranging from premature exploration of multimodal content to brittle tool and environment infrastructure [28, 93, 43]. Guided by this taxonomy, we then study three skill-injection settings inspired by recent work on reusable agent skills and harness optimization [39, 34]: human-in-the-loop skills that encode expert workflows, calibration-set self-iteration where the agent revises its skills using only binary correctness feedback, and log-driven self-distillation where a separate agent mines reusable skills from raw execution traces. All three improve over the no-skill baseline, raising average OmniGAIA accuracy from 61.4% to 76.7% in the best setting, with execution-trace distillation outperforming both binary self-iteration and human-written skills.

The above results are obtained with closed-source frontier models, which raises a natural follow-up question: can the same capability be elicited on open-source models through targeted post-training? As an exploratory baseline, we propose CODE-O, a complete recipe for many-modality terminal agents. We construct OMNICODING, a 6,035-example trajectory dataset spanning video, audio, image, and cross-modal tasks, and train Qwen-3.5-9B and Qwen-3.6-27B with a supervised warm-start followed by GSPO-based reinforcement learning under a process-aware verifiable reward that jointly penalizes modality-skipping, malformed tool calls, and answer mismatches. Evaluating on four challenging benchmarks, CODE-O yields substantial gains over direct MLLM baselines and achieves state-of-the-art results among open-source models, with the 27B model reaching 43.3% on

OmniGAIA, 60.0% on LVOmniBench, where it surpasses all existing coding agents, and more than doubling performance on VideoZeroBench.

Given the maturity of many-modality understanding, we believe the meaningful frontier lies in many-modality processing: editing videos, auditing image databases, producing spec-compliant multimodal outputs, where coding agents enjoy a structural advantage over end-to-end omni models. To draw attention to this second half of many-modality, we introduce **TERMINALBENCH-O**, a process-level benchmark for coding agents on omni-modal processing, with 50 real-world tasks across video, audio, image, document, and cross-modal categories, each evaluated along weighted dimensions over intermediate artifacts and final output compliance. Despite strong omni-understanding results, these tasks remain hard for current agents as they demand long-horizon planning, reliable tool chaining, cross-modal grounding, artifact verification, and strict output adherence; even the strongest GPT-5.5 Codex reaches only 24% on TERMINALBENCH-O. We hope our analysis, training recipe, and benchmark open new directions for many-modality processing.

2 Analysis of Sandboxed Agent Performance on Omni-Modal Tasks

Preliminary: Adapting Sandboxed Agents for Omni-Modal Tasks. We represent each omni-modal task as $x = (q, \mathcal{F}, \mathcal{Y})$, where q is a natural-language instruction, $\mathcal{F}$ is a set of input files, and $\mathcal{Y}$ is the set of acceptable final answers (formal definitions and the full rollout formulation are deferred to Appendix C). A native omni-modal model and a sandboxed agent differ mainly in *where* omni-modal processing happens:

$$\textbf{native omni:} \quad (q, \mathcal{F}) \xrightarrow{\text{pack}} \text{model context} \xrightarrow{M_\theta} \hat{y},$$

$$\textbf{sandboxed agent:} \quad (q, \mathcal{F}) \xrightarrow{\text{stage}} \text{workspace} \xrightarrow{\text{terminal interactions}} \tau \xrightarrow{\text{Ans}} \hat{y}.$$

A native omni-modal model M_θ packs all files into model context via media-specific encoding and sampling, treating video and audio as first-class model inputs to generate answers $\hat{y}$. A sandboxed agent, by contrast, stages $\mathcal{F}$ into an isolated workspace and is given only the instruction and file paths; it then inspects files, invokes tools such as `ffmpeg`, `ffprobe`, OCR, and ASR, writes and executes scripts, produces intermediate artifacts (transcripts, sampled frames, OCR text, timestamps, tables, crops), which compose a trajectory τ, and submits a final answer $\hat{y}$ parsed from τ. This view treats raw video, audio, images, and documents as environment state rather than mandatory model context, so the agent can solve omni-modal tasks by adaptively reducing raw media into compact evidence that fits the native reasoning interface of a general-purpose coding model.

Agent Harness and Models. We use Claude Code [1] and Codex [6, 7] as our two primary agent harnesses. Under CC, we evaluate Claude Opus 4.6 and Claude Sonnet 4.6 [2, 3], as well as MiniMax M2.7 [53] and Kimi K2.5 [29]. Claude Opus 4.6 and Claude Sonnet 4.6 both use the default `max` setting. Under Codex, we evaluate GPT-5.4 with `low`, `medium`, `high`, and `xhigh` reasoning effort settings [56]. Throughout evaluation, we follow the official harnesses as closely as possible in order to approximate realistic end-user usage.

For native omni models, we evaluate Gemini 3.1 Pro [72], Gemini 3 Flash [15], Qwen 3.5-Omni [62], and MiMo-V2-Omni [83]. We further include two modular omni-agent baselines, OmniAgent [69] and Agent-Omni [45], as strong system-level comparators. Unless otherwise specified, both agent baselines are instantiated with GPT-5.4 using `high` reasoning effort [56].

Evaluation Protocol. We use four benchmarks with complementary emphases: OmniGAIA for open-ended, tool-mediated factual and multi-hop omni-modal reasoning [40], SocialOmni for audio-visual social perception and identity recognition [84], LVOmniBench for long-form audio-video understanding [71], and VideoZeroBench for challenging video-centric reasoning [50]. We evaluate the same fixed subsets for all systems: 100 examples per level on SocialOmni and 100 examples in total across the three LVOmniBench difficulty levels, and on SocialOmni we report only Level-1 and Level-2 Q1 accuracy, omitting the aggregate Level-2 score since it is computed only over Level-2-eligible examples and introduces substantial selection bias. For VideoZeroBench, questions on the same video are asked in a grouped manner. Unless otherwise noted, all models are evaluated with pass@1, and the agent receives only the task instruction and filesystem path(s) to the input files, with no pre-built Skills.

Environments. All sandboxed-agent experiments run in a controlled workspace based on `Ubuntu 22.04.3 LTS` with `Python 3.13.2` and standard Unix utilities. Network and tool permissions

Table 1: ‡Minimax M2.7 uses its own MCP tool for image reading. §Direct image read is disabled for GPT-5.4-`high` in this ablation setting.

Model/System	Harness	OmniGAIA				Social Omni		LVOmniBench				VideoZero Bench
		Easy	Medium	Hard	Avg.	Level1	Level2 Q1 Acc	Easy	Medium	High	Avg.	Level 3
Omnimodal LLMs												
Gemini 3.1 Pro	–	**77.8**	**63.8**	**52.6**	**66.1**	**90.0**	62.0	**80.1**	**71.8**	**48.0**	**69.0**	17.6
Gemini 3 Flash	–	67.2	46.9	37.2	51.7	85.0	**67.0**	72.2	64.1	32.0	59.0	**17.8**
Qwen 3 Omni	–	19.7	10.6	9.0	13.3	61.0	39.0	30.6	17.9	28.0	25.0	5.2
Mimo V2 Omni	–	33.9	22.5	19.1	25.8	64.0	40.0	51.4	25.2	24.0	35.0	7.4
Qwen 3.5 Omni Plus	–	43.5	39.4	29.0	38.7	88.0	40.0	58.3	35.9	44.0	46.0	10.4
Coding Agents (Our Findings)												
GPT-5.4 Low	Codex	63.1	53.8	44.9	55.0	64.0	59.0	58.3	46.2	36.0	48.0	21.0
GPT-5.4 Medium	Codex	66.4	55.0	43.6	56.4	71.0	49.0	**72.2**	43.6	40.0	53.0	24.0
GPT-5.4 High	Codex	70.4	60.0	50.0	61.4	72.0	51.0	66.6	59.0	**44.0**	**58.0**	27.2
GPT-5.4 xHigh	Codex	**82.0**	**75.0**	**64.1**	**75.0**	**75.0**	60.0	66.7	**64.1**	32.0	57.0	**27.6**
Claude Opus 4.6	CC	74.6	69.4	57.7	68.6	54.0	**68.0**	47.2	41.0	28.0	40.0	-
Claude Sonnet 4.6	CC	73.8	62.5	48.8	63.3	-	-	-	-	-	-	-
MiniMax-M2.7‡	CC	38.5	34.4	23.1	33.3	28.0	46.0	50.0	15.4	24.0	30.0	5.8
Kimi K-2.5	CC	72.1	50.6	39.7	55.6	46.0	50.0	55.6	28.0	41.0	43.0	15.6
Tool-use Agent (GPT-5.4 High as backbone)												
OmniAgent	-	77.9	63.1	55.1	66.4	55.0	56.0	50.0	21.0	28.0	33.0	12.2
Agent-Omni	-	54.1	41.9	19.2	41.1	37.0	48.0	56.0	28.0	20.0	36.0	5.0
Text-Only Coding Agent												
GPT-5.4 High§	Codex	70.5	63.8	56.4	64.4	74.0	48.0	52.8	38.5	28.0	41.0	19.0

follow each benchmark in Appendix I.1. The environment provides common multimedia and scientific tooling (e.g., `ffmpeg`/`ffprobe`, Tesseract OCR, ImageMagick, OpenCV, Librosa), and agents may author task-specific scripts and install dependencies when network access is permitted.

2.1 Main Results and In-depth Analysis

From Table 1, coding agents prove highly competitive with native omnimodal LLMs across all four benchmark suites. On OmniGAIA, GPT-5.4 `xhigh` (75.0) clearly outperforms Gemini 3.1 Pro (66.1), especially on medium- and hard-level tasks; on VideoZeroBench, the best coding agent (27.6) also exceeds Gemini 3 Flash (17.8). Native omni models lead on Social Omni Level-1, though Claude Opus 4.6 slightly edges out the best native score on Level-2 Q1. On LVOmniBench, Gemini 3.1 Pro remains the strongest overall, but coding agents stay competitive on the easy and medium subsets. These patterns suggest that benchmarks emphasizing fine-grained temporal perception and audio–visual alignment favor end-to-end omni modeling, while tool-oriented benchmarks such as OmniGAIA align more naturally with the coding-agent paradigm.

> ***Finding 1:*** Coding agents with only text+image capability can perform competitively with, and in some cases outperform, *state-of-the-art* native omnimodal LLMs on omnimodal tasks.

Under the same Codex harness, increasing GPT-5.4 reasoning effort improves OmniGAIA accuracy from 55.0 (`low`) to 75.0 (`xhigh`). The same trend is visible on LVOmniBench, where higher-effort settings outperform lower-effort ones, though not strictly monotonically.

> ***Finding 2:*** Increasing reasoning effort generally improves coding-agent performance, suggesting that omnimodal task success depends not only on model perception capacity but also on the depth of agentic computation.

Tool-use Analysis. Table 2 summarizes tool-call frequencies and success/failure associations; Figure 2 visualizes usage patterns for GPT-5.4 `high` and Claude Opus 4.6 `max`. Across benchmarks, coding agents handle non-text inputs via a staged, tool-mediated pipeline in which `ffprobe`/`ffmpeg` inspect and transform media while `whisper` and `python3` support transcription and downstream

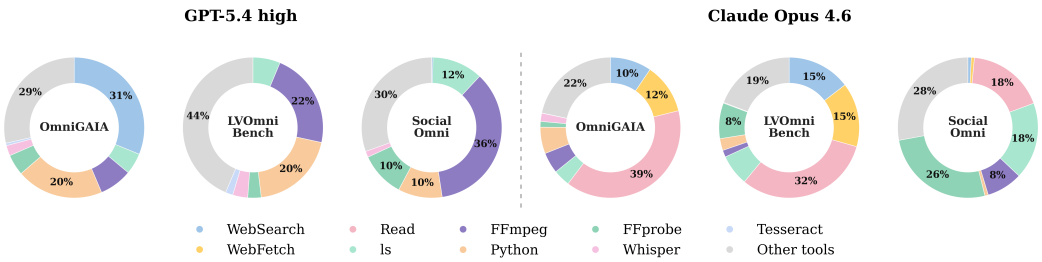

Figure 2: Tool-use distributions of GPT-5.4 `high` and Claude Opus 4.6 `max` across four benchmarks.

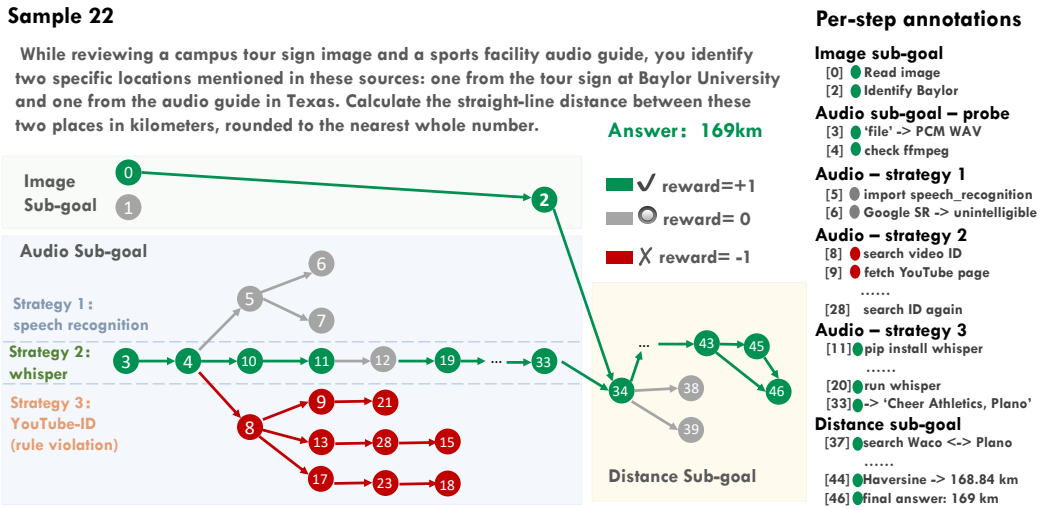

Figure 4: Trajectory DAG of a OMNIGAIA sample (Baylor campus-tour sign × Texas sports-facility audio; ground-truth 169 km). Nodes are annotated as agent steps, coloured by the step-supervisor reward (✓ / 0 / × for $+1$ / 0 / -1). *Audio* sub-goal includes three parallel strategies: *speech_recognition*, *whisper*, and *YouTube-ID lookup*; evidence from the *Image* and *Audio* sub-goals merges into *Distance*.

analysis. Media extraction, transcription, and external search show positive associations with success in several settings, whereas `python3`, our proxy for self-authored tools, is frequently invoked but shows mixed associations; these observational statistics do not establish causal tool benefits. Holding the harness and model family fixed, higher reasoning-effort settings tend to use more tools, most visibly on OmniGAIA and LVOmniBench.

Finding 3: Agents compose media extraction, transcription, search, and scripting into staged pipelines. Tool-use patterns vary across benchmarks and reasoning-effort settings; success/failure associations do not isolate individual tools' causal effects.

Text-Only Ablation. Since some agents (*e.g.*, DeepSeek V4 [11], MiniMax M2.7 [53]) are text-only LLMs, we include a text-only ablation in Table 1, evaluating Codex GPT-5.4-`high` under the same coding-agent setup but without direct image access. It remains competitive with many native omnimodal LLMs, showing only a modest gap to its vision-enabled counterpart on OMNIGAIA and SOCIALOMNI even matching or slightly exceeding it, while on the more vision-centric LVOMNIBENCH and VIDEOZEROBENCH performance drops sharply. This exposes a limitation of current omnimodal benchmarks: many tasks can be solved through text or shallow perception (e.g., OCR) rather than deep visual understanding.

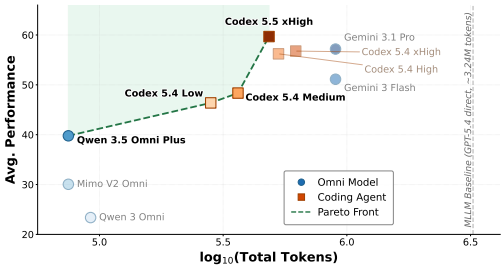

Figure 3: Accuracy versus final-call context length, averaged across four benchmarks. The gray line is an estimated direct-MLLM baseline.

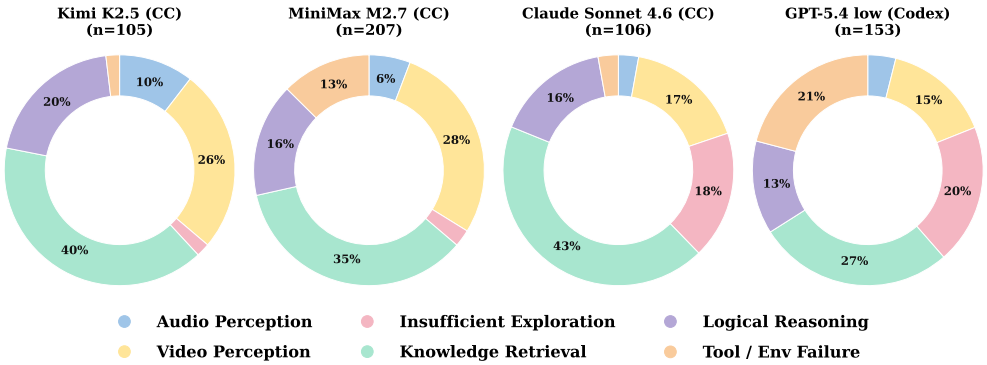

Figure 5: Distribution of primary error types across four coding agents on OMNIGAIA, one pie per agent. The sample size n is shown under each agent name.

Table 2: Tool-use statistics of representative multimodal tools across coding-agent systems on *OmniGAIA*, *SocialOmni* (Level 1 & Level 2), and *LVOmniBench*. "Avg." denotes the average number of calls per case. "Gap" is reported in percentage points and denotes *success_case_ratio* − *failure_case_ratio*. Positive gaps indicate tools more commonly used in successful cases, while negative gaps indicate tools more associated with failure.

System	Harness	Avg. Tool Use	ffmpeg		ffprobe		python3		whisper		web_search		tesseract	
			Avg.	Gap (%)	Avg.	Gap (%)	Avg.	Gap (%)	Avg.	Gap (%)	Avg.	Gap (%)	Avg.	Gap (%)
OmniGAIA														
Kimi K-2.5	CC	22.63	1.45	+12.1	0.34	+7.5	2.15	+9.1	0.62	+6.6	2.89	**+19.2**	0.44	+0.0
MiniMax-M2.7	CC	44.86	1.50	-0.8	0.49	-8.3	7.43	-5.4	0.31	+0.8	2.94	**+2.9**	0.06	-0.8
Claude Sonnet 4.6	CC	40.85	1.97	+0.4	0.54	+1.2	2.49	**+16.5**	0.77	+8.9	3.90	+14.5	0.02	+0.2
GPT-5.4 Low	Codex	25.50	2.59	-2.3	1.19	-6.6	4.36	-9.9	0.08	**+2.7**	5.74	+1.4	0.69	-9.9
GPT-5.4 Medium	Codex	31.83	3.05	-4.0	1.25	-0.1	4.98	-6.6	0.06	**+1.5**	7.97	-4.3	0.81	-7.3
SocialOmni Level1														
GPT-5.4 Low	Codex	16.28	6.94	**+11.1**	2.04	+4.0	2.94	+9.4	0.22	-1.7	0.00	+0.0	0.00	+0.0
GPT-5.4 Medium	Codex	21.52	9.18	+3.4	2.06	+0.0	3.22	+3.2	0.52	**+24.7**	0.00	+0.0	0.00	+0.0
GPT-5.4 High	Codex	19.68	5.06	+12.9	2.18	**+16.5**	2.32	+10.6	0.46	+12.2	0.00	+0.0	0.00	+0.0
GPT-5.4 xHigh	Codex	39.90	9.24	+33.3	2.06	**+34.7**	15.64	+28.0	1.22	+14.7	0.00	+0.0	0.04	-2.7
SocialOmni Level2														
GPT-5.4 Low	Codex	5.59	1.98	-4.0	0.76	-5.2	0.41	-5.7	0.01	-2.4	0.00	**+0.0**	0.00	**+0.0**
GPT-5.4 Medium	Codex	10.07	4.92	+0.0	1.20	**+2.0**	1.28	-16.8	0.04	-1.9	0.00	+0.0	0.00	+0.0
GPT-5.4 High	Codex	13.47	6.75	+0.0	1.25	+0.0	1.05	-3.1	0.00	+0.0	0.10	**+5.9**	0.00	+0.0
GPT-5.4 xHigh	Codex	12.76	3.67	-0.8	1.26	+0.8	1.55	-9.2	0.17	**+6.7**	0.00	+0.0	0.00	+0.0
LVOmniBench														
GPT-5.4 Low	Codex	31.20	7.10	**+0.0**	1.40	**+0.0**	6.45	-16.0	1.31	-9.5	0.00	**+0.0**	0.44	-8.0
GPT-5.4 Medium	Codex	28.72	6.62	+2.1	1.26	+0.0	4.67	+12.8	1.46	**+15.9**	0.00	+0.0	0.29	+8.2
GPT-5.4 High	Codex	38.75	8.56	-3.4	1.20	+0.0	7.64	-15.0	1.34	-1.6	0.00	+0.0	0.66	**+3.8**
GPT-5.4 xHigh	Codex	42.29	10.93	+0.6	1.23	+0.0	8.21	-4.2	1.65	**+13.2**	0.00	+0.0	0.69	+10.1

2.2 Failure Analysis: Taxonomy and Process-level Trajectory.

Given the strong performance of coding agents, we investigate the challenges they face in omni content processing. We propose a new failure mode taxonomy based on task type, and sample 200 trajectories (6,268 steps in total) across different models and benchmarks, along with their primary failure reasons. For each trajectory, we construct a DAG to identify its main trunk and branches, enabling process-level annotation. The resulting taxonomy comprises six failure modes: (i) **Audio Perception and Extraction**, (ii) **Video Perception and Extraction**, (iii) **Insufficient Exploration of Modal Content**, (iv) **Knowledge Retrieval and Factual**, (v) **Logical Reasoning and Calculation**, and (vi) **Tool and Environment Infrastructure**. Detailed definitions and representative examples are deferred to Appendix F. To scale the analysis beyond the 200 hand-annotated trajectories, we use GPT-5.5 as the process-level judge, which we validate against human labels on this small benchmark (see Appendix F for the validation protocol and results).

Figure 5 reports the empirical distribution of the six failure modes across four coding agents on OMNIGAIA. Knowledge-Retrieval dominates the residual errors for three of four agents (Kimi 40%, Sonnet 43%, MiniMax 35%), while Codex-GPT5.4-Low is distinctive in that Tool/Environment Failure (21%) and Insufficient Exploration (20%) each rival its knowledge errors (27%). MiniMax

further stands out for heavy Video-Perception errors (28%), indicating that different agents break down along different axes of the taxonomy. Beyond aggregate rates, Figure 4 dissects a single OMNIGAIA trajectory as a DAG of annotated steps, showing how multiple modes from our taxonomy—audio-perception errors, rule violations, and scraping failures—co-occur with exploration-with-recovery behavior inside the same run, and how the agent ultimately recovers the correct answer.

3 CODE-O: Towards Stronger Coding Agent for Omnimodal Content

Motivated by the competitive task performance and smaller context footprints of closed-source coding agents, we ask whether such capability can be elicited through post-training and reproduced on open-source models. To this end, we propose a strong baseline on Qwen-3.5-9B and Qwen-3.6-27B, with a complete SFT-then-RL training recipe, aiming to explore this direction on open-source coding agents on omnimodal content.

3.1 OMNICODING: Synthesized Omnimodal Coding Agent Trajectory Dataset

We construct OMNICODING, a dataset of 6,035 verifiable question–answer examples paired with video, audio, or image files for multimodal terminal-tool-use agents. Each example follows a unified schema with a question, a list of acceptable ground-truth answers, media paths, and source/modality/task metadata. We draw from four complementary sources, OmniGAIA-SFT-2K [40], OmniVideoBench [35], AVUTBenchmark [93], and Video-MME-v2 [20], keeping only the Video-MME-v2 subset requiring audio-visual or temporal reasoning. Table 10 and Figure 11 report the per-source composition and media-duration distribution. Construction proceeds in three steps: **(i) Normalization**, unifying all sources into the same schema with deduplicated questions and media consolidated into a flat directory; **(ii) Verifiability**, regenerating open-ended questions with ground-truth lists covering common surface variants (units, abbreviations, capitalization, number formats) and converting multiple-choice into option-letter answer sets; and **(iii) Cross-review**, verifying answer lists and task coverage. We stratified-split the data into 4,042 SFT and 1,993 RL examples by source-task key, and use exact match against the ground-truth list (extracted from an `<answer>` tag) as the binary RL reward. Trajectory generation, training and evaluation overlap are detailed in Appendices I.2 and I.3.

3.2 Method and Training Infrastructure

We train the terminal agent with supervised warm-start followed by reinforcement learning from verifiable rewards. All training examples follow the schema $x = (q, \mathcal{F}, \mathcal{Y})$, where multiple-choice tasks use option-letter answer sets and open-ended tasks are assigned curated answer sets covering common aliases, units, abbreviations, capitalization, and number formats. The agent submits its final answer inside `<answer>`.

Supervised warm start. For examples with demonstration trajectories, we fine-tune on assistant-generated tokens only, masking terminal outputs, tool responses, and environment observations from the loss. This stage teaches the basic interaction protocol: inspect files, run commands, create intermediate artifacts, recover from tool failures, and submit answers in the required format.

Reward design. We design a process-aware verifiable reward that goes beyond binary answer matching, addressing three failure patterns: (1) the agent guesses without engaging the workspace, (2) it ignores the relevant modality (e.g., answering a video question from text alone), and (3) it emits malformed, escape-attempting, or otherwise broken tool calls. For each rollout τ we compute

$$R(x, \tau) = \begin{cases} 0, & \text{if } \tau \text{ is removed,} \\ \text{base}(\tau) + r_{\text{mod}}(\tau) + r_{\text{tool}}(\tau), & \text{otherwise,} \end{cases}$$

where removed rollouts exceed the timeout, step limit, or context limit. The reward has the following components:

- **Three-tier base reward.** $\text{base}(\tau) = w_{\text{correct}}$ if gated-correct (normalized exact match against $\mathcal{Y}$, modality engaged, ≥ 1 tool call, no escape attempt); else $w_{\text{fmt}+}$ if the agent terminated cleanly via `task_complete` with a non-empty `<answer>`; else $w_{\text{fmt}-}$. The format bonus does *not* stack with correctness; it exists solely to keep gradient signal on tried-correctly-but-answered-wrong trajectories that would otherwise collapse the group to zero variance.

- **Modality penalty** r_{mod}**.** A fixed negative weight w_{mod} is charged when the task contains video or audio but the agent invokes no tool from the corresponding universe (`ffmpeg`, `whisper`, etc.), detected by substring matching on bash commands.
- **Bad-tool penalty** r_{tool}**.** A weighted ratio $-w_{\text{tool}} \cdot P_{\text{bad}}$ with $P_{\text{bad}} = (n_{\text{unparseable}} + n_{\text{disallowed}} + n_{\text{escape}} + n_{\text{syntax-fail}})/(1 + n_{\text{tool}})$ aggregates malformed calls, disallowed tools, gold-answer leakage attempts, and persistent shell errors.

Weight values are listed in Appendix I.4. We further apply group-level mechanisms: **(i) group advantage normalization** centers rewards within each group of G sibling rollouts; **(ii) zero-variance masking** drops groups where all rollouts succeed or fail; **(iii) active-only baseline** excludes removed samples from group statistics; and **(iv) dynamic filter** drops groups with ≤ 1 active sample or near-zero variance originated from DAPO [97].

GSPO update. For each task we sample G rollouts $\tau_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$ and compute group-normalized advantages $A_i = (R_i - \bar{R})/(\sigma_R + \epsilon_A)$. Let $z_i = (z_{i,1}, \ldots, z_{i,L_i})$ be the model-generated tokens in τ_i (tool outputs masked from the loss) and $c_{i,\ell}$ the transcript prefix before $z_{i,\ell}$. Following GSPO [106], we use a length-normalized sequence-level importance ratio and optimize the clipped objective:

$$\rho_i(\theta) = \exp\left(\frac{1}{L_i}\sum_{\ell=1}^{L_i}\left[\log \pi_\theta(z_{i,\ell} \mid c_{i,\ell}) - \log \pi_{\theta_{\text{old}}}(z_{i,\ell} \mid c_{i,\ell})\right]\right),$$

$$\mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}_x\left[\frac{1}{G}\sum_{i=1}^{G}\min\Big(\rho_i(\theta)A_i,\ \text{clip}\big(\rho_i(\theta), 1-\epsilon_-, 1+\epsilon_+\big)A_i\Big)\right].$$

Since reward is assigned to whole trajectories, sequence-level clipping is a natural fit: one importance ratio weights all generated tokens while tool outputs stay masked.

Training infrastructure. Our RL system decouples sandboxed rollout generation from policy optimization. Rollout workers run the current policy in isolated workspaces, execute terminal commands, collect trajectories, parse answers, compute reward components, and return old log-probabilities and masks; the trainer consumes these logs and applies GSPO without executing tools or touching raw media. This separation is useful because omni-modal rollouts involve non-differentiable operations, while the optimizer only needs token sequences, masks, old log-probabilities, and scalar rewards. Sandboxes are hosted on a Slurm cluster with CPU/GPU nodes and storage, the trainer runs on cloud servers, and the two are connected via a *Cloudflare* tunnel for trajectory collection.

3.3 Experiment Setups and Results

Agent Harness. To train OMNICODING, we build a local terminal-agent harness adapted from Terminus-KIRA [30] for converting raw multimodal files into small, checkable agent operations. At each episode, the environment creates an isolated workspace with the question and staged media files. The media remains in the filesystem rather than being packed into the model context, so the agent must decide how to inspect it through audio extraction, frame sampling, OCR/ASR, metadata reading, scripting, and intermediate verification. The visible action space is deliberately compact: `execute_commands` runs one command in a persistent shell, `image_read` inspects an image as a native multimodal observation, and `task_complete` ends the episode. Higher-level media processing is expressed through code and command-line tools rather than benchmark-specific APIs, keeping the harness lightweight but expressive. Unlike general software-engineering harnesses [75, 68], our scaffold focuses on offline multimedia processing and trajectory collection: tool calls, terminal/image observations, final answers, exit reasons, and rewards are serialized for SFT filtering, RL replay, and failure analysis. Full details are in Appendix J.1.

Baselines and Evaluation Protocol. We train Qwen3.5-9B and Qwen3.6-27B and compare against three baselines: **(i)** training-free direct MLLM inference, where the model is fed visual frames sampled at fps=1 (capped at 512) together with Whisper-large ASR transcripts; **(ii)** the same MLLM evaluated under our harness with direct image reading; and **(iii)** SFT warm-up baseline. We additionally include Codex GPT-5.4 (low) under our harness as a reference. The thinking budget is set to 4K tokens; if the model fails to produce a correctly formatted answer, we issue a multi-turn reminder to re-answer in the required format, and if reasoning is truncated, the model is allowed to continue thinking. We evaluate on the same benchmark suite as before, reporting pass@1 throughout.

Table 3: Performance of Qwen3.5-9B and Qwen3.6-27B under different settings on four benchmarks. For the Direct MLLM setting in the OmniGAIA benchmark, we provide models with tools following their codebase. The thinking budget is fixed at 4K tokens for all settings.

Model/Setting	OmniGAIA				Social Omni		LVOmniBench				VideoZero Bench
	Easy	Medium	Hard	Avg.	Level1	Level2	Easy	Medium	High	Avg.	Level 3
Strong Baselines											
OmniAtlas-Omni-30B-A3B	31.1	18.8	9.0	20.8	50.0	61.0	44.4	23.1	36.0	34.0	6.0
Codex-GPT-5.4 Low	63.1	53.8	44.9	55.0	64.0	59.0	58.3	46.2	36.0	48.0	21.0
Qwen3.5-9B											
Direct MLLM	23.0	16.2	23.1	20.0	37.0	59.0	44.4	33.3	16.0	33.0	7.2
Baseline (direct image)	23.8	14.4	17.9	18.3	41.0	58.0	41.7	15.4	36.0	30.0	5.9
CODE-O (Ours, SFT)	30.0	21.0	17.0	23.1	50.0	61.0	48.6	28.9	16.0	33.0	9.5
CODE-O (Ours, RL)	32.8	26.3	24.4	28.1	52.0	63.0	72.2	41.0	44.0	53.0	15.8
Δ vs Direct MLLM	*+42.6%*	*+62.3%*	*+5.6%*	*+40.5%*	*+40.5%*	*+6.8%*	*+62.6%*	*+23.1%*	*+175.0%*	*+60.6%*	*+119.4%*
Qwen3.6-27B											
Direct MLLM	44.3	38.1	**28.6**	38.2	61.0	47.0	66.7	46.2	36.0	51.0	7.4
Baseline (direct image)	32.8	18.8	23.1	24.4	46.0	55.0	52.8	33.3	32.0	40.0	13.0
CODE-O (Ours, SFT)	**50.0**	**45.0**	28.2	**43.3**	**64.0**	**67.0**	**75.0**	**53.8**	**48.0**	**60.0**	**18.4**
Δ vs Direct MLLM	*+12.9%*	*+18.1%*	*−1.4%*	*+13.4%*	*+4.9%*	*+42.6%*	*+12.4%*	*+16.5%*	*+33.3%*	*+17.6%*	*+148.6%*

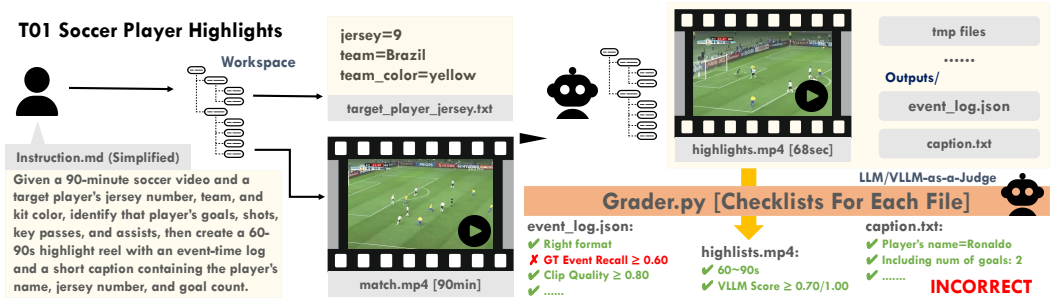

Figure 6: Overview of our benchmark, illustrated with Task T01. The coding agent is required to generate a highlight clip and caption from a soccer video and a player query, and its outputs are evaluated by deterministic checks and LLM/VLM judging for event accuracy and task correctness.

Experiment Results and Analysis. As shown in Table 3, model-tool-environment finetuned models are more adaptive to harness and perform much better than baseline and even direct MLLM settings that take up to 512 images. CODE-O-RL-9B shows extremely reasoning and terminal tool-use capability, achieving 15.8 on VideoZeroBench, which is almost competitive to Gemini 3.1 Pro.

4 TERMINALBENCH-O: From Many-modality Understanding to Processing

Beyond answering questions, practical agents must edit videos, audit image databases, and produce spec-compliant artifacts. OmniGAIA [40] and LVOmniBench [71] emphasize QA, while Terminal-Bench [51] and Claw-Eval [96] cover broader terminal workflows. We introduce TERMINALBENCH-O to evaluate multimodal processing through intermediate artifacts and final-output compliance (Table 5); concurrent MMTB [23] also studies multimedia-file workflows.

Design Principles. TERMINALBENCH-O comprises 50 challenging tasks across video (38%), audio (26%), image (14%), document (4%), and cross-modal (18%) categories. Four core principles guide the design: **(i) Real-world cases**, where all tasks are drawn from authentic sources (Internet Archive, HuggingFace datasets, public videos) rather than synthetic constructions; **(ii) Multi-dimensional grading**, where each task is scored along 4–8 weighted dimensions covering accuracy, cross-document consistency, and output quality; **(iii) Shortcut-resistant design**, which combines closed vocabularies, cross-file ID consistency checks, decoy injection, and word limits to prevent gaming; and **(iv) Process-level evaluation**, which validates intermediate artifacts and reasoning chains rather than only final answers.

Task Categories. Representative tasks include: *Video Understanding & Processing* (T01a: extract 60-90s highlights from 90-min soccer broadcast by jersey number; T03a: auto-edit 5-min final cut from 3-camera meeting footage); *Audio Forensics* (T107a: segment 22-min concert recording into songs with tempo/defect detection; T108a: extract rhetorical devices and thematic structure from political speech); *Image Database Analysis* (T13a: detect wood defects with pixel-level masks;

Table 5: Comparison of TERMINALBENCH-O with existing benchmarks.

Benchmark	Modalities	# Size	Target Capability	Real	Cross-source	MMU	MMP
OmniGAIA [40]	V+A+I+T	360	QA, factuality seeking	◐	✘	✔	✘
Terminal-Bench 2.0 [51]	T+F	89	Terminal workflows	✔	◐	✘	◐
Claw-Eval [96]	T+I+V+D	300	Agent workflows	✔	◐	◐	◐
WildClawBench [14]	T+I+V+A+D	60	In-the-wild agent tasks	✔	◐	◐	◐
TERMINALBENCH-O (Ours)	**T+V+A+I+D+F**	**50**	**Many-modality Processing**	✔	✔	✔	✔

✔ denotes full support, ◐ denotes partial or mixed support, and ✘ denotes absent support. MMU and MMP mean many-modality understanding and processing. **V**: video, **A**: audio, **I**: image, **T**: text, **D**: document, **F**: files.

T50a: audit mislabeled images in CIFAR-10 subset). Unlike QA benchmarks, TERMINALBENCH-O requires agents to *create* multimodal outputs (edited videos, text+video interleaved reports) that satisfy strict format and consistency constraints.

Evaluation Protocol. Each task defines a weighted scoring formula score $= \sum_i w_i \cdot m_i$ where m_i are normalized metrics (IoU, F1, accuracy, LLM-judged quality). A submission passes if score ≥ 0.70 *and* all gating conditions hold: required files exist, JSON schemas validate, cross-document IDs match, and outputs comply with closed vocabularies. Gating failures result in zero score regardless of partial correctness. For open-ended outputs (summaries, captions), we use LLM judges with detailed rubrics; for temporal tasks (video segmentation, audio boundaries), we apply tolerance windows (±2s) and Hungarian matching for IoU computation.

Table 4: Overall accuracy and average score of different coding agents on TERMINALBENCH-O. †Models without native vision capabilities use the image-read implemented with Qwen3.5-Flash.

Model	Harness	Pass(%)	Avg.
GPT-5.5 xHigh	Codex	**24**	**0.71**
GLM-5.1 (V)†	CC	16	0.69
DeepSeek V4 Flash (V)†	CC	10	0.60
Kimi K2.6	CC	8	0.56
MiniMax M2.7 (V)†	CC	4	0.55
Qwen3.6-27B	Ours	0	0.26
CODE-O-27B-SFT	Ours	0	0.29

Results Analysis. As shown in Table 4, GPT-5.5 obtains the strongest results among the evaluated settings, with high and xHigh achieving the highest average scores. CODE-O-27B-SFT improves over Qwen3.6-27B in average score (0.26 to 0.29), but both retain a 0% strict pass rate. Overall, the results show that TERMINALBENCH-O remains challenging across different model and harness configurations, especially for tasks requiring reliable long-horizon tool use.

5 Related Works

Omnimodal Models and Evaluation Recent multimodal foundation models have advanced from single-modality extensions to omni-modal systems that jointly process speech, audio, images, and video [10, 67, 86, 62, 72]. As models push toward unified vision-and-speech interaction, evaluation has correspondingly shifted from task-specific video or audio benchmarks [38, 47, 18, 80, 36] to omni-modal suites that more explicitly target joint reasoning over visual, auditory, and textual cues, with growing attention to long videos, grounded interaction, and real-world complexity [42, 24, 102, 40, 63, 71, 84]. Yet our audit shows that many supposedly audio-visual tasks still allow strong shortcuts or rely only weakly on audio [27], motivating more careful benchmark construction.

Multimodal Agentic Tool-Use and Sandboxed Agents. Tool use has become a central mechanism for extending LLM agents beyond their parametric knowledge [95, 66, 61, 58, 94, 76], and increasingly improves multimodal understanding via active search, code generation, and tool orchestration without retraining a monolithic model [25, 105, 101, 99, 100, 70, 45, 37]. The most mature form of sandboxed agency has emerged in software engineering, where models resolve real issues inside an execution environment [26, 89, 75, 32, 8], but existing work primarily targets natively textual or computational tasks.

6 Conclusion

We study offline omnimodal understanding through the lens of sandboxed coding agents in both training-free and fine-tune settings. Across multiple benchmarks, these agents solve a surprisingly large fraction of video-audio tasks by selectively retrieving and processing evidence from raw inputs, often rivaling or surpassing native omnimodal systems. We hope this work motivates a deeper study of multimodal intelligence at the agent sandbox interface.

Acknowledgement

We thank Chenlong Wang, Jieneng Chen, Xirui Li, Ming Li, Kwesi Cobbina and Misaki Wang for their invaluable support and discussion. Dongping Chen is supported by Modal, Tavily, and Linear's academic research project.

References

[1] Anthropic. Claude code overview. https://code.claude.com/docs/en/overview, 2026. Accessed: 2026-04-01.

[2] Anthropic. Introducing claude opus 4.6. https://www.anthropic.com/news/claude-opus-4-6, 2026. Accessed: 2026-04-01.

[3] Anthropic. Introducing claude sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6, 2026. Accessed: 2026-04-01.

[4] Avi Arora, Jinu Jang, and Roshanak Zilouchian Moghaddam. Setupbench: Assessing software engineering agents' ability to bootstrap development environments. *arXiv preprint arXiv:2507.09063*, 2025.

[5] Fuqing Bie, Shiyu Huang, Xijia Tao, Zhiqin Fang, Leyi Pan, Junzhe Chen, Min Ren, Liuyu Xiang, and Zhaofeng He. Omniplay: Benchmarking omni-modal models on omni-modal game playing. *arXiv preprint arXiv:2508.04361*, 2025.

[6] Michael Bolin. Unrolling the codex agent loop. https://openai.com/index/unrolling-the-codex-agent-loop/, 2026. Accessed: 2026-04-01.

[7] Celia Chen. Unlocking the codex harness: how we built the App Server. https://openai.com/index/unlocking-the-codex-harness/, 2026. Accessed: 2026-04-01.

[8] Daixuan Cheng, Shaohan Huang, Yuxian Gu, Huatong Song, Guoxin Chen, Li Dong, Wayne Xin Zhao, Ji-Rong Wen, and Furu Wei. Llm-in-sandbox elicits general agentic intelligence. *arXiv preprint arXiv:2601.16206*, 2026.

[9] Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, and Lidong Bing. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*, 2024.

[10] Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*, 2024.

[11] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf, April 2026. Technical report.

[12] Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.

[13] Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

[14] Shuangrui Ding, Xuanlang Dai, Long Xing, Shengyuan Ding, Ziyu Liu, Jingyi Yang, Penghui Yang, Zhixiong Zhang, Xilin Wei, Yubo Ma, Haodong Duan, Jing Shao, Jiaqi Wang, Dahua Lin, Kai Chen, and Yuhang Zang. Wildclawbench. https://github.com/InternLM/WildClawBench, 2026. GitHub repository.

[15] Tulsee Doshi. Gemini 3 Flash: frontier intelligence built for speed. https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/, 2025. Accessed: 2026-04-01.

[16] Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, et al. Agentprocessbench: Diagnosing step-level process quality in tool-using agents. *arXiv preprint arXiv:2603.14465*, 2026.

[17] Yukang Feng, Jianwen Sun, Zelai Yang, Jiaxin Ai, Chuanhao Li, Zizhen Li, Fanrui Zhang, Kang He, Rui Ma, Jifan Lin, Jie Sun, Yang Xiao, Sizhuo Zhou, Wenxiao Wu, Yiming Liu, Pengfei Liu, Yu Qiao, Shenglin Zhang, and Kaipeng Zhang. Longcli-bench: A preliminary benchmark and study for long-horizon agentic programming in command-line interfaces. *arXiv preprint arXiv:2602.14337*, 2026.

[18] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*, 2024.

[19] Chaoyou Fu, Haojia Lin, Zuwei Long, Yunhang Shen, Yuhang Dai, Meng Zhao, Yi-Fan Zhang, Shaoqi Dong, Yangze Li, Xiong Wang, Haoyu Cao, Di Yin, Long Ma, Xiawu Zheng, Rongrong Ji, Yunsheng Wu, Ran He, Caifeng Shan, and Xing Sun. Vita: Towards open-source interactive omni multimodal llm. *arXiv preprint arXiv:2408.05211*, 2024.

[20] Chaoyou Fu, Haozhi Yuan, Yuhao Dong, Yi-Fan Zhang, Yunhang Shen, Xiaoxing Hu, Xueying Li, Jinsen Su, Chengwu Long, Xiaoyao Xie, et al. Video-mme-v2: Towards the next stage in benchmarks for comprehensive video understanding. *arXiv preprint arXiv:2604.05015*, 2026.

[21] Arushi Goel, Sreyan Ghosh, Vatsal Agarwal, Nishit Anand, Kaousheik Jayakumar, Lasha Koroshinadze, Yao Xu, Katie Lyons, James Case, Karan Sapra, Kevin J. Shih, Siddharth Gururani, Abhinav Shrivastava, Ramani Duraiswami, Dinesh Manocha, Andrew Tao, Bryan Catanzaro, Mohammad Shoeybi, and Wei Ping. Mmou: A massive multi-task omni understanding and reasoning benchmark for long and complex real-world videos. *arXiv preprint arXiv:2603.14145*, 2026.

[22] Kaixiong Gong, Kaituo Feng, Bohao Li, Yibing Wang, Mofan Cheng, Shijia Yang, Jiaming Han, Benyou Wang, Yutong Bai, Zhuoran Yang, and Xiangyu Yue. Av-odyssey bench: Can your multimodal llms really understand audio-visual information? *arXiv preprint arXiv:2412.02611*, 2024.

[23] Chiyeong Heo, Jaechang Kim, Junhyuk Kwon, Hoyoung Kim, Dongmin Park, Jonghyun Lee, and Jungseul Ok. MMTB: Evaluating terminal agents on multimedia-file tasks. *arXiv preprint arXiv:2605.10966*, 2026.

[24] Jack Hong, Shilin Yan, Jiayin Cai, Xiaolong Jiang, Yao Hu, and Weidi Xie. Worldsense: Evaluating real-world omnimodal understanding for multimodal llms. *arXiv preprint arXiv:2502.04326*, 2025.

[25] Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. *arXiv preprint arXiv:2406.09403*, 2024.

[26] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kilian Lieret, Karthik Narasimhan, and Ofir Press. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[27] Geewook Kim and Minjoon Seo. Do modern video-llms need to listen? a benchmark audit and scalable remedy. *arXiv preprint arXiv:2509.17901*, 2025.

[28] Seunghee Kim, Ingyu Bang, Seokgyu Jang, Changhyeon Kim, Sanghwan Bae, Jihun Choi, Richeng Xuan, and Taeuk Kim. Omhbench: Benchmarking balanced and grounded omni-modal multi-hop reasoning. *arXiv preprint arXiv:2508.16198*, 2025.

[29] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, et al. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

[30] KRAFTON AI and Ludo Robotics. Terminus-kira: Boosting frontier model performance on terminal-bench with minimal harness. `https://github.com/krafton-ai/KIRA`, 2026. Accessed: 2026-05-02.

[31] Mohammed Irfan Kurpath, Jaseel Muhammad Kaithakkodan, Jinxing Zhou, Sahal Shaji Mullappilly, Mohammad Almansoori, Noor Ahsan, Beknur Kalmakhanbet, Sambal Shikhar, Rishabh Lalla, Jean Lahoud, Mariette Awad, Fahad Shahbaz Khan, Salman Khan, Rao Muhammad Anwer, and Hisham Cholakkal. A benchmark and agentic framework for omni-modal reasoning and tool use in long videos. *arXiv preprint arXiv:2512.16978*, 2025.

[32] Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring ai ability to complete long software tasks. *arXiv preprint arXiv:2503.14499*, 2025.

[33] Hwiwon Lee, Ziqi Zhang, Hanxiao Lu, and Lingming Zhang. Sec-bench: Automated benchmarking of llm agents on real-world software security tasks. *arXiv preprint arXiv:2506.11791*, 2025.

[34] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

[35] Caorui Li, Yu Chen, Yiyan Ji, Jin Xu, Zhenyu Cui, Shihao Li, Yuanxing Zhang, Wentao Wang, Zhenghao Song, Dingling Zhang, Ying He, Haoxiang Liu, Yuxuan Wang, Qiufeng Wang, Jiafu Tang, Zhenhe Wu, Jiehui Luo, Zhiyu Pan, Weihao Xie, Chenchen Zhang, Zhaohui Wang, Jiayi Tian, Yanghai Wang, Zhe Cao, Minxin Dai, Ke Wang, Runzhe Wen, Yinghao Ma, Yaning Pan, Sungkyun Chang, Termeh Taheri, Haiwen Xia, Christos Plachouras, Emmanouil Benetos, Yizhi Li, Ge Zhang, Jian Yang, Tianhao Peng, Zili Wang, Minghao Liu, Junran Peng, Zhaoxiang Zhang, and Jiaheng Liu. Omnivideobench: Towards audio-visual understanding evaluation for omni mllms. *arXiv preprint arXiv:2510.10689*, 2025.

[36] Guangyao Li, Yake Wei, Yapeng Tian, Chenliang Xu, Ji-Rong Wen, and Di Hu. Learning to answer questions in dynamic audio-visual scenarios. *arXiv preprint arXiv:2203.14072*, 2022.

[37] Keliang Li, Yansong Li, Hongze Shen, Mengdi Liu, Hong Chang, and Shiguang Shan. Lenswalk: Agentic video understanding by planning how you see in videos. *arXiv preprint arXiv:2603.24558*, 2026.

[38] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. Mvbench: A comprehensive multi-modal video understanding benchmark. *arXiv preprint arXiv:2311.17005*, 2023.

[39] Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Haoran Lyu, Ze Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Yiqi Xue, Steven Dillmann, and Han chung Lee. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

[40] Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Shijian Wang, Guanting Dong, Jiajie Jin, Hao Wang, Yinuo Wang, Ji-Rong Wen, Yuan Lu, and Zhicheng Dou. Omnigaia: Towards native omni-modal ai agents. *arXiv preprint arXiv:2602.22897*, 2026.

[41] Yadong Li, Haoze Sun, Mingan Lin, Tianpeng Li, Guosheng Dong, Tao Zhang, Bowen Ding, Wei Song, Zhenglin Cheng, Yuqi Huo, Song Chen, Xu Li, Da Pan, Shusen Zhang, Xin Wu, Zheng Liang, Jun Liu, Tao Zhang, Keer Lu, Yaqi Zhao, Yanjun Shen, Fan Yang, Kaicheng Yu, Tao Lin, Jianhua Xu, Zenan Zhou, and Weipeng Chen. Baichuan-omni technical report. *arXiv preprint arXiv:2410.08565*, 2024.

[42] Yizhi Li, Yinghao Ma, Ge Zhang, Ruibin Yuan, Kang Zhu, Hangyu Guo, Yiming Liang, Jiaheng Liu, Zekun Wang, Jian Yang, Siwei Wu, Xingwei Qu, Jinjie Shi, Xinyue Zhang, Zhenzhu Yang, Yidan Wen, Yanghai Wang, Shihao Li, Zhaoxiang Zhang, Zachary Liu, Emmanouil Benetos, Wenhao Huang, and Chenghua Lin. Omnibench: Towards the future of universal omni-language models. *arXiv preprint arXiv:2409.15272*, 2024.

[43] Zhengyang Liang, Yan Shu, Xiangrui Liu, Minghao Qin, Kaixin Liang, Paolo Rota, Nicu Sebe, Zheng Liu, and Lizi Liao. Video-browsecomp: Benchmarking agentic video research on open web. *arXiv preprint arXiv:2512.23044*, 2025.

[44] Geuntaek Lim, Minho Shim, Sungjune Park, Jaeyun Lee, Inwoong Lee, Taeoh Kim, Dongyoon Wee, and Yukyung Choi. Video-oasis: Rethinking evaluation of video understanding. *arXiv preprint arXiv:2603.29616*, 2026.

[45] Huawei Lin, Yunzhi Shi, Tong Geng, Weijie Zhao, Wei Wang, and Ravender Pal Singh. Agent-omni: Test-time multimodal reasoning via model coordination for understanding anything. *arXiv preprint arXiv:2511.02834*, 2025.

[46] Jiaqi Liu, Zipeng Ling, Shi Qiu, Yanqing Liu, Siwei Han, Peng Xia, Haoqin Tu, Zeyu Zheng, Cihang Xie, Charles Fleming, Mingyu Ding, and Huaxiu Yao. Omnimem: Autoresearch-guided discovery of lifelong multimodal agent memory. *arXiv preprint arXiv:2604.01007*, 2026.

[47] Yuanxin Liu, Shicheng Li, Yi Liu, Yuxiang Wang, Shuhuai Ren, Lei Li, Sishuo Chen, Xu Sun, and Lu Hou. Tempcompass: Do video llms really understand videos? *arXiv preprint arXiv:2403.00476*, 2024.

[48] Run Luo, Ting-En Lin, Haonan Zhang, Yuchuan Wu, Xiong Liu, Min Yang, Yongbin Li, Longze Chen, Jiaming Li, Lei Zhang, Xiaobo Xia, Hamid Alinejad-Rokny, and Fei Huang. Openomni: Advancing open-source omnimodal large language models with progressive multimodal alignment and real-time self-aware emotional speech synthesis. *arXiv preprint arXiv:2501.04561*, 2025.

[49] Matias Martinez and Xavier Franch. What's in a benchmark? the case of swe-bench in automated program repair. *arXiv preprint arXiv:2602.04449*, 2026.

[50] Jiahao Meng, Yue Tan, Qi Xu, Haochen Wang, Zhongwei Ren, Weisong Liu, Yuhao Wang, Renrui Zhang, Haodong Duan, and Yunhai Tong. Videozerobench: Probing the limits of video mllms with spatio-temporal evidence verification. *arXiv preprint arXiv:2604.01569*, 2026.

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

[52] Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbj"orn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

[53] MiniMax. MiniMax M2.7: Early echoes of self-evolution. `https://www.minimax.io/news/minimax-m27-en`, 2026. Accessed: 2026-04-01.

[54] Nous Research. Hermes agent: The agent that grows with you. `https://github.com/nousresearch/hermes-agent`, 2026. Accessed: 2026-04-27.

[55] OpenAI. Introducing swe-bench verified. OpenAI blog, 2024. Updated February 24, 2025.

[56] OpenAI. Introducing GPT-5.4. `https://openai.com/index/introducing-gpt-5-4/`, 2026. Accessed: 2026-04-01.

[57] OpenClaw. OpenClaw: Personal AI Assistant. `https://github.com/openclaw/openclaw`, 2026. GitHub repository, accessed April 1, 2026.

[58] Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[59] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

[60] Thanosan Prathifkumar, Noble Saji Mathews, and Meiyappan Nagappan. Does swe-bench-verified test agent ability or model memory? *arXiv preprint arXiv:2512.10218*, 2025.

[61] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

[62] Qwen Team. Qwen3.5-Omni: Scaling up, toward native omni-modal AGI. `https://qwen.ai/blog?id=qwen3.5-omni`, 2026. Accessed: 2026-04-01.

[63] Ahmed Y. Radwan, Christos Emmanouilidis, Hina Tabassum, Deval Pandya, and Shaina Raza. Sonic-o1: A real-world benchmark for evaluating multimodal large language models on audio-video understanding. *arXiv preprint arXiv:2601.21666*, 2026.

[64] David Rein, Joel Becker, Amy Deng, Seraphina Nix, Chris Canal, Daniel O'Connel, Pip Arnott, Ryan Bloom, Thomas Broadley, Katharyn Garcia, Brian Goodrich, Max Hasin, Sami Jawhar, Megan Kinniment, Thomas Kwa, Aron Lajko, Nate Rush, Lucas Jun Koba Sato, Sydney Von Arx, Ben West, Lawrence Chan, and Elizabeth Barnes. Hcast: Human-calibrated autonomy software tasks. *arXiv preprint arXiv:2503.17354*, 2025.

[65] S Sakshi, Utkarsh Tyagi, Sonal Kumar, Ashish Seth, Ramaneswaran Selvakumar, Oriol Nieto, Ramani Duraiswami, Sreyan Ghosh, and Dinesh Manocha. Mmau: A massive multi-task audio understanding and reasoning benchmark. *arXiv preprint arXiv:2410.19168*, 2024.

[66] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[67] Guangzhi Sun, Wenyi Yu, Changli Tang, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, Yuxuan Wang, and Chao Zhang. video-salmonn: Speech-enhanced audio-visual large language models. *arXiv preprint arXiv:2406.15704*, 2024.

[68] SWE-agent Team. mini-swe-agent: The minimal ai software engineering agent. `https://github.com/SWE-agent/mini-swe-agent`, 2025. Accessed: 2026-05-02.

[69] Keda Tao, Wenjie Du, Bohan Yu, Weiqiang Wang, Jian Liu, and Huan Wang. Active perception agent for omnimodal audio-video understanding. *arXiv preprint arXiv:2512.23646*, 2025.

[70] Keda Tao, Wenjie Du, Bohan Yu, Weiqiang Wang, Jian Liu, and Huan Wang. Active perception agent for omnimodal audio-video understanding. *arXiv preprint arXiv:2512.23646*, 2025.

[71] Keda Tao, Yuhua Zheng, Jia Xu, Wenjie Du, Kele Shao, Hesong Wang, Xueyi Chen, Xin Jin, Junhan Zhu, Bohan Yu, Weiqiang Wang, Jian Liu, Can Qin, Yulun Zhang, Ming-Hsuan Yang, and Huan Wang. Lvomnibench: Pioneering long audio-video understanding evaluation for omnimodal llms. *arXiv preprint arXiv:2603.19217*, 2026.

[72] The Gemini Team. Gemini 3.1 pro: A smarter model for your most complex tasks. `https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/`, February 2026. Google Blog, accessed 2026-04-01.

[73] Chenyu Wang, Weixin Luo, Qianyu Chen, Haonan Mai, Jindi Guo, Sixun Dong, Xiaohua Xuan, Zhengxin Li, Lin Ma, and Shenghua Gao. Mllm-tool: A multimodal large language model for tool agent learning. *arXiv preprint arXiv:2401.10727*, 2024.

[74] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. *arXiv preprint arXiv:2402.01030*, 2024.

[75] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

[76] Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and Eugene Siow. Mcp-bench: Benchmarking tool-using llm agents with complex real-world tasks via mcp servers. *arXiv preprint arXiv:2508.20453*, 2025.

[77] Lai Wei, Liangbo He, Jun Lan, Lingzhong Dong, Yutong Cai, Siyuan Li, Huijia Zhu, Weiqiang Wang, Linghe Kong, Yue Wang, Zhuosheng Zhang, and Weiran Huang. Zooming without zooming: Region-to-image distillation for fine-grained multimodal perception. *arXiv preprint arXiv:2602.11858*, 2026.

[78] Xin-Cheng Wen, Binbin Chen, Haoxuan Lan, Hang Yu, Peng Di, and Cuiyun Gao. Swe-fuse: Empowering software agents via issue-free trajectory learning and entropy-aware rlvr training. *arXiv preprint arXiv:2603.07927*, 2026.

[79] Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Holden Karnofsky, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts. *arXiv preprint arXiv:2411.15114*, 2024.

[80] Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *arXiv preprint arXiv:2407.15754*, 2024.

[81] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

[82] Chunqiu Steven Xia, Zhe Wang, Yan Yang, Yuxiang Wei, and Lingming Zhang. Live-swe-agent: Can software engineering agents self-evolve on the fly? *arXiv preprint arXiv:2511.13646*, 2025.

[83] Xiaomi. MiMo-V2-Omni. `https://mimo.xiaomi.com/mimo-v2-omni`, 2026. Accessed: 2026-04-01.

[84] Tianyu Xie, Jinfa Huang, Yuexiao Ma, Rongfang Luo, Yan Yang, Wang Chen, Yuhui Zeng, Ruize Fang, Yixuan Zou, Xiawu Zheng, Jiebo Luo, and Rongrong Ji. Socialomni: Benchmarking audio-visual social interactivity in omni models. *arXiv preprint arXiv:2603.16859*, 2026.

[85] 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, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*, 2024.

[86] Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, Bin Zhang, Xiong Wang, Yunfei Chu, and Junyang Lin. Qwen2.5-omni technical report. *arXiv preprint arXiv:2503.20215*, 2025.

[87] Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*, 2023.

[88] Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.

[89] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024.

[90] 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 I. Wang, and Ofir Press. Swe-bench multimodal: Do ai systems generalize to visual software domains? *arXiv preprint arXiv:2410.03859*, 2024.

[91] John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025.

[92] Qian Yang, Jin Xu, Wenrui Liu, Yunfei Chu, Ziyue Jiang, Xiaohuan Zhou, Yichong Leng, Yuanjun Lv, Zhou Zhao, Chang Zhou, and Jingren Zhou. Air-bench: Benchmarking large audio-language models via generative comprehension. *arXiv preprint arXiv:2402.07729*, 2024.

[93] Yudong Yang, Jimin Zhuang, Guangzhi Sun, Changli Tang, Yixuan Li, Peihan Li, Yifan Jiang, Wei Li, Zejun Ma, and Chao Zhang. Audio-centric video understanding benchmark without text shortcut. *arXiv preprint arXiv:2503.19951*, 2025.

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

[95] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[96] Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, et al. Claw-eval: Toward trustworthy evaluation of autonomous agents. *arXiv preprint arXiv:2604.06132*, 2026.

[97] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[98] Daoan Zhang, Wenlin Yao, Xiaoyang Wang, Yebowen Hu, Jiebo Luo, and Dong Yu. A versatile multimodal agent for multimedia content generation. *arXiv preprint arXiv:2601.03250*, 2026.

[99] Lu Zhang, Tiancheng Zhao, Heting Ying, Yibo Ma, and Kyusong Lee. Omagent: A multi-modal agent framework for complex video understanding with task divide-and-conquer. *arXiv preprint arXiv:2406.16620*, 2024.

[100] Xiaoyi Zhang, Zhaoyang Jia, Zongyu Guo, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Deep video discovery: Agentic search with tool use for long-form video understanding. *arXiv preprint arXiv:2505.18079*, 2025.

[101] Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, Haonan Fan, Kaibing Chen, Jiankang Chen, Haojie Ding, Kaiyu Tang, Zhang Zhang, Liang Wang, Fan Yang, Tingting Gao, and Guorui Zhou. Thyme: Think beyond images. *arXiv preprint arXiv:2508.11630*, 2025.

[102] Yiman Zhang, Ziheng Luo, Qiangyu Yan, Wei He, Borui Jiang, Xinghao Chen, and Kai Han. Omnieval: A benchmark for evaluating omni-modal models with visual, auditory, and textual inputs. *arXiv preprint arXiv:2506.20960*, 2025.

[103] Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. Autocoderover: Autonomous program improvement. *arXiv preprint arXiv:2404.05427*, 2024.

[104] Shifang Zhao, Yihan Hu, Ying Shan, Yunchao Wei, and Xiaodong Cun. Cutclaw: Agentic hours-long video editing via music synchronization. *arXiv preprint arXiv:2603.29664*, 2026.

[105] Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. Pyvision: Agentic vision with dynamic tooling. *arXiv preprint arXiv:2507.07998*, 2025.

[106] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

[107] Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. Mlvu: Benchmarking multi-task long video understanding. *arXiv preprint arXiv:2406.04264*, 2024.

[108] Zetong Zhou, Dongping Chen, Zixian Ma, Zhihan Hu, Mingyang Fu, Sinan Wang, Yao Wan, Zhou Zhao, and Ranjay Krishna. Reinforced visual perception with tools. *arXiv preprint arXiv:2509.01656*, 2025.

A Limitations

Our study has several limitations. First, due to computing-resource constraints, the CODE-O training recipe is only validated at the 9B and 27B scales (Qwen-3.5-9B and Qwen-3.6-27B), and we have not yet characterized how the recipe scales to frontier-size open-source models. Second, due to API credit constraints, several closed-source coding agents could not be evaluated on the full benchmark suite. In particular, Claude Opus 4.6 results on VideoZeroBench and Claude Opus 4.7 results across all four benchmarks are incomplete in the current draft, which means the comparison between coding agents and native omni-modal models is somewhat uneven on the closed-source side. Third, our scope is restricted to *offline* omni-modal tasks, where files are staged into a workspace ahead of time; conclusions do not transfer directly to streaming or interactive settings. Fourth, fixed-subset, single-run pass@1 results do not quantify run-to-run variance or subset-selection sensitivity. Fifth, final-call context length excludes external tool compute and does not establish end-to-end efficiency. Sixth, OmniGAIA shares some media sources with our training data despite no detected exact/core-question overlap (Appendix I.3). Finally, reward choices are supported by qualitative pilots rather than controlled sensitivity sweeps.

B Broader Impacts

Our findings identify a pathway for offline omni-modal understanding with a smaller final-call context footprint, showing that text+image coding agents with sandboxed tool use can match or surpass native omni-modal models on many video- and audio-centric tasks while using shorter final-call contexts. External media-processing compute remains necessary; the approach broadens interface choices for users without native omni-model APIs, while our open-sourced CODE-O recipe, OMNICODING dataset, and TERMINALBENCH-O benchmark further support reproducible research at the model–tool–environment interface. As with any tool-using agent that processes video and audio, downstream deployments should incorporate standard safeguards such as content provenance, access controls, and user consent; we note that our agents operate only on user-staged files within an isolated workspace and do not autonomously take real-world actions.

C Adapting Sandboxed Agents for Omni-Modal Tasks

We represent each omni-modal task as

$$x = (q, \mathcal{F}, \mathcal{Y}),$$

where q is a natural-language instruction, $\mathcal{F}$ is a set of input files, and $\mathcal{Y}$ is the set of acceptable final answers. We use an answer set rather than a single reference string so that aliases, option letters, numerical formats, and unit variants can be handled uniformly in both evaluation and reinforcement learning.

A native omni-modal model solves the task by packing the files into model context:

$$\hat{y}_{\text{omni}} = M_\theta(\text{Pack}(q, \mathcal{F}))\,,$$

where $\text{Pack}(\cdot)$ denotes model-specific media encoding, sampling, and serialization. In this formulation, video and audio are first-class model inputs.

In contrast, our sandboxed agent does not need to ingest all raw media as context. The files $\mathcal{F}$ are staged into an isolated workspace, and the agent is given the instruction and the corresponding file paths. It then interacts with the workspace through a terminal interface: it can inspect files, invoke tools such as `ffmpeg`, `ffprobe`, OCR, or ASR systems, write and execute scripts, create intermediate artifacts, and finally submit an answer.

A sandbox rollout is written as

$$\tau = (o_0, a_0, o_1, a_1, \ldots, o_T, a_T),$$

where o_t is the current observation, including command outputs and workspace feedback, and a_t is the agent's next message or terminal action. During a rollout, raw omni-modal files may be converted into task-relevant artifacts such as transcripts, sampled frames, OCR text, timestamps, tables, crops, or executable analysis scripts. The final prediction is parsed from the trajectory:

$$\hat{y} = \text{Ans}(\tau),$$

where $\mathrm{Ans}(\cdot)$ extracts the submitted answer, e.g., the content inside an `<answer>` tag when this format is required.

Thus the two settings differ mainly in where omni-modal processing happens:

$$\textbf{native omni:} \quad (q, \mathcal{F}) \xrightarrow{\mathrm{Pack}} \text{model context} \xrightarrow{M_\theta} \hat{y},$$

$$\textbf{sandboxed agent:} \quad (q, \mathcal{F}) \xrightarrow{\mathrm{stage}} \text{workspace} \xrightarrow{\text{terminal interaction}} \tau \xrightarrow{\mathrm{Ans}} \hat{y}.$$

This view treats raw video, audio, images, and documents as environment state rather than mandatory model context. The agent can therefore solve omni-modal tasks by adaptively reducing raw media into compact evidence that fits the native reasoning interface of a general-purpose coding model.

D Discussion and Future Works

Do non-native modalities really require native end-to-end perception? Our results suggest that non-native modalities do not always require native end-to-end perception. Even in the simplest Codex and Claude Code settings, coding agents solve a surprisingly large fraction of video-audio tasks, and on OmniGAIA they can even outperform strong native omnimodal baselines. The key is not that the base model suddenly becomes an audio-video perceiver, but that the sandbox lets it decompose raw multimedia into artifacts that match its native reasoning interface, such as transcripts, key frames, timestamps, OCR, metadata, and compact summaries. In this sense, multimodal processing becomes analogous to protocol-based software access in text-domain agents: the model delegates non-native operations to the environment and then reasons over the returned evidence. This perspective is consistent with the broader shift from static function calling to environment-mediated problem solving in tool-use and terminal-agent evaluation [76, 52, 8]. It also resonates with recent tool-augmented perception work, where explicit visual-tool policies and tool-derived perceptual signals both improve fine-grained multimodal reasoning [108, 77]. A natural next question is therefore not only whether a model is natively omni, but which capabilities should remain externalized, which can be compiled into reusable skills, and how agents should detect and recover from imperfect tool outputs. Our failure taxonomy and process-level benchmark point in this direction, suggesting that the central bottleneck is often tool selection, verification, and repair rather than the mere absence of raw multimodal encoders.

Shortcuts in current video-audio benchmarks. Are current video-audio benchmarks measuring what they claim to measure? Our findings suggest only partially. When a text+image coding agent can answer many ostensibly omnimodal tasks by selectively extracting transcripts, sparse frames, or metadata, benchmark accuracy no longer cleanly measures native end-to-end audio-video understanding. This does not make such benchmarks uninformative, but it does mean that they frequently conflate at least three capabilities: native perception, external evidence acquisition, and general knowledge-based reasoning. Our results are aligned with recent audits and diagnostic studies showing that many video or audio-visual benchmarks admit strong visual, textual, or temporal shortcuts [27, 44]. They are also consistent with benchmark-design efforts that explicitly enforce stronger audio dependence or reduce text-based shortcut channels [77]. Going forward, we believe benchmark design should standardize at least three controls: reporting text-only, text+image, and agentic baselines; auditing modality dependence with masking or filtering tests; and minimizing leakage from subtitles, transcripts, metadata, and external retrieval whenever the target claim is native multimodal understanding. Conversely, when the target is multimodal agentic competence, retrieval, tool orchestration, and evidence processing should be treated as first-class evaluation targets rather than accidental shortcuts. Making this distinction explicit would help separate backbone perception from system-level problem solving, which are both important but not interchangeable.

Can the same sandbox move from understanding to generation? We believe this is the most promising next step. Once an agent can inspect raw video, audio, and documents, compress them into structured evidence, and maintain task-relevant state over long horizons, it can in principle also plan edits, retrieve assets, invoke generative models, and validate outputs in a closed loop. Early multimedia-generation and agentic video-editing systems already point in this direction, showing that multimodal agents can coordinate text, images, video, music, and external tools under explicit planning and feedback loops [98, 104]. The next frontier is to connect that generative loop with persistent multimodal memory, so that an assistant can build private memos, recap videos, highlight reels, or even broadcast-ready drafts grounded not only in a single uploaded file but in a user's longer-term audiovisual history [46]. Doing so will require stronger memory management, preference

modeling, provenance tracking, and process-level verification, since generated content must remain faithful both to source media and to user intent. More broadly, this points to a different picture of future omni assistants: not a single monolithic model that passively consumes everything in one shot, but a sandboxed system that can inspect, remember, transform, and generate across modalities with the right mix of reusable skills and external tools. Under this view, offline omnimodal understanding is not the endpoint of this line of work, but the substrate for practical multimedia copilots.

Scope of the evidence. Our results establish the effectiveness of terminal-mediated evidence processing on the studied offline benchmarks, not superior intrinsic audio-video perception. Modality-reducible tasks still support a system-level comparison, while revealing that benchmark success need not require native video/audio encoders. We expect native perception to remain valuable for continuous motion and state tracking, tightly synchronized audio-visual grounding, and causal real-time streams where repeated seeking or transcription is impractical.

Relation to MMTB. Concurrent MMTB [23] evaluates 105 multimedia-file tasks and introduces Terminus-MM. Its controlled same-backbone comparisons show benefits from native audio/video access. These findings complement our answer-oriented evaluations, where ASR, OCR, sparse frames, and metadata often suffice; TerminalBench-O likewise exposes the difficulty of artifact-producing workflows. Backbone differences complicate cross-paper comparisons but do not explain away MMTB's controlled result. Together, the studies motivate hybrid agents combining native perception with terminal processing.

E Additional Experiment Results

E.1 Skills and Self-Evolution for Strong Many-modality Content Processing

Experiment Setups. Following recent work, we treat Skills as structured, inference-time packages of procedural knowledge that supply reusable workflows, instructions, and optionally executable resources to an agent [39, 88]. Unless explicitly stated otherwise, the main results in this paper are reported without Skills. We study three settings as follows:

- **Calibration-set self-iteration.** The agent is given a small calibration set and iteratively revises its Skills using only binary correctness feedback; the only supplied feedback is correct/incorrect, without reference solutions, reference trajectories, or process-level diagnoses.
- **Log-driven self-distillation.** A separate agent reviews raw execution logs, extracts recurring success and failure patterns, and writes Skills from these traces. This setting is inspired by recent log- and trace-driven harness optimization work such as Meta-Harness [34].
- **Human-in-the-loop Skills.** We provide manually written Skills that encode a strong human route or best-known workflow. This setting serves as an approximate upper bound on the value of explicit procedural knowledge, while also introducing human prior and route bias.

Results and Analysis. Figure 8 shows that explicit *Skills* consistently improve OMNIGAIA performance over the *No Skills* baseline across all difficulty levels. All skill-enhanced variants raise average accuracy, with gains remaining visible on the *High* split. The best setting improves average accuracy from 61.4% to 76.7%, suggesting that inference-time procedural knowledge can substantially strengthen omni-modal agent performance. Figure 7 provides a representative example, where Skills help the agent avoid an incorrect application of regional rules and reach the correct answer.

Both self-improving methods outperform the baseline, with *Log-driven self-distillation* achieving the strongest results: 76.7% average accuracy versus 73.0% for *Calibration-set self-iteration*, and the gap widens on harder problems (65.4% vs. 59.0% on the *High* split). The richer process information in execution logs may explain this difference, but the single-run comparison does not establish greater stability or generalization.

> ***Finding 4:*** Execution-trace-based skill refinement achieves the highest accuracy among the evaluated skill settings.

Table 6 reports the performance of different *Skills* settings on OMNIGAIA across the *Low*, *Medium*, and *High* difficulty splits, together with the overall average accuracy.

Case Study

Question. Watch the video showcasing a prominent surfing destination in British Columbia, Canada. Identify the specific town featured. Then, according to the Canadian Marine Mammal Regulations and 2024/2025 management measures for killer whale protection, what is the minimum mandatory vessel approach distance in meters required for killer whales in the waters immediately offshore of that town?
Labeled Answer. 200

Shared Prefix.

I will first use the video to identify the town shown in the clip. From the coastal scenery and visible landmarks, I will narrow the location down to `Tofino` before looking up the whale-protection rule.
`tool_call` Extract frames from the video and inspect visible location cues. `/tool_call`
`tool_response` The clip shows a prominent surfing destination in British Columbia, consistent with `Tofino`. `/tool_response`

`divergence` After identifying `Tofino`, the trajectories diverge on how they handle the Southern B.C. special protection measure: the failed branch applies the stricter `400 m` rule directly, while the successful branch checks whether that special corridor actually covers waters immediately offshore of Tofino. `/divergence`

Failed Trajectory

I have identified the town as `Tofino`. Next I will check official 2024/2025 killer-whale protection pages and use the applicable vessel-distance rule for this location.
`tool_call` `web_search:` official killer-whale measures near Tofino, 2024/2025 `/tool_call`
`tool_response` The search returns official pages mentioning a `400 m` approach distance in a protected Southern B.C. corridor. `/tool_response`
I will use that stricter regional measure as the answer for waters offshore of `Tofino`.
`final_answer` `Tofino, 400 meters` `/final_answer`

Successful Trajectory

I have identified the town as `Tofino`. Next I will check whether the stricter Southern B.C. killer-whale protection corridor actually applies to waters immediately offshore of this town, rather than assuming that the regional `400 m` rule governs here.
`tool_call` `web_search:` official killer-whale measures near Tofino, 2024/2025 `/tool_call`
`tool_response` The search returns official pages mentioning a `400 m` approach distance in a protected Southern B.C. corridor. `/tool_response`
I will then check the general regulation for killer whales in Pacific waters so I can compare the special regional measure against the baseline legal rule.
`tool_call` `web_search:` Marine Mammal Regulations killer whales Pacific waters 200 m `/tool_call`
`tool_response` The official Marine Mammal Regulations give a baseline minimum approach distance of `200 m` for killer whales in Pacific waters. `/tool_response`
I will verify that the Southern B.C. special corridor does not directly govern the waters immediately offshore of `Tofino`, so I will use the general `200 m` rule instead.
`final_answer` `Tofino, 200 metres` `/final_answer`

Key Difference. The failed trajectory directly applies the Southern B.C. special `400 m` measure to Tofino, whereas the successful trajectory explicitly checks whether that special corridor covers Tofino's offshore waters before falling back to the general `200 m` rule.

Figure 7: A representative OmniGAIA case comparing GPT-5.4 `high` without Skills and with Human-in-the-loop Skills. The no-Skills run answers incorrectly by directly applying the Southern B.C. special `400 m` rule to Tofino, whereas the Human-in-the-loop-Skills run answers correctly using `200 m` rule for the waters immediately offshore of Tofino.

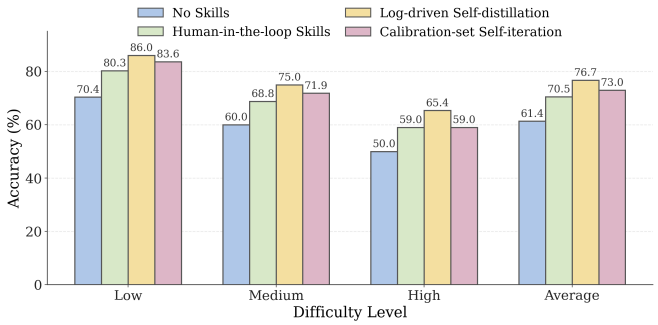

Figure 8: Performance of GPT-5.4 `high` on OmniGAIA under four settings.

Skill Setting	Low	Medium	High	Avg.
No Skills	70.4	60.0	50.0	61.4
Human-in-the-loop Skills	80.3	68.8	59.0	70.5
Log-driven Self-distillation	**86.0**	**75.0**	**65.4**	**76.7**
Calibration-set Self-iteration	83.6	71.9	59.0	73.0

Table 6: Performance of different *Skills* settings on OMNIGAIA across three difficulty levels. We report accuracy on the *Low*, *Medium*, and *High* splits, together with the overall average accuracy.

F Failure Mode Taxonomy and Process-level Trajectory Dataset

Here we provide detailed definitions of the six failure modes introduced in the main text.

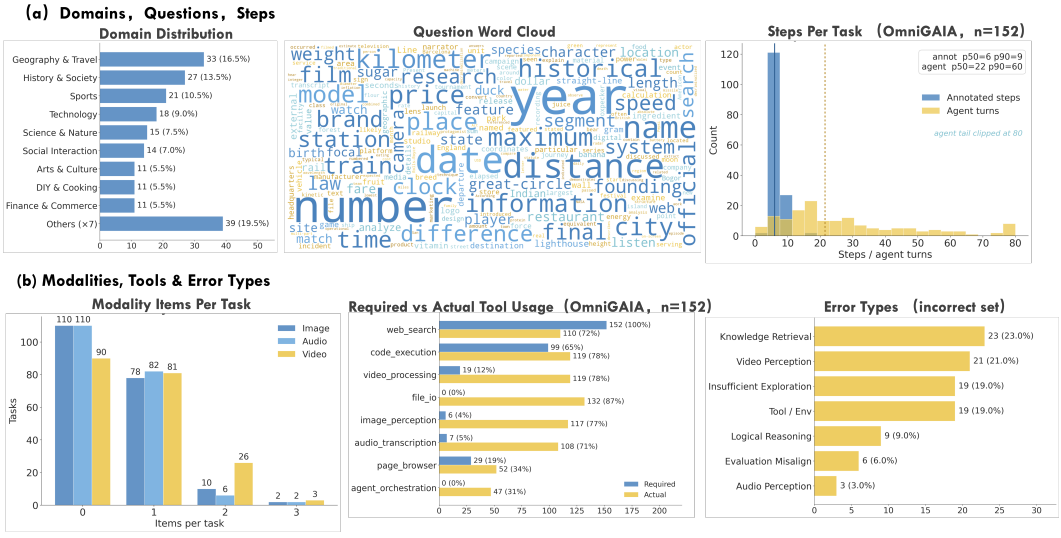

Figure 9: Overview of our process-level benchmark. **(a) Dataset Overview:** domain distribution across the union of OmniGAIA, LVOmniBench and SocialOmniBench (200 tasks); a word cloud of the question text; the distribution of annotated logical steps per task overlaid with the agent's actual turn count. **(b) Capability Analysis:** per-task counts of image / audio / video inputs; required-versus-actual tool-category coverage on the OmniGAIA subset (n=152) – "Required" is drawn from `required_external_tools` ∪ `sources`, *"Actual"* is recovered by argument-level classification of the agent trajectory; and the distribution of primary error types on the 100 incorrect trajectories we annotate.

- **Audio Perception and Extraction Error.** The agent mis-transcribes, mis-segments, or mis-attributes the audio signal itself (*e.g.*, ASR errors under noise or speaker confusion), so all downstream reasoning is conditioned on content that diverges from the true recording.
- **Video Perception and Extraction Error.** The visual stream is misread at the source via blank or off-key frame sampling, OCR mistakes, or inverted event ordering, leaving later stages with incorrect evidence.
- **Insufficient Exploration of Modal Content.** With multimodal inputs available, the agent answers from textual cues or priors, or terminates exploration before critical evidence is verified. The failure lies in the decision to stop probing rather than in perception quality.
- **Knowledge Retrieval and Factual Error.** Given correct perception, the agent links it to wrong external knowledge through poorly specified queries, unreliable sources, conflated entities, or fabricated facts when retrieval fails.
- **Logical Reasoning and Calculation Error.** Evidence is largely correct, but the derivation breaks down via wrong formulas (*e.g.*, planar vs. spherical distance), arithmetic mistakes, or dropped conditions in multi-step deductions.
- **Tool and Environment Infrastructure Failure.** The agent picks the right tool but is blocked by missing dependencies, sandbox permissions, or resource limits, forcing later steps to build on corrupted intermediates.

To scale failure mode analysis beyond the 200 human-annotated trajectories, we benchmark off-the-shelf LLMs as process-level judges and validate the chosen judge against human labels on this annotated set. Each model replays all 200 trajectories under a unified prompt, predicting a step reward $\in \{-1, 0, +1\}$ for every assistant step and, on the 100 incorrect trajectories, a primary error type from our taxonomy. We follow the AgentProcessBench [16] definition for process reward, with the rubric summarized below.

Table 7: LLMs as process-level judges on our 200-trajectory subset.

Judge model	Step	Primary	Budget
GPT-5.4-mini	0.588	0.657	3.17×
GPT-5.4	0.737	0.830	10.54×
GPT-5.5	**0.746**	**0.840**	32.49×
Qwen3.6-35B-A3B	0.605	0.813	1.68×
DeepSeek-V4-Flash	0.636	0.786	0.46×

Table 8: Step-type-specific criteria for the $\{-1, 0, +1\}$ process reward.

Step type	$+1$ **(effective)**	0 **(neutral)**	-1 **(harmful)**
Text-only step (planning, reasoning)	Useful plan or constraint with concrete task value.	Generic narration or low-value planning.	States false facts or commits to a bad premise.
Tool call	Tool choice is appropriate and materially useful.	Reasonable but low-yield, redundant, externally blocked, or based on incomplete information.	Clearly wrong tool, violates instructions, or repeats a failed pattern without strategy change.
Post-tool interpretation	Interpretation is supported by the tool output and advances the task.	Cautious but inconclusive.	Misreads the output or overclaims beyond the evidence.
Final answer / decisive claim	Supported by evidence already collected in the trajectory.	Cautious partial synthesis that does not materially help.	Guesses, hallucinates, or depends on an unresolved earlier error.

Step reward rubric. Only *assistant* messages are scored; *user* and *tool* messages serve as evidence. Each assistant step is judged using only information visible up to that step (no hindsight), and the three-level label is defined as follows:

- $+1$ **(correct and effective).** The step is factually correct given the available evidence and clearly advances the task, e.g., appropriate tool use, evidence-supported interpretation of a tool output, an effective decision, or an explicit correction of a prior mistake.

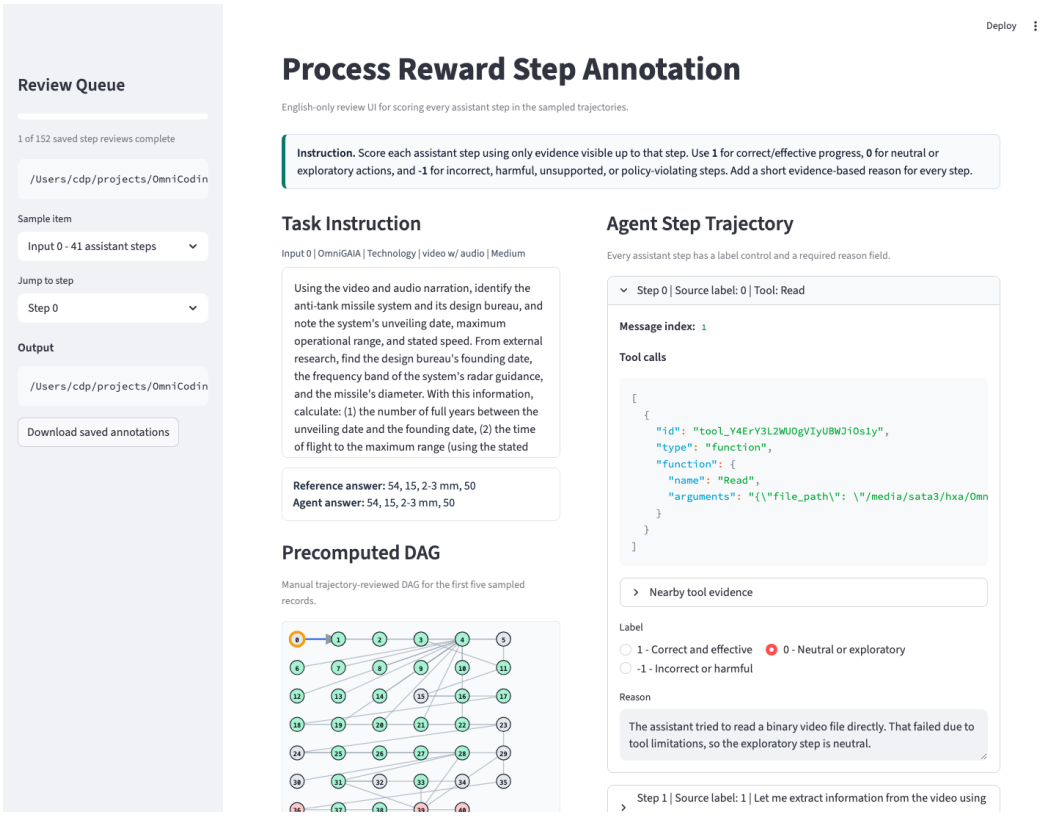

Figure 10: Annotation interface of our process-level dataset.

- 0 **(neutral or exploratory).** The step is reasonable but yields limited or unclear progress, e.g., sensible exploration, tentative planning, redundant recap, or a reasonable tool attempt that fails for external reasons such as a 404, timeout, or tool limitation.
- -1 **(incorrect or harmful).** The step is factually wrong, misreads or fabricates evidence, violates system or policy constraints, repeats a failed pattern without a meaningful strategy change, or otherwise moves the trajectory away from success.

We additionally apply **error propagation**: once a step is labeled -1, dependent later assistant steps remain -1 until the agent explicitly corrects the mistake or starts an independent subtask no longer relying on the bad premise. We provide our annotation interface in Figure 10.

Table 7 compares predictions against human labels via Step Acc. and Primary Acc. GPT-5.5 wins on both metrics; GPT-5.4 is second on step agreement; Qwen3.6-35B-A3B matches GPT-5.4 on error-type accuracy at much higher throughput; GPT-5.4-mini and DeepSeek-V4-Flash trail on step reward, indicating that smaller models are not yet adequate for fine-grained process supervision here. Based on these results, we adopt GPT-5.5 as the process-level judge for all large-scale failure mode analyses in the main paper.

G Full Related Works

G.1 Multimodal Models and Evaluation

Recent progress toward omni-modal assistants reflects a shift from modality-specific perception modules to unified models that can process speech, general audio, images, and video within a single system. On the model side, audio-language systems such as Qwen2-Audio [10] and video-centered models such as VideoLLaMA 2 [9] and video-SALMONN [67] established strong foundations for audio-visual understanding. More recent omni-modal systems, including VITA [19], OpenOmni [48], Baichuan-Omni [41], Moshi [12], and Qwen2.5-Omni [86], move toward unified and often real-time

interaction across speech and vision. These systems substantially improve the practicality of omni interaction, but they also make it harder to separate intrinsic model ability from gains due to decoding strategies, longer context, or external processing pipelines.

Benchmarking has evolved in parallel. Earlier suites mainly targeted either video understanding, such as MVBench [38], Video-MME [18], and MLVU [107], or audio / audio-visual understanding, such as AIR-Bench [92], MMAU [65], and AV-Odyssey [22]. While these benchmarks advanced multimodal evaluation, many still under-represent audio, weakly couple audio with video, or emphasize short-context perception more than long-horizon multimodal reasoning.

Recent work therefore moves toward explicitly omni-modal and long-form evaluation. Benchmarks such as OmniBench [42], OmniEval [102], OmniGAIA [40], LVOmniBench [71], SocialOmni [84], MMOU [21], and OmniPlay [5] target more realistic settings that require integrating visual, auditory, and textual evidence over longer horizons. At the same time, recent audits show that several audio-video benchmarks admit strong visual shortcuts [27], suggesting that progress in omni-modal evaluation depends not only on stronger models but also on shortcut-resistant benchmark design and explicit measurement of true modality dependence.

G.2 Agentic Tool-Use

The literature on tool-augmented language models treats external tools as a way to expand the action space of models rather than as simple add-ons. Foundational work such as ReAct [95] and Toolformer [66] showed that reasoning can be interleaved with actions and that models can learn when to invoke tools. Later efforts, including Gorilla [59], ToolBench [87], ToolLLM [61], BFCL [58], and τ-bench [94], scaled this paradigm to large tool collections and more interactive, stateful environments. Across these works, the central shift is from static capability elicitation to environment-mediated problem solving.

Executable code has emerged as one of the most expressive forms of tool use. CodeAct [74] argues that executable programs provide a general interface for acting in digital environments, allowing agents to manipulate files, call libraries, and compose multi-step behaviors more flexibly than fixed action vocabularies. In multimodal settings, Visual Sketchpad [25] and MLLM-Tool [73] show that models can externalize perception into sketches, programs, and structured tool calls. PyVision [105] and Thyme [101] further demonstrate that models can synthesize and execute Python-based image operations on demand, blurring the boundary between model capability and environment capability.

This perspective is increasingly visible in video and omni-modal reasoning. OmAgent [99], Omni-Agent [70], Agent-Omni [45], and LensWalk [37] all treat multimodal understanding as an active process of decomposition, search, and selective inspection rather than passive end-to-end perception. LongShOTBench [31] explicitly evaluates tool use and iterative refinement in long-video settings. More broadly, LLM-in-Sandbox argues that rich environments can elicit capabilities not apparent in static chat settings, reinforcing the view that multimodal competence may emerge from the interaction between a strong base model and an expressive tool substrate [8]. Together, these works motivate viewing multimodal intelligence as a property of a model–environment system rather than of a standalone backbone.

G.3 Terminal Coding Agents and Benchmarks

The most mature form of sandboxed agency has so far emerged in software engineering. SWE-bench [26] established the canonical issue-resolution setting by grounding evaluation in real GitHub issues and executable repository states. Systems such as SWE-agent [89], AutoCodeRover [103], Agentless [81], and OpenHands [75] explored different agent designs for this setting, ranging from search-heavy repair to lightweight execution-driven pipelines. This line of work reframes coding as long-horizon environment interaction rather than static code generation.

Benchmarks have expanded accordingly. SWE-bench Verified [55] improves reliability through a filtered subset, while SWE-bench Multimodal [90] extends evaluation to visually grounded software issues. Terminal-Bench [52] and LongCLI-Bench [17] move beyond repository patching toward broader command-line tasks and longer-horizon CLI interaction. Other recent benchmarks, including SetupBench [4], SWE-Bench Pro [13], and SEC-bench [33], further broaden evaluation to environment setup, longer-horizon issue resolution, and security workflows.

Table 9: Summary statistics for TERMINALBENCH-O.

Statistic	Value
Tasks	50
Agent-visible fixture files	3,013
audio/video/image/PDF/FITS files	1,293

The ecosystem has also expanded beyond benchmark construction into self-improvement and broader autonomy evaluation. SWE-smith [91] studies scalable data generation for software engineering agents, Live-SWE-agent [82] explores self-evolution during deployment, and SWE-Fuse [78] uses trajectory learning to improve agent performance. Beyond software-specific tasks, RE-Bench [79], HCAST [64], and TheAgentCompany [85] evaluate agents on longer-horizon work in realistic environments. At the same time, recent meta-analyses caution that benchmark realism, contamination, and task validity remain open concerns, and that high scores do not automatically imply broader engineering competence [49, 60]. These critiques are especially relevant when interpreting terminal-agent performance as evidence of general agentic ability.

H TERMINALBENCH-O Construction Details

TerminalBench-0 contains 50 curated terminal tasks under a shared workspace interface, each requiring the agent to create concrete artifacts under `/workspace/output` rather than return a short answer.

Raw data. We ground the suite in real-source data: public research datasets, public archives, public-domain scans, and direct-download media. The sources cover broadcast/web video, AMI and legal speech, action and egocentric video, industrial and scientific imagery, geospatial data, music stems, bioacoustics, score-audio pairs, FITS files, and historical comics. Synthetic content is reserved for controlled corruptions, decoys, overlays, and auxiliary labels needed to make evaluation precise.

Subset	All	SFT	RL	Type / Modality
By source				
Omnimodal-Agent-SFT-2K	2,110	1,413	697	open-ended, multimodal
OmniVideoBench	997	670	327	MCQ, video+audio
AVUTBenchmark	1,700	1,136	564	MCQ, video+audio
Video-MME-v2 selected	1,228	823	405	MCQ, video
By modality				
Video+audio	2,369	1,585	784	dual-stream media
Video	2,225	1,493	732	visual-temporal media
Audio+image	1,437	961	476	cross-modal media
Audio	4	3	1	audio-only media
Total	**6,035**	**4,042**	**1,993**	

Table 10: Statistics of OMNICODING.

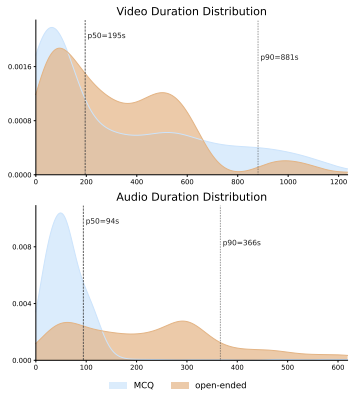

Figure 11: Duration distribution.

Preprocessing & Construction. Each source is normalized into a stable workspace representation before task packaging. We also prepare closed vocabularies, masks, catalogs, WCS headers, timestamp maps, and source-to-fixture mappings for downstream grading.

Step 1: Candidate mining. We first select source segments whose native structure supports a realistic operation task, such as eventful video intervals, speaker turns, action clips, score-aligned music excerpts. Candidate segments are filtered for duration, visibility, signal quality, annotation availability, and non-triviality before fixture generation.

Step 2: Task-specific transformation. We then impose an operation or creation objective on the selected source. Some tasks add controlled corruptions, hidden mappings, decoys, overlays, splice events, synchronization errors, or multi-source audio compositions. Others keep the media unchanged

but require structured deliverables such as ledgers, catalogs, action datasets, OMR alignments, reading-order tables, or redaction reports.

Step 3: Hidden reference synthesis. For each instance, we write hidden references that specify the answer geometry used by the grader. These references include event windows, speaker maps, transcripts, quote spans, crop paths, clean media, masks, stem targets, timestamp curves, object tracks and closed-vocabulary labels. They are withheld from the agent but used to score both final outputs and intermediate artifacts.

Step 4: Output-surface packaging. Finally, we expose each task as a small working directory rather than a question-only prompt. The visible files may include media, manifests, closed vocabularies, policy snippets, PDFs, CSVs, point clouds, FITS files, or auxiliary metadata, and the expected outputs are ordinary files such as MP4/WAV/PNG artifacts, CSV/JSON ledgers, subtitles,plots, and reports. This format keeps the benchmark native to terminal tools, `ffmpeg`, OCR/ASR, numerical Python, and file-system coordination.

Filtering. We apply filtering before and after packaging. Source candidates are removed if they are too short, ambiguous, silent, weakly annotated, or trivially solvable. Packaged tasks are then checked for complete schemas, aligned hidden references, active closed vocabularies, decoy coverage, and resistance to shortcuts such as pass-through media, empty manifests, constant labels, or copied templates.

Evaluation. Each task uses a dedicated grader with weighted dimensions and gating conditions. Deterministic checks cover temporal IoU, event recall, closed-vocabulary accuracy, WER/CER, speaker attribution, ASR preservation, signal metrics, SSIM/PSNR, mask IoU, timestamp error, catalog matching, photometric error, geospatial overlap, table consistency, and schema validity. For naturally open-form artifacts such as memos, overlays, or semantic summaries, some graders add constrained LLM/MLLM checklist axes. The final score therefore measures process-level completion: media quality, structured intermediate artifacts, format compliance, and cross-document consistency.

The reported aggregate combines deterministic and LLM/VLM-based assessments rather than representing a deterministic-only score. Reliance on proprietary judges and the absence of separate scores for these two assessment types remain limitations to reproducibility and interpretability.

I Implementation Details

I.1 Evaluation Scope, Permissions, and Resources

Comparison scope and permissions. We compare complete systems under benchmark-recommended interfaces, rather than matching architectures or total compute. Across all four benchmarks (OmniGAIA, SocialOmni, LVOmniBench, and VideoZeroBench), both coding agents and native omnimodal systems are permitted to use web search and benchmark-supported tools. General-purpose coding agents compose shell commands and self-authored programs, whereas OmniAgent and Agent-Omni use specialized multimodal controllers and tool interfaces; both categories are tool-using systems. Table 11 summarizes the evaluation policy separately from the restricted training harness.

Table 11: Benchmark-specific search policy and sandbox access. Search permissions are distinct from permission to download dependencies.

Setting	External search	Tools and software
OmniGAIA evaluation	Web search is permitted for coding agents and native omni systems.	Benchmark-supported tools; terminal agents may compose local software.
SocialOmni evaluation	Web search is permitted for coding agents and native omni systems.	Local media processing; dependency installation when permitted.
LVOmniBench / VideoZeroBench evaluation	Web search is permitted for coding agents and native omni systems.	Local media processing; dependency installation when permitted.
CODE-O training	Network primitives blocked.	`execute_commands`, `image_read`, and `task_complete`; paths outside the workspace blocked.

Sandbox interface and software. Agent-level interfaces expose terminal execution, image reading when supported, task completion, and benchmark-permitted search. These interfaces are distinct from software inside the shell: Ubuntu 22.04.3, Python 3.13.2, `ffmpeg`/`ffprobe`, Tesseract OCR, ImageMagick, OpenCV, and Librosa. Missing packages may be installed only when network access is permitted, within the isolated per-run environment rather than the root environment or other workspaces. This is a general-purpose multimedia environment, not an item-specific tool configuration.

Sample sizes and statistical scope. Every system uses the same fixed evaluation examples: 360 OmniGAIA tasks, 100 examples for each of SocialOmni Level-1 and Level-2, 100 LVOmniBench examples in total across its three difficulty levels, and 500 VideoZeroBench Level-3 questions. VideoZeroBench questions sharing a video are submitted together. LVOmniBench difficulty columns describe subsets of the same 100-example evaluation, not separate 100-example runs. The reported single-run pass@1 results do not estimate stochastic run-to-run variance or sensitivity to subset selection. Cross-benchmark coverage does not remove these limitations. Based on observed API usage, a full Gemini 3.1 Pro evaluation on the 1,014-example LVOmniBench was estimated to exceed USD 2,000; this is a cost estimate, not a measured full-benchmark expenditure.

Context-length accounting. Figure 3 reports the final-call input context of each rollout, containing the task prompt, retained prior model outputs, and text/image tool results returned to the model. A transcript or sampled frame contributes once represented in that context; the CPU/GPU work that produced it does not. This metric is neither cumulative input/output tokens across calls nor billed token usage, and caching does not make these quantities equivalent. The gray direct-MLLM reference is an estimate. We therefore interpret the figure as a comparison of context footprints, not total computational efficiency.

External compute and latency. Rollout workers used four CPU cores, 80 GB memory, and access to one GPU from a heterogeneous Slurm pool containing GTX 1080 Ti, RTX 2080 Ti, A4000, A5000, and A6000 nodes. Four processes shared one sandbox. GPU access allowed agents to install and invoke Whisper or other GPU-backed models. End-to-end latency was not recorded, and the heterogeneous, shared hardware does not support a controlled latency comparison. These resource allocations describe the environment, not measured per-task CPU/GPU consumption.

I.2 OmniCoding Demonstration Generation

We generate SFT demonstrations with GPT-5.4 in our local terminal-agent harness adapted from Terminus-KIRA. Tasks are first attempted at `low` or `medium` reasoning effort; failed cases are escalated to `high` or `xhigh`. We retain trajectories with valid tool-call formatting and a final answer matching the accepted answer set, then convert them to a model-agnostic chat format for SFT. This demonstration-generation procedure is distinct from the single-run pass@1 evaluation protocol.

I.3 Training–Evaluation Overlap Audit

We audited all 6,035 OmniCoding tasks against the evaluated benchmark units using normalized exact/core-question matching, TF-IDF retrieval, dense cosine similarity with Qwen3-Embedding-0.6B, identifiable media-source matching, and manual inspection. Table 12 reports the audit summary. The cosine threshold of 0.8 identifies nearest-neighbor candidates for inspection, not confirmed duplicates.

Table 12: Overlap audit against the evaluated examples. "No" means no overlap detected by the audit, not a guarantee against every form of contamination.

Benchmark	Exact/core question overlap	Identifiable media overlap	Cosine similarity ≥ 0.8
OmniGAIA	No	Yes	37/360
LVOmniBench	No	No	18/100
SocialOmni Level-1	No	No	99/100
SocialOmni Level-2	No	No	3/100
VideoZeroBench	No	No	7/500

Recurring counting, temporal-localization, and speaker-identification templates can yield high textual similarity across distinct media. SocialOmni Level-1's high candidate rate predominantly reflects such template similarity. No exact/core-question or identifiable media-source overlap was detected for SocialOmni, LVOmniBench, or VideoZeroBench.

OmniGAIA requires separate treatment. Its contribution comes from the officially released Omnimodal-Agent-SFT-2K source, following the training/evaluation protocol used for OmniAtlas. The source release contains 2,156 synthesized trajectories; OmniCoding retains 2,110 examples from this source (Table 10). The additional same-media candidate audit is summarized in Table 13.

Table 13: Additional audit of 221 OmniGAIA same-media candidates. Different questions or answers do not by themselves rule out benefits from shared source content.

Check	Result
Exact normalized question matches	0
Cosine similarity ≥ 0.8	30/221
Cosine similarity ≥ 0.9	2/221
Non-identical final answers	204/221

Same-source media can support questions about different segments, events, entities, or derived quantities, so source overlap is not equivalent to task duplication. Nevertheless, it remains a limitation for interpreting OmniGAIA gains; this audit does not establish performance under a media-disjoint split. The largest relative gains over direct MLLM inference occur on VideoZeroBench (+119.4% for 9B and +148.6% for 27B), where no direct overlap was detected, so OmniGAIA source overlap cannot account for gains across all four benchmarks.

I.4 CODE-O Reward Hyperparameters

Table 14 lists the reward weights and group-level coefficients used in all RL experiments. The values are shared across the 9B and 27B runs.

Table 14: Reward and group-shaping hyperparameters used in our RL training.

Symbol	Value	Description
Per-trajectory reward components		
w_{correct}	$+1.0$	Base reward for gated-correct trajectories.
$w_{\text{fmt}+}$	$+0.2$	Format bonus: wrong answer, clean `task_complete` exit, non-empty `<answer>`.
$w_{\text{fmt}-}$	-0.2	Format penalty: any other exit (timeout, step-limit, no answer, etc.).
w_{mod}	-0.3	Modality penalty when video/audio task is solved without a corresponding tool.
w_{tool}	0.5	Bad-tool weight; effective penalty is $-w_{\text{tool}} \cdot P_{\text{bad}} \in [-0.5, 0]$.
Group-level shaping		
G	8	Sibling rollouts per task (i.e., `n_samples_per_prompt`).
α	0.2	Length-penalty slope.
$f_{\min}$	0.5	Minimum length factor for correct samples.
ϵ_A	10^{-6}	Stabilizer in advantage normalization.
GSPO clipping		
ϵ_-	0.20	Lower clip threshold for $\rho_i(\theta)$.
ϵ_+	0.28	Upper clip threshold (DAPO clip-higher).

Reward-design observations. Preliminary grouped-rollout experiments motivated these choices. Without length regularization, rollouts grew progressively longer and training became unstable; overly strong length/step penalties caused premature termination, insufficient exploration, and groups with no correct trajectories. Modality penalties addressed answers produced without inspecting the corresponding media, while bad-tool penalties addressed persistent malformed or invalid calls after SFT. These observations motivate the design but are not controlled ablations: we did not conduct a full sweep of reward weights, GSPO clipping ranges, or alternative credit-assignment schemes.

Tool universes for modality detection. Modality engagement (used by w_{mod} and the correctness gate) is detected by substring-matching the bash text inside `execute_commands` arguments against the following tool universes:

- **Video:** `ffmpeg`, `ffprobe`, `ffplay`, `yt-dlp`, `mpv`, `vlc`, `VideoCapture`, `VideoWriter`, `VideoFileClip`, `moviepy`, `decord`, `VideoReader`, `imageio`, `mediainfo`.
- **Audio:** `whisper`, `faster_whisper`, `librosa`, `torchaudio`, `soundfile`, `sf.read`, `pydub`, `AudioSegment`, `wave.open`, `sox`, `demucs`, `spleeter`, `lame`.
- **Image:** `PIL.Image.open`, `Image.open`, `cv2.imread`, `cv2.imwrite`, `tesseract`, `pytesseract`, `convert`, `magick`, `identify`, `mogrify`, `easyocr`, `paddleocr`, `pdftotext`, `pdftoppm`, `exiftool`, `image_read`.

A task with `has_video=True` requires at least one match from the video or image set; a task with `has_audio=True` requires at least one match from the audio set. Image-only tasks are not modality-gated.

Bad-tool fail categories. The numerator of $P_{\text{bad}} = (n_{\text{unparseable}} + n_{\text{disallowed}} + n_{\text{escape}} + n_{\text{syntax-fail}})/(1 + n_{\text{tool}})$ aggregates four categories:

- **Unparseable:** malformed JSON in tool call, missing or non-dict `arguments`, or missing tool `name`.
- **Disallowed:** tool name outside the allowed set {`execute_commands`, `image_read`, `task_complete`}, or arguments matching network primitives (`curl`, `wget`, `ssh`, `scp`, `rsync`, `nc`, `telnet`, `http(s)://`).
- **Escape:** arguments matching gold-answer patterns (`ground_truth`, `answer_key`, `gold_answer`, `*.gold`, `all_final.jsonl`, `rl_train.jsonl`, `sft_train.jsonl`), or absolute paths resolving outside the workspace. Any escape event additionally triggers a hard zero on correctness.
- **Syntax-fail:** the tool reply contains `SyntaxError`, `invalid syntax`, `unexpected EOF`, `parse error`, `unexpected token`, `missing operand`, or `unterminated string`.

Unparseable tool calls short-circuit the other static checks, since they never actually execute in the sandbox.

Removed-trajectory criterion. A trajectory is marked *removed* (score zeroed, sample masked from the policy-gradient loss) iff the agent emits no `<answer>` *and* the exit reason is one of `timeout`, `step_limit`, or `context_overflow`. Group-level statistics (mean, std, dynamic filter) are computed over active (non-removed) samples only.

I.5 Prompts in Benchmark

Calibration protocol summaries and benchmark prompts are listed in Figures 12, 13, 14, 15, 16, 17.

Calibration-set self-iteration: feedback

The agent receives its previous reusable guide and binary correctness feedback on calibration attempts. The supplied feedback states only whether each attempt was correct or incorrect. It contains no reference answers, reference solution processes, expert diagnoses, or step-level annotations.
The agent uses these outcomes to revise generic strategies in its guide. The revised guide must not contain calibration questions, answers, named entities, or case-specific clues.

Figure 12: Calibration feedback protocol (summary).

Calibration-set self-iteration: guide update

The update produces a concise, reusable guide covering search strategy, tool selection, media processing, verification, and answer formatting.
The guide is revised using binary outcome feedback only; reference-process comparisons are not part of this setting.
The guide must preserve workspace restrictions and exclude benchmark-specific examples, copied questions, names, dates, locations, answer strings, and hidden hints tied to calibration cases.

Figure 13: Calibration guide-update protocol (summary).

Prompt for testing SocialOmni Leve1

You are evaluating a SocialOmni benchmark item in an isolated local workspace.
Work only inside the current workspace.
Your current working directory is the workspace root for this run.
Treat the current working directory tree as the only allowed workspace root.
Do not access, read, inspect, rely on, or search for files, directories, or other local content outside the current workspace.
Never open absolute paths or parent-directory paths outside the workspace, and never use symlinks, environment discovery, shell expansion, or search commands to reach files beyond the workspace root.
Treat any host file, cache, home-directory content, sibling directory, mounted dataset source, or path outside this workspace as strictly forbidden, even if it appears readable from the shell.
Use only the staged files listed below; do not assume any hidden benchmark files exist.
You may install dependencies without sudo when necessary, but keep them inside the current workspace.
User-level install prefixes and caches are redirected into the workspace for this run.
Available staged files:
`[LOCAL_VIDEO_PATH]`
Network access may be available for shell commands in this run.
You may use the network only when it materially helps process the current sample, including downloading tools or packages into the current workspace only.
Do not use the network to search for benchmark answers, leaked annotations, dataset-specific solutions, or existing evaluation outputs.
`[SYSTEM]`
`[LEVEL1_SYSTEM_PROMPT]`
`[LOCAL_VIDEO]`
Analyze the local video file at: `[LOCAL_VIDEO_PATH]`
`[ASR]`
`[ASR_CONTENT_IF_ENABLED]`
Options:
`A. ...`
`B. ...`
`C. ...`
`D. ...`
`[LEVEL1_USER_PROMPT_BASE]`
`[ANSWER_FORMAT]`
Output requirements:
Return ONLY the single option letter (A, B, C, D).
Do not repeat the option text.
Do not provide any explanation, reasoning, analysis, punctuation, markdown, or extra words.
Your entire response must be exactly one of: `A/B/C/D`.
`[QUESTION_TEXT]`

Figure 14: Prompt for testing SocialOmni Level1.

Prompt for testing SocialOmni Level2

You are evaluating a SocialOmni benchmark item in an isolated local workspace.
Work only inside the current workspace.
Your current working directory is the workspace root for this run.
Treat the current working directory tree as the only allowed workspace root.
Do not access, read, inspect, rely on, or search for files, directories, or other local content outside the current workspace.
Never open absolute paths or parent-directory paths outside the workspace, and never use symlinks, environment discovery, shell expansion, or search commands to reach files beyond the workspace root.
Treat any host file, cache, home-directory content, sibling directory, mounted dataset source, or path outside this workspace as strictly forbidden, even if it appears readable from the shell.
Use only the staged files listed below; do not assume any hidden benchmark files exist.
You may install dependencies without sudo when necessary, but keep them inside the current workspace.
User-level install prefixes and caches are redirected into the workspace for this run.
Available staged files:
`[LOCAL_PREFIX_VIDEO_PATH]`
Network access may be available for shell commands in this run.
You may use the network only when it materially helps process the current sample, including downloading tools or packages into the current workspace only.
Do not use the network to search for benchmark answers, leaked annotations, dataset-specific solutions, or existing evaluation outputs.
`[SYSTEM]`
`[LEVEL2_SYSTEM_PROMPT]`
`[LOCAL_VIDEO]`
Analyze the local prefix video file at: `[LOCAL_PREFIX_VIDEO_PATH]`
`[ASR]`
`[FULL_ASR_IF_ENABLED]`
`[LEVEL2_USER_PROMPT_BASE]`
Task 1 (Q1):
Question: `[Q1_QUESTION_TEXT]`
Options:
`A. [OPTION_A]`
`B. [OPTION_B]`
Task 2 (Q2):
Question: `[Q2_QUESTION_TEXT]`
If your Q1 answer is A, provide a concise natural utterance for Q2.
If your Q1 answer is B, set Q2 to N/A.
Output requirements:
Do not provide explanation, reasoning, analysis, markdown, or extra words.
Return exactly in the following format:
`<q1>A_or_B</q1>`
`<q2>utterance_or_NA</q2>`

Figure 15: Prompt for testing SocialOmni Level2.

Prompt for testing LVOmniBench

You are evaluating a long audio-video benchmark item in an isolated local workspace.
Work only inside the current workspace.
Do not access, read, inspect, rely on, or search for any files, directories, or other content outside the current workspace.
Never open absolute paths or parent-directory paths outside the workspace, and never use symlinks, environment discovery, shell expansion, or search commands to reach files beyond the workspace root.
Treat any host file, cache, home-directory content, sibling directory, mounted dataset source, or path outside the current workspace as strictly forbidden, even if it appears readable from the shell.
Use the staged video file(s) listed below and any local tools available in the workspace.
Do not assume access to any original benchmark directories outside this workspace.
Network access may be available for shell commands in this run.
You may use the network only when it materially helps process the current sample, including downloading tools or packages into the current workspace only.
Do not use the network to search for benchmark answers, leaked annotations, dataset-specific solutions, or existing evaluation outputs.
Available staged files:
`[STAGED_VIDEO_FILES]`
Use the benchmark's official answering format below.
Question: `[QUESTION_TEXT]`
Options:
`A. ...`
`B. ...`
`C. ...`
`D. ...`
Select the best answer from the options above. Directly provide the letter representing your choice (A/B/C/D) and nothing else. Do not include the full text of the option, do not provide any explanation.

Figure 16: Prompt for testing LVOmniBench.

Prompt for testing VideoZeroBench

You are evaluating VideoZeroBench questions in a sandboxed local workspace.
The model can access only the staged video file and files it creates during the current run. Do not access, read, inspect, rely on, or search for files outside the sandbox, including benchmark annotations, gold answers, evidence windows, evidence boxes, previous evaluation outputs, cached results, or any other files that may reveal the answer.
Network access is allowed only for general-purpose assistance that does not reveal benchmark-specific information. Do not use the network to search for the benchmark name, dataset, question text, video id, video filename, leaked annotations, gold answers, existing evaluation outputs, or papers/pages that evaluate or discuss the same sample. Do not retrieve any content that directly or indirectly reveals the answer.
If video inspection is needed, use information derived only from the provided staged video and the question. Do not use external sources or prior knowledge to retrieve or infer benchmark-specific answers.
All questions below refer to the same video and must be answered in a single response.
Available staged video:
- `{VIDEO_FILENAME}.mp4`
Video id / filename: `{VIDEO_FILENAME}.mp4`
Requested evaluation level: Level-3.
Return ONLY valid JSON.
Return a JSON array with exactly one object per `question_id` shown below.
Do not include markdown fences, explanations, extra keys, chain-of-thought, or commentary.
The entire final response must be the JSON array itself.
Your first output character must be `[` and your final output character must be `]`.
If you cannot determine an answer, use an empty string.
Required output object schema:

```
{
   "question_id":  "string",
   "level3_answer":  "string"
}
```

Question bundle:

```
[
   {
      "question_id":  "{QUESTION_ID_1}",
      "question":  "{QUESTION_TEXT_1}"
   },
   {
      "question_id":  "{QUESTION_ID_2}",
      "question":  "{QUESTION_TEXT_2}"
   }
]
```

Level-3 QA rule:
- This is a question answering task without benchmark-provided temporal or spatial evidence.
- Answer according to the video content and the question.
- Return only the final answer text in `level3_answer`.

Figure 17: Prompt for testing VideoZeroBench, where all questions for the same video are answered together in a single response.

I.6 Experimental Configuration Details

This subsection summarizes the evaluation configurations used in our experiments for each agent and omnimodal model setting, including controller settings, model assignments, modality-processing rules, and benchmark-specific runtime defaults.

I.6.1 Agent-Omni

Controller and graph configuration. Agent-Omni is evaluated with a fixed multi-agent graph consisting of modality summarization, master reasoning, modality-specific execution, and master decision. The default reasoning budget is two rounds, i.e., `max_round_num=2`. The global retry budget is `retry_times=10`.

Model assignment. The committed benchmark configuration uses the following model split. **We use even stronger models compared to the configuration in their paper**:

- orchestrate agent: `openai/gpt-5.4`;
- text agent: `openai/gpt-5.4`;
- image agent: `openai/gpt-5.4`;
- video agent: local `Qwen/Qwen2.5-Omni-7B`;
- audio agent: local `Qwen/Qwen2.5-Omni-7B`.

Generation settings. The orchestrate, text, and image agents use `max_tokens=4096` and `temperature=0`. The video agent uses `max_tokens=2048`, `temperature=0`, `max_image_input=12`, `max_frames_num=448`, and `image_size=448` $\times$ `448`. The audio agent uses `max_tokens=4096` and `temperature=0`.

Modality processing. For video benchmarks, the audio track is extracted from each video and provided jointly with the original video input, so the effective input to the graph is audio-video rather than video only. All four modality branches remain part of the graph, while the master reasoning module determines which agents receive non-empty questions.

Output normalization. For multiple-choice evaluation, the graph output is not used directly. Instead, a separate selector or finalizer model is applied after graph execution. By default, this selector/finalizer uses the same model as the master agent.

LVOmniBench configuration. The default runtime configuration is:

- `concurrent_limit=1`,
- `max_round_num=2`,
- `request_timeout=600`,
- `ffmpeg_timeout=300`,
- `empty_answer_retry_rounds=3`.

SocialOmni configuration. The default runtime configuration is:

- `max_concurrency=1`,
- `max_round_num=2`,
- `request_timeout=600`,
- `ffmpeg_timeout=300`,
- `empty_answer_retry_rounds=3`.

For Level-2 evaluation, a prefix video is first cropped up to the target timestamp. The model predicts Q1 before Q2, and Q2 is executed only when the gold Q1 label is `A` and the Q1 prediction is correct. The default judge model for Level-2 is `gpt-5.4`.

VideoZeroBench configuration. The default runtime configuration is:

- `level=level3`,
- `group_by_video=True`,
- `concurrent_limit=1`,
- `max_round_num=2`,

- request_timeout=600,
- ffmpeg_timeout=300,
- empty_answer_retry_rounds=2.

When grouped by video, all questions attached to the same video are answered from a single graph run.

I.6.2 OmniAgent

Controller configuration. OmniAgent is evaluated as a single tool-using controller. The default controller budget is max_iterations=30. The controller model uses temperature=1, and the default reasoning effort is set to high when supported by the serving API.

Model assignment. The default released configuration is listed below. **We use even stronger models compared to the configuration in their paper**:

- controller model: openai/gpt-5.4;
- video tool model: local Qwen/Qwen2.5-Omni-7B;
- audio tool model: openai/gpt-4o-audio-preview.

Tool set. The default core tools are:

- audio_global_caption,
- audio_qa,
- audio_ASR,
- Audio_EventList,
- Audio_EventLocation,
- video_global_qa,
- video_clip_qa,
- video_metadata.

External web and code tools are implemented but disabled by default in the benchmark runs.

Video sampling configuration. The default multimodal settings are:

- VIDEO_INPUT_MODE=frames,
- VIDEO_TARGET_FPS=0.5,
- VIDEO_CLIP_TARGET_FPS=2.0,
- VIDEO_MAX_FRAMES=24,
- VIDEO_CLIP_MAX_FRAMES=40,
- VIDEO_MAX_EDGE=1024,
- MULTIMODAL_MAX_TOKENS=4096.

For oversized or empty multimodal responses, the video tool falls back to smaller payloads using 16/12/8 frames and 768/640/512 maximum edge size.

Audio processing. The audio tool can consume audio files directly and can also extract audio from video inputs. Audio event localization and timeline generation are performed with chunk-based prompting followed by a merge step.

LVOmniBench configuration. The default runtime configuration is:

- concurrent_limit=1,
- max_iterations=30,
- item_timeout=5400,
- empty_answer_retry_rounds=3,
- external tools disabled by default.

SocialOmni configuration. The default runtime configuration is:

- concurrent_limit=2,

- `max_iterations=30`,
- `item_timeout=5400`,
- `empty_answer_retry_rounds=3`,
- external tools disabled by default.

For Level-2 evaluation, a prefix video is cached up to the target timestamp. Q1 is executed before Q2, and Q2 is run only when the gold Q1 label is `A` and the Q1 prediction is correct. When available, the benchmark ASR field is injected into the prompt. The Level-2 judge uses the evaluation endpoint specified by the evaluation model settings, which default to the controller-model endpoint unless overridden.

VideoZeroBench configuration. The default runtime configuration is:

- `level=level3`,
- `request_unit=video`,
- `concurrent_limit=1`,
- `max_iterations=30`,
- `item_timeout=5400`,
- `empty_answer_retry_rounds=3`,
- external tools disabled by default.

Under `request_unit=video`, all questions corresponding to the same video are answered in one batched response, and the final answers are parsed from a JSON answer map.

J Licenses and Use Terms of Existing Assets

We list below all third-party benchmarks, datasets, models, and software libraries used in this work, together with their licenses to the best of our knowledge. All assets are used in a manner consistent with their stated terms; we used them only for non-commercial academic research.

Benchmarks for Evaluation

- **OmniGAIA** [40]: an open-ended, tool-mediated factual and multi-hop omni-modal reasoning benchmark. Released under the MIT License for the code and CC-BY-4.0 for the data annotations; the underlying media inherits the licenses of its original public sources.
- **SocialOmni** [84]: an audio-visual social interaction benchmark for omni models. Released under the MIT License for the code; data is provided for non-commercial research use, with the underlying videos governed by their original platform licenses.
- **LVOmniBench** [71]: a long audio-video understanding benchmark. Released under the MIT License for the code and Apache-2.0 for the released annotations; underlying videos retain their original licenses.
- **VideoZeroBench** [50]: a video MLLM benchmark with spatio-temporal evidence verification. Released under the MIT License for the code; data annotations are provided under CC-BY-4.0 for academic use.

Datasets Used in OMNICODING

- **OmniGAIA-SFT-2K** [40]: the supervised-trajectory subset distributed with OmniGAIA. Released alongside OmniGAIA under the same MIT/CC-BY-4.0 terms.
- **OmniVideoBench** [35]: an audio-visual understanding benchmark for omni MLLMs. Released under the MIT License; underlying videos retain their original public licenses.
- **AVUTBenchmark** [93]: an audio-centric video understanding benchmark designed to remove text shortcuts. Released under the MIT License for code and CC-BY-4.0 for annotations.
- **Video-MME-v2** (audio-visual / temporal subset) [20]: a comprehensive video understanding benchmark. Released under the MIT License (code) and CC-BY-NC-4.0 (data) for academic, non-commercial use.

Source Material for TERMINALBENCH-O

All raw media in TERMINALBENCH-O is drawn from authentic public sources:

- **Internet Archive**: content used under the licenses indicated on each Internet Archive item page (typically Public Domain, CC-BY, or CC0).
- **HuggingFace public datasets**: each used under the dataset-specific license declared on its HuggingFace card (the specific licenses are listed in our benchmark release card).
- **Public web videos**: incorporated only when the source platform's terms permit research use, with attribution preserved in our task metadata.

Models

- **Qwen-3.5-9B** and **Qwen-3.6-27B** [62]: open-weight base models we fine-tune. Released by Alibaba under the Tongyi Qianwen / Apache-2.0-style license, which permits research and commercial use subject to the model's published terms.
- **Whisper (large-v3)**: ASR model used inside agent rollouts. Released by OpenAI under the MIT License.
- **Closed-source models** (GPT-5.4, Claude Opus 4.6 / Sonnet 4.6, Gemini 3.1 Pro, Gemini 3 Flash, etc.) are accessed exclusively through their official APIs in compliance with each provider's terms of service.

Software and Libraries

- **ffmpeg / ffprobe**: LGPL-2.1+ (with GPL components in some builds).
- **Tesseract OCR**: Apache-2.0.
- **OpenCV**: Apache-2.0.
- **Librosa**: ISC License.
- **ImageMagick**: ImageMagick License (Apache-2.0-compatible).
- **yt-dlp**: Unlicense (public domain).
- **Python** and standard scientific stack (NumPy, SciPy, etc.): each under their respective permissive licenses (BSD/PSF/MIT).

J.1 Omni-terminal Harness

This appendix details our terminal-agent harness, adapted from Terminus-KIRA [30] and designed around a single operation: turning raw multimodal files into small, checkable evidence. Each episode begins with an isolated workspace containing the question and the staged media files. Rather than packing media into the model context, files remain on disk, and the agent must decide how to inspect them: extracting audio, sampling frames, running OCR or ASR, reading metadata, writing scripts, and verifying intermediate results before answering. The agent operates through a small action space (`execute_commands`, `image_read`, `task_complete`) and expresses higher-level media operations through code and command-line tools (`ffmpeg`/`ffprobe`, ASR, OCR, Python, OpenCV, NumPy, and self-authored scripts) rather than benchmark-specific APIs. Beyond execution, the harness is centered on trajectory collection: every tool call, terminal observation, image observation, final answer, exit reason, and reward field is serialized so the same rollout can be reused for SFT filtering, RL replay, and failure analysis.

Concretely, the harness has four properties that are important for training:

- **Filesystem-grounded media processing.** Raw video, audio, images, and documents are workspace state. The agent learns to transform them into transcripts, key frames, OCR text, timestamps, tables, and other compact evidence.
- **Persistent terminal interaction.** Commands run in a stateful bash session with marker-based completion, bounded observations, timeout recovery, and support for scripts written during the episode.
- **Replay-faithful multimodal observations.** `image_read` inserts the actual image into the main conversation, rather than replacing it with a caption from a separate model. The saved rollout therefore matches the observation distribution used for multimodal SFT/RL.

Tool	Function
`execute_commands`	Runs one shell command in a persistent bash session. The schema is flat: the call contains `analysis`, `plan`, and `keystrokes`. Multi-step work is expressed with shell syntax, scripts, or heredocs.
`image_read`	Loads an image file as a native multimodal observation for the main agent. This is used for frames, crops, figures, screenshots, OCR targets, and visual verification.
`task_complete`	Ends the rollout after the model has emitted the final answer wrapper. If the wrapper is missing, the harness issues one format reminder before scoring the run as empty.

Table 15: Model-visible tools in the OMNICODING harness.

- **Qwen-compatible rollout collection.** The tool schema is flat, malformed tool-call blocks are conservatively recovered, and requests are routed through sticky SGLang endpoints to preserve prefix-cache locality during large parallel rollouts.

Episode state. Each task starts from an isolated workspace

$$s_0 = (q, W_0, U_0),$$

where q is the task instruction, W_0 contains the staged input files, and U_0 contains the available command-line tools and Python libraries. The model does not receive the raw video or audio stream as context. Instead, it interacts with files through the terminal. During a rollout, the workspace may accumulate derived artifacts such as extracted frames, audio clips, ASR transcripts, OCR outputs, metadata files, temporary scripts, and numerical tables. The final prediction is accepted only from the benchmark-normalized `<answer>...</answer>` wrapper.

Tool interface. The model sees three tools. The small tool set is deliberate: the terminal is the compositional interface, and the agent decides how to combine existing utilities with code it writes during the episode.

Agent loop. At each step, the harness sends the current conversation and tool schema to the model. If the model calls `execute_commands`, the command is executed in the persistent shell and the bounded output is returned as a tool observation. If it calls `image_read`, the harness first records a short tool acknowledgement and then appends the image itself as a multimodal observation in the main conversation. If it calls `task_complete`, the harness checks whether a wrapped answer already appears in the trajectory. The loop stops on successful completion, step limit, no-tool-call exhaustion, or runtime error.

Shell runtime. The shell is a persistent bash subprocess. This preserves state across turns, including the current directory, environment variables, installed packages, and files created by the agent. Command completion is detected by appending a hidden marker after each command and polling until the marker appears. Terminal observations are capped with head–tail truncation, which keeps both the beginning of long outputs and the end where errors or final answers often appear. When a command times out, the harness terminates the direct child processes while keeping the shell alive. If the shell itself dies, it is restarted and the next observation warns the model that shell-local state has been lost.

Native multimodal replay. The main change from a standard terminal harness is the native `image_read` path. A caption-only image reader is acceptable for evaluation, but it is not acceptable for multimodal policy training: the saved trajectory would contain text produced by a separate model rather than the pixels observed during rollout. Our harness instead stores image observations as multimodal content blocks in the main conversation. For Qwen-style chat templates, these messages are folded into the provider-specific tool-response format only at send time. The saved artifact remains provider-neutral and can be replayed for SFT or RL without changing the observation sequence.

Qwen serving and tool-call robustness. Open-weight models served through SGLang can emit partially malformed tool-call blocks, especially when shell commands contain quotes, heredocs, or escape characters. We therefore use a flat `execute_commands` schema and a conservative recovery layer. If the backend returns no structured tool call but the assistant text contains a recognizable `<tool_call>` block, the harness reconstructs the call only when the function name is in the declared tool list and the arguments can be parsed without inventing content. Recovered calls are marked in the trajectory, so downstream data filtering can keep or drop them explicitly.

Table 16: Trajectory fields saved by the harness.

Artifact field	Purpose
Messages	Replayable conversation with tool calls and observations.
Workspace outputs	Derived files such as frames, transcripts, OCR text, scripts, and tables.
Image references	Native multimodal payloads used by `image_read`.
Answer field	Extracted text inside `<answer>...</answer>`.
Exit reason	Completion, step limit, no-tool-call exit, or runtime error.
Reward metadata	Automatic verification result and benchmark-specific scoring fields.

Trajectory artifacts. Each rollout stores the full message sequence, parsed tool calls, shell outputs, image-observation references, final extracted answer, exit reason, and evaluation metadata. The same artifact supports three downstream uses. First, successful trajectories can be filtered into SFT examples. Second, RL training can replay the same observation distribution seen during rollout. Third, failures can be inspected at the process level: whether the agent sampled the wrong frames, trusted a bad ASR transcript, stopped too early, chose the wrong external fact, or made a calculation error.

Prompt organization. The prompt is factored into reusable pieces. A static system prompt defines the tool contract. A shared workspace prefix defines sandbox, network, and answer-format rules. Benchmark-specific prompt bodies supply the actual question, options, files, and output schema. Runtime reminders are used only when the model stops without a tool call or calls `task_complete` before producing a wrapped answer. The exact prompt templates are shown in Figures 18–23.

Common workspace and answer-format prefix

```
You are evaluating a benchmark item in an isolated local workspace.

Workspace rules:
- Work only inside the current workspace.
- Treat the current directory tree as the only allowed workspace root.
- Do not access, read, inspect, rely on, or search for files outside it.
- Do not open absolute paths or parent-directory paths outside the workspace.
- Do not use symlinks, shell expansion, environment discovery, or search
  commands to reach host files, caches, sibling directories, mounted datasets,
  benchmark annotations, gold answers, evidence windows, or prior outputs.
- Use only the staged files listed below and files you create during this run.

Network rules:
- Network access may be available.
- Use it only when it materially helps process the current sample, such as
  installing a dependency into the workspace or checking general public facts.
- Do not search for benchmark names, question text, video ids, file names,
  leaked annotations, gold answers, existing evaluation outputs, or pages that
  directly reveal the answer to this sample.

Available staged files:
[STAGED_FILES]

Question:
[QUESTION_TEXT]

Final-answer protocol:
- First determine the benchmark-specific answer text.
- The inner answer must follow the benchmark's official format.
- Emit the final prediction exactly as:
  <answer>FINAL_ANSWER</answer>
- Do not include reasoning, markdown, or extra text inside the answer wrapper.
```

Figure 18: Shared workspace, leakage-prevention, network-use, and final-answer rules.

Runtime reminder prompts

```
[No-tool-call reminder]

Please continue solving the task.
If you need more evidence, run a tool.
If you have enough evidence, emit the final answer using the exact answer
wrapper required by the task:
<answer>FINAL_ANSWER</answer>

Unwrapped prose is treated as no answer.
Do not stop, do not call task_complete, and do not respond with plain text
unless it is the wrapped final answer.


[Missing-answer reminder after task_complete]

task_complete was called, but no <answer>...</answer> wrapper is present in
the trajectory. The grader extracts the prediction only from that wrapper.

In your next assistant turn, emit exactly:
<answer>YOUR_ANSWER</answer>

After that answer appears in the trajectory, call task_complete in the next
turn. Prefer a plain assistant answer over shell echo, because shell quoting
can corrupt answers containing apostrophes, angle brackets, or dollar signs.
```

Figure 19: Runtime reminders used when the model stops without a tool call or calls `task_complete` before producing a wrapped answer.

K Case Study

K.1 Representative Failure Case per Error Category

To make the failure taxonomy concrete, we present one representative case from each category, drawn from our 100 manually annotated incorrect trajectories. Each case shows the question, ground-truth answer, the agent's final prediction, and a brief analysis of where the trajectory went wrong.

SocialOmni prompt bodies

```
[SocialOmni Level 1]

Analyze the local video file:
[LOCAL_VIDEO_PATH]

Optional ASR transcript, if provided by the benchmark:
[ASR_CONTENT]

Options:
A. [OPTION_A]
B. [OPTION_B]
C. [OPTION_C]
D. [OPTION_D]

Question:
[QUESTION_TEXT]

Output:
Return the single best option letter as the inner answer.
The final response must be:
<answer>A_or_B_or_C_or_D</answer>


[SocialOmni Level 2]

Analyze the local prefix video file:
[LOCAL_PREFIX_VIDEO_PATH]

Optional full ASR transcript, if provided by the benchmark:
[FULL_ASR_CONTENT]

Task 1 (Q1):
Question: [Q1_QUESTION_TEXT]
Options:
A. [OPTION_A]
B. [OPTION_B]

Task 2 (Q2):
Question: [Q2_QUESTION_TEXT]

If Q1 is A, provide a concise natural utterance for Q2.
If Q1 is B, set Q2 to N/A.

Output:
Return exactly the following XML as the inner answer:
<q1>A_or_B</q1>
<q2>utterance_or_NA</q2>

The final response must be:
<answer><q1>A_or_B</q1><q2>utterance_or_NA</q2></answer>
```

Figure 20: Benchmark-specific prompt bodies for SocialOmni Level 1 and Level 2.

LVOmniBench prompt body

```
You are evaluating a long audio-video benchmark item.

Available staged video file(s):
[STAGED_VIDEO_FILES]

Question:
[QUESTION_TEXT]

Options:
A. [OPTION_A]
B. [OPTION_B]
C. [OPTION_C]
D. [OPTION_D]

Instructions:
- Use the staged video/audio content as needed.
- Extract frames, audio, transcripts, metadata, or clips when useful.
- Select the best answer from the options above.
- The inner answer must be only the option letter.

Final response:
<answer>A_or_B_or_C_or_D</answer>
```

Figure 21: Benchmark-specific prompt body for LVOmniBench.

VideoZeroBench grouped-video prompt body

```
You are evaluating VideoZeroBench questions in a sandboxed local workspace.

All questions below refer to the same staged video and must be answered in a
single response.

Available staged video:
[VIDEO_FILENAME].mp4

Requested evaluation level:
Level-3

Question bundle:
[
  {
    "question_id": "[QUESTION_ID_1]",
    "question": "[QUESTION_TEXT_1]"
  },
  {
    "question_id": "[QUESTION_ID_2]",
    "question": "[QUESTION_TEXT_2]"
  }
]

Level-3 QA rule:
- This is a question-answering task without benchmark-provided temporal or
  spatial evidence.
- Answer according to the staged video content and the question.
- If you cannot determine an answer, use an empty string.

Inner answer schema:
Return a JSON array with exactly one object per question_id:
[
  {
    "question_id": "string",
    "level3_answer": "string"
  }
]

Final response:
<answer>[JSON_ARRAY]</answer>
```

Figure 22: Benchmark-specific grouped-video prompt body for VideoZeroBench.

Calibration-set self-iteration: protocol summary

```
Inputs: previous reusable guide and binary correct/incorrect outcomes.
Update: revise generic strategies using those outcomes.
Excluded feedback: reference answers, reference trajectories, expert diagnoses,
and step-level annotations.
Output: a concise reusable guide with no case-specific facts or answer clues.
```

Figure 23: Binary-feedback calibration protocol (summary).

Audio Perception and Extraction Error (OmniGAIA, sonnet-4.6)

Question: Based on two audio clips and an image of toys-to-life gaming platforms (Disney Infinity and Skylanders), identify the fictional universes referenced in each audio clip, determine which universe was officially integrated into Disney Infinity (and how), identify which other toys-to-life platform integrated the non-Disney universe, and provide the relevant release dates.

Labeled Answer: Clip 1 references both *Kingdom Hearts* and *Sonic the Hedgehog*; clip 2 references *Sonic the Hedgehog*. Kingdom Hearts was integrated into Disney Infinity 3.0 (released August 30, 2015) via the King Mickey Power Disc and the Sora Keyblade Ultimate Unlock. Sonic was integrated into LEGO Dimensions via the Sonic Level Pack (released November 18, 2016).

Model Output by OmniAtlas-sonnet-4.6.
I'll transcribe both audio clips with Whisper and then cross-reference the YouTube IDs to identify the source material.
`tool_call` "name": "Bash", "arguments": {"command": "whisper .../audios/hYOcABfDmtk.wav –model tiny"} `/tool_call`
`tool_response` [...] "Eggman's army has everyone tear off." [...] "May your heart be your guiding key." [...] `/tool_response`
`tool_call` "name": "Bash", "arguments": {"command": "whisper .../audios/m7z29_PohpY.wav –model tiny"} `/tool_call`
`tool_response` [...] "psychotic robot doctor chasing a supersonic blue head job" [...] `/tool_response`
Instead of reasoning about the transcripts directly, let me reverse-lookup the YouTube video IDs to confirm the source.
`tool_call` "name": "WebFetch", "arguments": {"url": "https://www.youtube.com/watch?v=hYOcABfDmtk&format=json"} `/tool_call`
`tool_response` The title of the YouTube video is "Team Robot's First Adventure Begins Cinematic Trailer (Fan-Made)." `/tool_response`
`tool_call` "name": "WebSearch", "arguments": {"query": "Kingdom Hearts Disney Infinity integration Power Disc figure"} `/tool_call`
`tool_response` [D23 King Mickey Power Disc; Disney Infinity 3.0 Ultimate Unlock is Kingdom Hearts Keyblade ...] `/tool_response`
`final_answer` Audio clip 1 (hYOcABfDmtk, fan-made "Team Robot" trailer) references only the *Kingdom Hearts* universe. Clip 2 references *Sonic the Hedgehog*. Kingdom Hearts was integrated into Disney Infinity 3.0 (August 30, 2015); Sonic was integrated into LEGO Dimensions (November 18, 2016). `/final_answer`

Failure Analysis. The Whisper transcript of clip 1 already contained the explicit Sonic signal "Eggman's army." By routing source identification through the YouTube-ID lookup (which returned only the *Kingdom Hearts*-flavoured fan title) the agent overwrote its own audio evidence, and the aggregation step locked clip 1 to a single universe.

Figure 24: Representative failure case: audio perception and extraction.

Video Perception and Extraction Error (OmniGAIA, sonnet-4.6)

Question: Given an image of a riverside restaurant and an audio clip discussing duck sourcing, determine the restaurant's founding year, the duck farm's founding year, the maximum weight per duck (converted to pounds, one decimal place), and the number of ducks clearly visible swimming in the river. Compute (weight $\times$ duck count) $+$ (farm year $-$ restaurant year).

Labeled Answer: 52.0

Model Output by OmniAtlas-sonnet-4.6.
I'll transcribe the audio for the farm and weight information, then carefully inspect the image to count the ducks in the river.
`tool_call` "name": "Bash", "arguments": {"command": "whisper inputs/audios/bIlloRTNKhU.wav –model base –language en"} `/tool_call`
`tool_response` [. . .] "I brought in this Irish duck . . . Silver Hill . . . " [. . .] duck weight "from 2.2 to 2.7 kilos." [. . .] `/tool_response`
`tool_call` "name": "WebSearch", "arguments": {"query": "Silver Hill Farm duck Ireland founding year established"} `/tool_call`
`tool_response` [Silver Hill Duck, Emyvale, Co. Monaghan, founded 1962 . . .] `/tool_response`
`tool_call` "name": "WebSearch", "arguments": {"query": "Casa Rio restaurant San Antonio River Walk founding year"} `/tool_call`
`tool_response` [Casa Rio, 430 E Commerce St, San Antonio, opened 1946 . . .] `/tool_response`
Now I'll crop and enhance the water region repeatedly to count ducks.
`tool_call` "name": "Bash", "arguments": {"command": "python3 -c 'Image.open(...).crop((0, .55H, .4W, H)).save(/tmp/left_water.jpg); ...'"} `/tool_call`
`tool_response` [Sections saved: left_water.jpg, mid_water.jpg, right_water.jpg, enhanced_water.jpg, far_right_zoom.jpg, far_left_zoom.jpg] `/tool_response`
After several crop/enhance passes, I count 2 ducks on the left and 3 on the right, for a total of 5 ducks visible.
`tool_call` "name": "Bash", "arguments": {"command": "python3 -c 'w=2.7*2.2046; print(round(w,1)*5 + (1962-1946))'"} `/tool_call`
`tool_response` Product: 6.0 $\times$ 5 = 30.0; Difference: 16; Final sum: 46.0 `/tool_response`
`final_answer` 46 `/final_answer`

Failure Analysis. All non-visual evidence (restaurant identity, duck farm, per-duck weight) is correct, and the arithmetic chain is clean. The single perceptual miss – counting 5 rather than 7 swimming ducks after repeated crop/enhance passes – fully determines the final error.

Figure 25: Representative failure case: video perception and extraction.

Insufficient Exploration of Modal Content (OmniGAIA, gpt-5.4_low)

Question: Godson's video demonstrates a Clash of Clans wall-breaking technique where two P.E.K.K.A.s simultaneously attack a wall segment. Using max-level P.E.K.K.A. damage and the Level 13 wall HP (as of late 2024), how many such simultaneous volleys are needed to destroy the wall?

Labeled Answer: 3 simultaneous volleys

Model Output by OmniAtlas-gpt-5.4_low.
I'll verify the two quantities needed – max-level P.E.K.K.A. per-hit damage and Level 13 Wall HP – using web search before computing the volley count.
`tool_call` "name": "web_search", "arguments": {"query": "Clash of Clans P.E.K.K.A max level damage per hit late 2024 Home Village official"} `/tool_call`
`tool_response` {"status": null, . . . } `/tool_response`
`tool_call` "name": "web_search", "arguments": {"query": "https://clashofclans.fandom.com/wiki/Wall"} `/tool_call`
`tool_response` {"status": null, . . . } `/tool_response`
`tool_call` "name": "web_search", "arguments": {"query": "'13 |' in https://clashofclans.fandom.com/wiki/Wall"} `/tool_call`
`tool_response` {"status": null, . . . } `/tool_response`
`final_answer` 4 `/final_answer`

Failure Analysis. After two shallow lookups returned no numeric content, the agent did not retry with different queries, try fetching the wiki directly, or derive the quantity from the video. It emitted "4" with no retrieval evidence and no derivation – the failure is premature termination of exploration, not a retrieval-quality or calculation error.

Figure 26: Representative failure case: insufficient exploration of modal content.

Knowledge Retrieval and Factual Error (OmniGAIA, kimi)

Question: From two wetland-bird images identify each species and family, then use Table 1 of a cited Alabama marsh-bird study to obtain mean nest-site water depths. Given that spinnerbaits sink at 1 ft/s, compute the count-down time for each depth, pick the weight from the audio's depth–weight recommendations, and identify the optimal month for pre-spawn bass fishing in Purple Gallinule habitat.

Labeled Answer: Common Gallinule: 1.0 s count-down, 3/8 oz spinnerbait. Purple Gallinule: 1.4 s count-down, 3/8 oz spinnerbait. Optimal month: April.

Model Output by OmniAtlas-kimi.
I'll identify both species from the images and transcribe the audio for the weight–depth rule, then link to the Alabama study.
`tool_call` "name": "Read", "arguments": {"file_path": ".../images/000000156858.jpg"} `/tool_call`
`tool_response` {"type": "image"} *[Common Gallinule]* `/tool_response`
`tool_call` "name": "Read", "arguments": {"file_path": ".../images/000000189159.jpg"} `/tool_call`
`tool_response` {"type": "image"} *[Purple Gallinule]* `/tool_response`
`tool_call` "name": "Bash", "arguments": {"command": "whisper .../audios/XTnFUc2WneU.wav –model tiny"} `/tool_call`
`tool_response` [...] "three-eighths, half, three-quarter, and the one ounce ..." [rules linking depth to weight] [...] `/tool_response`
I have species and the weight rule. For the Alabama Table 1 depths I'll rely on general knowledge of marsh-bird nesting ecology.
`final_answer` Species 1: Common Gallinule (Rallidae), nest depth ≈ 0.8 ft, count-down 0.8 s, 3/8 oz. Species 2: Purple Gallinule (Rallidae), nest depth ≈ 1.5 ft, count-down 1.5 s, 1/2 oz. Optimal month: May. `/final_answer`

Failure Analysis. Image and audio evidence were correctly obtained. Without ever locating or reading the cited Alabama study, the agent substituted hallucinated nest-depth values (0.8 ft, 1.5 ft) for the actual Table 1 entries (1.0 ft, 1.4 ft) and mis-selected the optimal month. Every derived quantity (count-down, weight, month) is conditioned on this factual substitution.

Figure 27: Representative failure case: knowledge retrieval and factual error.

Logical Reasoning and Calculation Error (OmniGAIA, sonnet-4.6)

Question: Observing the image of Grand Central Terminal's four-faced clock and listening to the audio from a video about Andon lamps, determine the number of years between the terminal's opening and the lamps' estimated creation date.

Labeled Answer: 292

Model Output by OmniAtlas-sonnet-4.6.
I can see the four-faced opal clock at Grand Central Terminal; the terminal opened in 1913. Now I'll transcribe the audio to recover the Andon-lamp date reference.
`tool_call` "name": "Bash", "arguments": {"command": "whisper .../audios/VrcfGK-_iIo.wav –model base"} `/tool_call`
`tool_response` [...] "They're called Andon ... that design was created 400 years ago." [...] `/tool_response`
I have both pieces of information. Grand Central opened in 1913. The audio says Andon lamps were designed "400 years ago" – I'll anchor that to the current year (2026) to get the lamp design year.
`tool_call` "name": "Bash", "arguments": {"command": "python3 -c 'print(1913 - (2026-400))'"} `/tool_call`
`tool_response` 287 `/tool_response`
`final_answer` 287 `/final_answer`

Failure Analysis. All perceptual and retrieval evidence is correct (opening year 1913; "400 years ago" in transcript) and there are no infrastructure issues. The decisive error is temporal anchoring: "400 years ago" should be taken relative to the video's 2021 publication date (giving 1621, difference 292), not to 2026 (giving 1626, difference 287). One wrong anchor fully determines the error.

Figure 28: Representative failure case: logical reasoning and calculation.

Tool and Environment Infrastructure Failure (OmniGAIA, gpt-5.4_low)

Question: A multi-part question pairs a train image (with a visible unit number) and a Jakarta-region audio clip describing a commuter journey. Identify the train, the audio's departure station and destination, the great-circle and rail distances, the expected versus actual KRL fare, and the average journey speeds.

Labeled Answer: Train `ON 5937` (Sydney Trains H-set "Oscar"); departure Cikini, destination Bogor; straight-line ≈ 44 km; rail distance 46.8 km; expected fare Rp 6,000 matches actual; average speeds 28.1 and 35.1 km/h.

Model Output by OmniAtlas-gpt-5.4_low.
The train unit number `ON 5937` is clearly visible; the livery and numbering format are consistent with the Sydney Trains H-set. Now I need the audio transcript for the departure station.
`tool_call` "name": "command_execution", "arguments": {"command": "python -c 'import whisper, faster_whisper, speech_recognition, vosk, transformers'"} `/tool_call`
`tool_response` whisper NO; faster_whisper NO; speech_recognition NO; vosk NO; transformers NO `/tool_response`
`tool_call` "name": "command_execution", "arguments": {"command": "which whisper && ffmpeg -filters | rg 'asr|whisper|pocket'"} `/tool_call`
`tool_response` whisper: command not found; ffmpeg has no ASR filters. `/tool_response`
`tool_call` "name": "command_execution", "arguments": {"command": "which uv && uv pip install transformers accelerate tokenizers"} `/tool_call`
`tool_response` uv: not found; dependencies cannot be self-installed; no GPU/CUDA available. `/tool_response`
With no available ASR, I'll infer the departure station from web search using the KRL Commuterline context.
`tool_call` "name": "web_search", "arguments": {"query": "Manggarai Station coordinates"} `/tool_call`
`tool_response` [Manggarai railway station, Jakarta: -6.2099, 106.8502 ...] `/tool_response`
`final_answer` Train `ON 5937` (Sydney H-set/Oscar). Audio journey: Manggarai Station to Bogor. Straight-line ≈ 43 km; rail distance 44.1 km. Expected fare Rp 5,000 matches actual. Average speeds 44.1 and 35.3 km/h. `/final_answer`

Failure Analysis. Every local ASR route failed at the environment level (`whisper/faster-whisper/vosk/speech_recognition` absent; `transformers/accelerate/tokenizers` missing; GPU/CUDA unavailable; `uv` not installed so dependencies cannot be self-installed). The audio modality was blocked before the agent could produce a transcript, and the downstream "Manggarai" guess (and every distance/fare/speed value built on it) is a consequence of that infrastructure failure, not a separate reasoning error.

Figure 29: Representative failure case: tool and environment infrastructure failure.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state our four contributions: (i) showing that sandboxed coding agents are competitive with native omni-modal models on omni-modal benchmarks, (ii) an Omni Terminal Failure Taxonomy with a process-level analysis, (iii) the CODE-O training recipe with the OMNICODING dataset, and (iv) the TERMINALBENCH-O benchmark. Each claim is supported by experiments in Sections 2–4 and the appendices.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include an explicit Limitations section (Section A) discussing constraints in model scale, missing closed-source results due to credit limits, the size of our manually annotated trajectory set, and the scope being restricted to offline omni-modal tasks.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper is empirical and does not include formal theoretical results, theorems, or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 2 specifies harnesses, models, and benchmarks. Section 3 and Appendix I.4–J.1 document the dataset construction, reward design, GSPO objective, harness, hyperparameters, and prompt templates. Code is publicly available at the repository linked in the abstract; see Q5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is publicly available at `https://github.com/Dongping-Chen/OmniCoding`, as linked in the abstract. The dataset construction, benchmark design, and evaluation protocols are described in Sections 3 and 4 and the implementation appendices.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Data splits for OMNICODING are reported in Table 10; reward weights and group-level coefficients are listed in Appendix I.4; harness configuration, prompts, and runtime defaults are documented in Appendix J.1 and I.6; evaluation protocol (pass@1, per-benchmark sample sizes, judge models) is specified in Section 2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: Following the standard convention for agent-evaluation benchmarks (Omni-GAIA, LVOmniBench, SocialOmni, VideoZeroBench), we report pass@1 single-run results because each agent rollout is computationally expensive (long-horizon, multi-tool, often involving external API calls). Multiple-seed estimates are not reported for the same reason. We acknowledge this as a limitation in interpreting small numerical gaps between systems.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: CODE-O training was conducted on 8×NVIDIA H200 GPUs, with the SFT-then-RL pipeline taking approximately 14 hours total for Qwen-3.5-9B and approximately 36 hours for Qwen-3.6-27B. Sandboxed agent rollouts ran on a Slurm cluster with CPU/GPU nodes (Section 3, Appendix J.1). Closed-source backbone inference used official APIs; sandbox-side media tools could use local GPUs. Hardware and measurement limits are reported in Appendix I.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that our research complies with it. Our datasets are drawn from public-domain or research-licensed sources, no human-subjects data is collected.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We include a Broader Impacts section (Appendix B) discussing both the positive impact of terminal-mediated evidence processing with smaller final-call contexts for offline omni-modal understanding and the potential negative impacts (misuse for unauthorized content extraction, surveillance), together with mitigation strategies.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifacts are an offline-task trajectory dataset (OMNICODING), an offline benchmark (TERMINALBENCH-O), and fine-tuned Qwen-3.5-9B/Qwen-3.6-27B models specialized for sandboxed terminal tool use on omni-modal understanding tasks. All source media is drawn from public-domain or research-licensed material with no harmful or sensitive content, and the released models do not introduce capabilities (e.g., generative misuse, biometric identification) that would meaningfully exceed what existing open-source LLMs already provide. We therefore consider the release low risk and do not impose additional safeguards beyond standard licensing and documentation.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third-party benchmarks, datasets, models, and software libraries used in this work are cited in the main paper; their licenses and use terms are listed in Appendix J. We use each asset in a manner consistent with its stated license and for non-commercial academic research only.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce three new assets: the OMNICODING trajectory dataset (Section 3, Table 10), the TERMINALBENCH-O benchmark (Section 4), and the CODE-O fine-tuned Qwen models. Each is accompanied by documentation covering data sources, schema, evaluation protocol, license, and intended use; documentation is included with the released artifacts.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: Our work does not involve crowdsourcing or research with human subjects. The 200 process-level trajectory annotations were performed in-house by the authors, and no external annotators or participants were recruited.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Our work does not involve human subjects research and therefore does not require IRB approval.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are central to the methodology of this paper. Closed-source LLMs (GPT-5.4, Claude Opus/Sonnet 4.6, Gemini 3.1 Pro, etc.) and open-source LLMs (Qwen-3.5-9B, Qwen-3.6-27B, Kimi K-2.5, MiniMax M2.7) are evaluated as the agents under study; LLMs are used as process-level judges (GPT-5.5; validated against human labels in Appendix F); and we fine-tune Qwen models in our CODE-O recipe. These usages are documented in Section 2, Section 3, and Appendix J.1.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.