GeoToolAct: Evaluating Tool Execution in Remote Sensing Agents

Anonymous authors
Paper under double-blind review

Abstract

Remote sensing agents rely on multi-step tool calls to perform complex analytical tasks, yet reliable execution remains challenging. To analyze execution failures, we conceptually divide the model's tool-call generation process into two stages: Tool Plan, which selects tools and their invocation order, and Tool Act, which constructs arguments after target-tool selection and before execution. Using this distinction, our trajectory analysis shows that Tool Act errors are less frequent than Tool Plan errors but more strongly associated with end-to-end task failure. This finding motivates examining argument construction alongside tool planning. However, existing remote sensing and geospatial agent benchmarks report argument accuracy but offer limited support for comparing performance across argument-construction requirements. To enable this comparison, we analyze argument-construction errors and identify three capability dimensions: adherence to the current tool schema (Schema-Prior), locating and faithfully reproducing required historical values (Contextual Value), and determining currently valid values after resource updates (State-Update). We build GeoToolAct around these dimensions by reconstructing observed failures into diagnostic tasks with fixed target tools and controlled non-target requirements. The benchmark covers 144 core tasks and 44 target tools, with four execution-history variants per task up to 120K tokens, preserving core tasks and reference answers. Evaluating 19 models on GeoToolAct, we find that the highest Overall Direct Accuracy is only 78.53% despite explicit target-tool specification, indicating substantial room to improve argument-construction reliability. Models also exhibit substantially different performance profiles, and sensitivity to longer execution histories varies across models and capabilities. Retries with evaluator-generated argument feedback yield uneven gains and incur additional generation costs. These results highlight the need for dedicated evaluation and targeted improvement of argument-construction capabilities to support reliable tool use in remote sensing.

1 Introduction

Advances in large language models (LLMs) for planning and tool use are extending remote sensing agents beyond image perception and understanding to urban change analysis, agricultural and forest monitoring, and geospatial analysis. Specialized tools and multi-agent collaboration support these workflows (Xu et al., 2026b; Xiao & Ma, 2025; Lee et al., 2025; Sun et al., 2026), spanning data retrieval, image preprocessing, spatial analysis, and product generation with diverse interface constraints and parameter conventions (Feng et al., 2026). Long chains of dependent calls require selecting appropriate intermediate products for downstream operations and tracking changes in their identifiers and validity, making remote sensing and GIS a demanding setting for studying general challenges in reliable argument construction. Reliable execution remains challenging in multi-step tool use (Feng et al., 2026; Shabbir et al., 2025). Our analysis of representative benchmark trajectories further shows that end-to-end (E2E) task success declines as observed tool calls increase (Fig. 1(a1)).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

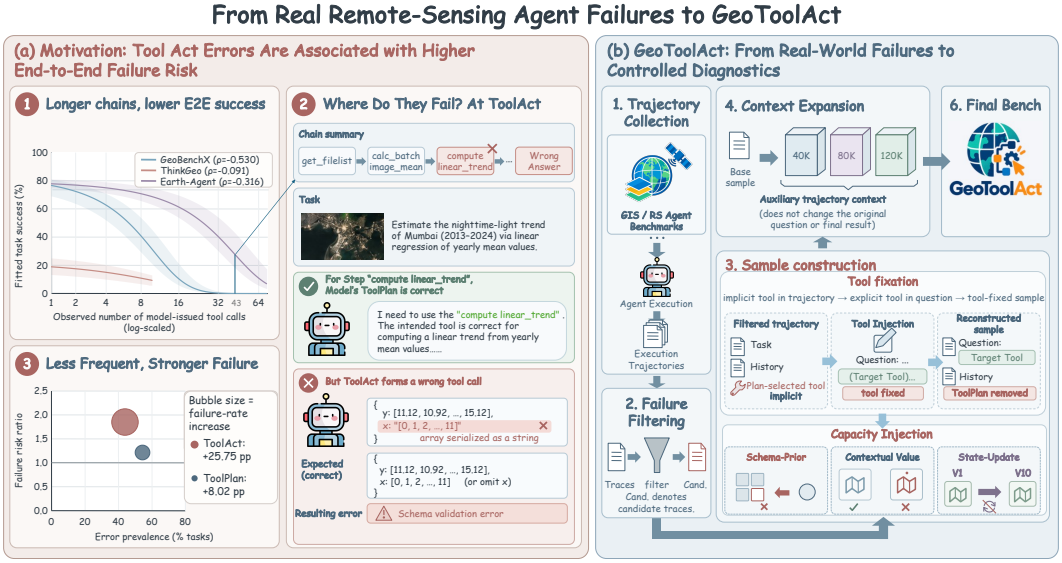


Figure 1: **Motivation and overview of GeoToolAct.** (a) E2E success decreases with observed tool-call count. Execution analysis distinguishes Tool Plan from Tool Act and shows a higher observed failure risk ratio for Tool Act errors (1.85× vs. 1.22×). (b) GeoToolAct isolates Tool Act through target-tool fixation and controlled reconstruction along Schema-Prior, Contextual Value, and State-Update, with task-invariant context expansion.

To identify E2E failure sources, we examine agents' selected tools, generated arguments, and tool responses. We find that, even with the correct tool, models may violate parameter type or structural constraints, bind incorrect historical values, or make other input-construction errors. In Fig. 1(a2), the model correctly selects `compute_linear_trend` but encodes a numerical array as a string, failing interface validation.

Selecting the target tool and constructing its inputs are distinct decisions. Accordingly, we conceptually divide the tool-calling process into two stages based on the decisions required of the model. **Tool Plan** determines which tools to invoke and in what order, whereas **Tool Act** refers to the model-controlled process between target tool selection and actual tool execution, in which the model constructs a complete and accurate tool call based on the task intent, tool interface, and execution history. Thus, Tool Act errors can arise independently of Tool Plan errors and constitute a potential source of end-to-end task failure.

We analyze Qwen3.5-27B execution trajectories for all 248 Earth-Bench tasks (Feng et al., 2026; Qwen Team, 2026). An LLM-based judge assesses both Tool Plan and Tool Act errors in every trajectory; the two error types may co-occur. For each type separately, we compare E2E failure rates between trajectories with and without that error using a failure risk ratio (RR). As shown in Fig. 1(a3), Tool Act errors occur in 109 trajectories (44.0%), compared with 54.4% for Tool Plan errors, with corresponding RRs of 1.85× and 1.22×. These descriptive associations motivate examining argument construction alongside tool planning; the statistical protocol is detailed in Appendix H.

However, although existing remote sensing and geospatial agent benchmarks include metrics such as tool-call accuracy and argument accuracy, their evaluations largely remain embedded in end-to-end agent execution, offering limited support for isolated, fine-grained diagnosis of Tool Act capabilities. To address this gap, we introduce **GeoToolAct**, a controlled diagnostic benchmark for argument construction in remote sensing agents. Its central contribution is to turn argument-correctness evaluation into targeted diagnosis of interface adherence, historical value binding, and state maintenance by specifying the target tool and controlling non-target task requirements.

As illustrated in Fig. 1(b), we first run models on existing remote sensing and geospatial agent benchmarks, collecting tool calls and execution feedback to form complete task trajectories. We then use an error filter to extract candidate Tool Act failures and analyze their parameter-reasoning requirements to derive three capability dimensions: Schema-Prior, Contextual Value, and State-Update. We explicitly provide the target tool to remove tool-selection interference and perform controlled task reconstruction around these dimensions, producing diagnostic instances with well-defined reference answers. Finally, deterministic answer verification and task-invariant context expansion yield GeoToolAct for systematic evaluation across execution-history lengths.

Evaluation of 19 representative models reveals substantial Tool Act limitations: with the target tool specified, the highest Overall Direct Accuracy is 78.53%, with distinct Schema-Prior, Contextual Value, and State-Update profiles. History sensitivity varies: GLM-5's State-Update accuracy drops from 84.4% at Base to 7.8% at 120K, while other model–capability pairs show non-monotonic changes. Feedback-based correction yields larger average gains on Schema-Prior than on State-Update, at additional generation cost. These findings highlight capability heterogeneity, long-context robustness, recoverability, and efficiency challenges.

Our contributions are threefold:

1. We distinguish **Tool Plan** from **Tool Act** and derive three capability dimensions from failure analysis, providing a fine-grained framework for characterizing tool-input performance.
2. We introduce **GeoToolAct**, using failure-driven sample mining and controlled task construction to isolate Tool Act evaluation across execution-history lengths.
3. We evaluate 19 models, revealing capability disparities, long-context degradation, and limited error recoverability to inform targeted parameter-reasoning improvements and capability-aware recovery mechanisms.

2 Related Work

Remote Sensing and Geospatial Agents. Remote sensing and geospatial intelligence is expanding from image understanding to tool-based task execution. Vision–language and multimodal advances enable natural-language interaction (Kuckreja et al., 2024; Zhang et al., 2024; Zhan et al., 2025; Li et al., 2025; Shao et al., 2025). LLM-Geo, GeoGPT, and GIS Copilot translate requests into geospatial workflows and executable operations (Li & Ning, 2023; Zhang et al., 2023; Akinboyewa et al., 2025); RS-Agent and ChangeAgent support remote sensing task orchestration and interactive change analysis with specialized tools (Xu et al., 2026b; Liu et al., 2024). GeoLLM-Engine and GeoLLM-Squad integrate tools and multi-agent collaboration into complex geospatial workflows (Singh et al., 2024; Lee et al., 2025). Earth-Agent, GeoMMAgent, and GeoDisaster extend tool-augmented analysis to multi-source Earth observation, expert geoscience question answering, and disaster assessment (Feng et al., 2026; Xiao et al., 2026; Hasan et al., 2026). Interdependent calls require appropriate tool selection and accurate argument construction under interface constraints, historical value dependencies, and resource-state changes. GeoToolAct provides fine-grained diagnosis of this argument-construction process for remote sensing and geospatial agents.

Benchmarking Remote Sensing and Geospatial Agents. Domain benchmarks cover workflow generation, multi-step tool use, and execution. GeoAnalystBench evaluates spatial analysis workflows and code generation, and GeoBenchX evaluates multi-step geospatial tasks (Zhang et al., 2025; Krechetova & Kochedykov, 2025). ThinkGeo and Earth-Bench assess tool use and complex Earth observation tasks (Shabbir et al., 2025; Feng et al., 2026). GeoAgentBench and GISAgentBench evaluate task completion in execution environments, with GeoAgentBench reporting Parameter Execution Accuracy (Yu et al., 2026; Pothuri et al., 2026). GeoMMBench and GeoDisaster cover expert multimodal question answering and disaster analysis (Xiao et al., 2026; Hasan et al., 2026). TaskBench assesses parameter prediction in task automation; BFCL and ToolSandbox also assess stateful tool use (Shen

et al., 2024; Patil et al., 2025; Lu et al., 2025). Complementing these evaluations, including GeoAgentBench's Parameter Execution Accuracy for executed workflows, GeoToolAct assesses argument construction at predefined decision points with the target tool supplied. Controlled schema presentation, value sources, and resource updates emphasize interface adherence, historical value binding, or current-state validity for comparing model profiles. Paired history-length variants preserve task intent, target tool, target evidence, and reference call, enabling within-task comparisons as execution history grows.

3 Tool Act: Task Definition and Capability Decomposition

To formalize the input-output structure and correctness criteria of Tool Act, and to characterize the reasoning capabilities required for constructing correct tool inputs, we first provide a formal task definition (Sec. 3.1) and then decompose Tool Act into three core capability dimensions (Sec. 3.2) based on an analysis of execution failures.

3.1 Formal Definition

Tool Act Task Definition. To establish the task boundary of parameter construction within the tool-calling process, we define Tool Act as the model-controlled decision process that occurs after the target tool has been determined but before its actual execution. Its objective is to generate a complete and accurate structured tool call based on the task intent, tool interface, and prior execution history. Formally, given the current task intent q, a tool registry $\mathcal{R}$, a predetermined target tool $t \in \mathcal{R}$, and the preceding execution history H, we denote the effective interface constraints of the target tool by $S = \text{schema}_{\mathcal{R}}(t)$. Tool Act is then formulated as

$$\hat{c} = M(q, t, S, H), \tag{1}$$

where M denotes the model performing Tool Act and $\hat{c}$ is the generated tool call. If the output can be parsed into a structured call to the target tool, it is represented as $\hat{c} = (t, \hat{a})$, where $\hat{a}$ denotes the predicted argument dictionary. Successful completion of Tool Act requires the generated call to satisfy the current interface constraints while accurately determining parameter values and their current validity from the task intent and execution history. In GeoToolAct, we explicitly provide the target tool as a known input to eliminate the influence of Tool Plan and independently evaluate the model's parameter-construction capability.

3.2 Capability Decomposition

From recurring Tool Act errors in agent execution trajectories, we derive three complementary performance dimensions: **Schema-Prior**, **Contextual Value**, and **State-Update** (Fig. 2). Diagnostic scores characterize observed performance under these argument-construction requirements, rather than identify internal failure mechanisms.

Schema-Prior: Interface Constraint Compliance. Schema-Prior evaluates adherence to the current tool schema when constructing argument fields, types, and structures, including cases where supplied values or familiar API conventions suggest a different representation. Fig. 2(a) illustrates a required list incorrectly generated as a string.

Contextual Value: Contextual Parameter Binding. LLMs generate argument values through next-token prediction, which does not guarantee exact reproduction of values available in the execution history. Contextual Value evaluates whether models can locate the required historical values and faithfully reproduce them in the corresponding argument fields, as illustrated by the file-path error in Fig. 2(b).

State-Update: Current-State Validity Assessment. In multi-step execution, subsequent operations and tool feedback may change the state of resources referenced earlier, for example through renaming, replacement, or invalidation. State-Update evaluates whether models can track these changes and use the currently valid values in the target call, as illustrated by the use of an outdated artifact in Fig. 2(c). These dimensions describe complementary, overlapping requirements of argument construction: State-Update combines

contextual binding with determining current validity after resource updates. The controlled diagnostic tasks in Sec. 4 each emphasize one of these requirements.

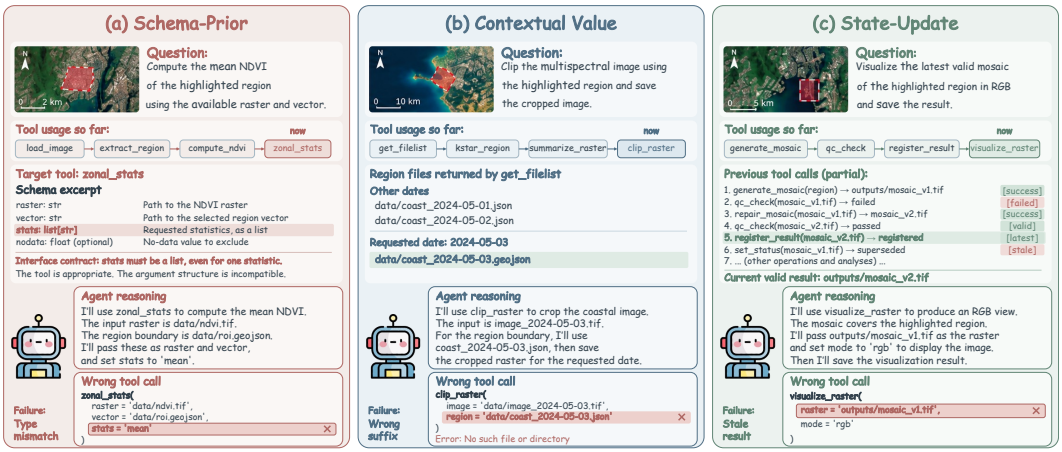


Figure 2: **Three core capability dimensions of Tool Act.** (a) Schema-Prior: Violating the required parameter type. (b) Contextual Value: Binding an incorrect file path. (c) State-Update: Selecting an outdated artifact. The examples illustrate distinct failures in interface compliance, contextual value binding, and state tracking.

4 GeoToolAct Benchmark Construction

Building on the Tool Act failures identified above, we construct GeoToolAct to transform execution-time failures into independently evaluable and controllable diagnostic tasks. The benchmark construction pipeline consists of four stages: **Execution Trajectory Collection** (Sec. 4.1), **Failure Filtering** (Sec. 4.2), **Controlled Diagnostic Task Reconstruction** (Sec. 4.3), and **Task-Invariant Context Expansion** (Sec. 4.4). We further apply automated consistency verification and human quality assessment (Sec. 4.5) to ensure the validity and reliability of the finalized benchmark.

4.1 Execution Trajectory Collection

We run Qwen3.5-27B and DeepSeek-v4-flash on ThinkGeo, Earth-Bench/Earth-Agent, and GeoAgentBench, collecting complete trajectories containing task requests, tool calls, and execution feedback for failure filtering. These benchmarks cover multi-stage analysis, long tool-call sequences, and intermediate-artifact interactions. Appendix F details their task settings and tool suites.

4.2 Failure Filtering

Gemini-3-Flash serves as the LLM Judge, using only information available before each call to classify it as a Tool Plan error, Tool Act error, or no error. Each Tool Act error receives one predominant capability subtype. Cases are grouped into three capability-specific pools and prioritized by the empirical Tool Act error rates of their associated tools; Appendix A provides the full protocol.

4.3 Controlled Diagnostic Task Reconstruction

After error filtering, we obtain candidate Tool Act failures grouped by their primary capability requirements. However, error categorization alone cannot eliminate tool-selection interference during re-execution, nor can it ensure effective separation among different parameter-reasoning capabilities. We therefore apply two operations—target-tool fixation and capability-controlled reconstruction—to transform these candidates into independently evaluable diagnostic tasks.

Tool Fixation. Tool calls in the original agent trajectories involve both tool selection and argument construction. Consequently, even when an erroneous tool call is observed, end-to-end execution outcomes cannot isolate failures attributable to Tool Act. To eliminate tool selection as a confounding factor, we identify the correct target tool for the current decision point from each candidate trajectory and explicitly provide it to the model, while preserving the task intent, tool interface, and preceding execution history required for the call. The model therefore no longer needs to select the target tool and is instead evaluated on constructing the correct arguments for the specified tool. We further validate this design through a paired controlled experiment in Appendix B.

Capability-Controlled Reconstruction. Although error filtering categorizes candidate cases into three major capability dimensions, a complete tool call often involves multiple parameter-reasoning requirements. Error labels alone therefore cannot ensure that a diagnostic task primarily evaluates its intended capability. To address this issue, inspired by evaluation approaches that control non-target subtasks by providing correct intermediate information (Tsunokake et al., 2026), we reconstruct the task questions and execution contexts of candidate cases in a controlled manner. Specifically, we design control conditions along three aspects: schema presentation, parameter-value sources, and resource-state changes. Table 1 summarizes the controls. Schema-Prior explicitly supplies all target argument values; Contextual Value requires retrieval from designated historical sources without resource updates; State-Update supplies initial values and the necessary updates, with a uniquely determined final state. These controls emphasize the intended requirement while reducing non-target demands, but the task families still differ in information presentation, tool and template distributions, and argument requirements. Cross-family score differences therefore describe empirical difficulty under these diagnostic conditions.

Table 1: Controlled task design for the three capability dimensions in GeoToolAct. Bold text denotes the requirement emphasized by each dimension.

Capability	Schema Constraints	Parameter Values	Resource State
Schema-Prior	**Schema provided earlier**; not repeated near the question	Initial values provided directly	No state changes
Contextual Value	Parameter requirements repeated near the question	**Historical source specified**; values must be located and bound	No state changes
State-Update	Parameter requirements repeated near the question	Initial values provided directly	**Resource modifications and feedback introduced**

The two-level controlled reconstruction transforms Tool Act failure candidates into diagnostic instances with predetermined target tools and clearly defined primary diagnostic capabilities. Each instance retains the task intent, tool interface, and execution context required for argument construction and is paired with a reference tool call, providing the basis for subsequent task-invariant context expansion.

4.4 Task-Invariant Context Expansion

These core instances have a fixed target tool and a clearly defined primary diagnostic capability. To evaluate Tool Act stability across execution-history lengths, we perform task-invariant context expansion. This task-invariant design reduces confounding from differences in task content and difficulty, enabling more direct comparison across context-length variants. Specifically, we sample unrelated tool calls and their execution results from a shared pool of other tasks across the three source benchmarks and insert them into the core instances as distractor history, constructing Base, 40K, 80K, and 120K variants (actual token counts: Appendix C). During expansion, we keep the task intent, target tool, target evidence, and reference call unchanged, while updating the corresponding references whenever the inserted history shifts the position of the target evidence.

4.5 Consistency Verification and Quality Assessment

Following automated consistency verification, the finalized GeoToolAct benchmark contains 144 core diagnostic instances, including 54 Schema-Prior, 60 Contextual Value, and 30 State-Update instances. Each core instance is instantiated at four context lengths (Base, 40K, 80K, and 120K), resulting in 576 evaluation samples in total. We further conduct a human audit on a subset of the finalized benchmark to assess task clarity, reference-call correctness, and the specificity of the intended diagnostic capability. Detailed review protocols and results are provided in Appendix D.

5 Experiments

5.1 Experimental Setup

Evaluated Models. We evaluate 15 general-purpose and four remote-sensing specialized models (Table 2). The general-purpose models span Gemini-3-Flash (Google DeepMind, 2025), DeepSeek-v4-flash (Xu et al., 2026a), GPT-5 (OpenAI, 2025), Qwen3.5 (Qwen Team, 2026), GLM-5/5.2 (Zeng et al., 2026), GLM-4.7-Flash (Z.ai, 2026), Kimi-K2.7-Code (Moonshot AI, 2026b), Kimi-K2.6 (Moonshot AI, 2026a), Gemma-4 (Team et al., 2026), and Nemotron-3-Nano (Blakeman et al., 2025). The specialized models comprise RSCoVLM-7B-2512 (Li et al., 2026), RemoteSensing-Qwen-3B, GeoLLaVA-8K-text (Wang et al., 2026), and EarthDial-4B-RGB (Soni et al., 2025).

Evaluation Settings. Across the four context lengths, **Direct Accuracy** measures first-generation tool-call correctness. We normalize parsed calls, fill declared defaults, and compare the resulting tool and arguments with the normalized reference; semantically equivalent representations are accepted. Overall Accuracy weights capability scores by sample counts. All models use temperature = 1.0, top-p = 0.95, top-k = 50, and a 16,384-token generation limit; Appendix E examines decoding sensitivity. After an incorrect call, we retain the output as an assistant message and append evaluator-generated argument feedback as a user message. Feedback identifies erroneous fields or violations without revealing reference argument values (Appendix G). We stop upon success or after four attempts, including the initial generation. We report **ReAct@4 Accuracy**, **Avg. Turns**, and **Token Overhead**, the mean sample-level additional completion-token cost relative to the first generation: $\frac{1}{N}\sum_i (T_i^{\text{all}}/T_i^{(1)} - 1)$.

5.2 Main Results

Overall Direct Accuracy ranges from 0.69% to 78.53% (Table 2). The leading models, Gemini-3-Flash (78.53%) and DeepSeek-v4-flash (77.14%), have similar overall scores but different profiles: Gemini-3-Flash performs better on Schema-Prior and State-Update, whereas DeepSeek-v4-flash achieves the highest Contextual Value accuracy.

The 15 general-purpose models average 64.6%, 50.2%, and 37.6% Direct Accuracy on Schema-Prior, Contextual Value, and State-Update, respectively. This aggregate ordering is not universal: Gemini-3-Flash and GPT-5 both perform strongly on State-Update, illustrating model-specific performance profiles.

The four remote-sensing specialized models reach at most 5.86% Overall Direct Accuracy, suggesting that specialization in perception, understanding, or question answering does not ensure reliable argument reasoning. Three are evaluated only through 80K due to context-window limits, so their aggregates cover a different context-length range from models evaluated at all four lengths.

Error analysis across the 19 models identifies type, field, and default-handling errors in Schema-Prior, value errors in Contextual Value, and stale-value reuse or cross-version mixing in State-Update. These patterns are consistent with the intended diagnostic requirements (Appendix I).

Table 2: **Main results on GeoToolAct.** Direct results average three runs; values are reported as mean ± population standard deviation. † indicates models evaluated on Base, 40K, and 80K only due to model-side context-window limitations.

Model	Schema	Context	State	Overall	Turns	Tok. OH
General Models						
Gemini-3-Flash	**86.88 ± 0.44**	66.39 ± 3.30	**87.78 ± 5.54**	**78.53 ± 2.55**	1.35 ± 0.04	0.32 ± 0.12×
DeepSeek-v4-flash	73.61 ± 3.27	**84.44 ± 3.09**	68.89 ± 3.75	77.14 ± 2.44	1.40 ± 0.04	0.45 ± 0.17×
Qwen3.5-Plus	74.69 ± 7.76	66.67 ± 0.59	64.17 ± 4.08	69.16 ± 3.05	1.49 ± 0.03	0.98 ± 0.20×
GLM-5.2	83.95 ± 0.95	56.81 ± 1.87	57.22 ± 1.42	67.07 ± 1.22	1.72 ± 0.04	0.22 ± 0.06×
GPT-5	67.59 ± 0.76	55.00 ± 1.23	86.67 ± 1.18	66.32 ± 0.79	1.47 ± 0.05	0.47 ± 0.08×
GLM-5	72.69 ± 1.31	66.11 ± 0.39	38.61 ± 1.96	62.85 ± 0.65	1.79 ± 0.01	0.87 ± 0.07×
Qwen3.5-122B-A10B	77.16 ± 2.79	64.31 ± 1.42	18.06 ± 1.42	59.49 ± 1.93	1.87 ± 0.02	1.39 ± 0.78×
Kimi-K2.7-Code	60.19 ± 3.84	67.92 ± 2.70	38.61 ± 6.14	58.91 ± 2.10	1.81 ± 0.06	0.65 ± 0.02×
Kimi-K2.6	59.10 ± 1.22	57.92 ± 0.90	37.78 ± 5.71	54.17 ± 1.78	1.94 ± 0.04	0.56 ± 0.05×
Qwen3.5-35B-A3B	62.65 ± 1.53	42.08 ± 2.38	9.17 ± 2.45	42.94 ± 0.22	2.32 ± 0.02	1.37 ± 0.18×
gemma-4-12B-it	59.26 ± 3.09	24.86 ± 0.20	29.17 ± 3.12	38.66 ± 1.21	2.64 ± 0.09	2.54 ± 0.11×
Qwen3.5-9B	46.91 ± 0.44	36.25 ± 3.02	10.83 ± 0.68	34.95 ± 1.01	2.51 ± 0.02	2.07 ± 0.12×
GLM-4.7-Flash	58.33 ± 2.30	28.19 ± 1.29	0.83 ± 0.68	33.80 ± 0.83	2.67 ± 0.15	2.46 ± 0.20×
gemma-4-26B-A4B-it	49.54 ± 1.00	26.94 ± 1.42	16.67 ± 0.68	33.28 ± 1.00	2.65 ± 0.11	3.33 ± 0.27×
Nemotron-3-Nano-30B-A3B	36.73 ± 2.52	8.61 ± 0.20	0.00 ± 0.00	17.36 ± 0.93	3.01 ± 0.38	4.24 ± 0.55×
Remote-Sensing Specialized Models						
RSCoVLM-7B-2512†	3.29 ± 0.29	**11.11 ± 1.64**	0.00 ± 0.00	**5.86 ± 0.72**	3.70 ± 0.02	5.14 ± 1.08×
RemoteSensing-Qwen-3B†	**4.53 ± 0.58**	0.00 ± 0.00	0.00 ± 0.00	1.70 ± 0.22	3.88 ± 0.00	11.05 ± 2.10×
GeoLLaVA-8K-text	3.40 ± 0.58	0.00 ± 0.00	0.00 ± 0.00	1.27 ± 0.22	3.89 ± 0.01	2.56 ± 0.05×
EarthDial-4B-RGB†	1.85 ± 0.50	0.00 ± 0.00	0.00 ± 0.00	0.69 ± 0.19	3.96 ± 0.01	12.39 ± 3.45×

5.3 Long-Context Robustness

To assess Tool Act robustness under accumulated execution history, we compare Direct Accuracy on the same core task across four context conditions: Base, 40K, 80K, and 120K. In Figure 3(a), longer histories are associated with lower accuracy for several models, though not always monotonically. For example, GLM-5's State-Update accuracy declines progressively from 84.4% at Base to 7.8% at 120K, whereas its Contextual Value accuracy drops from 100.0% to 31.1% at 80K before recovering to 55.0% at 120K. These results show no simple, uniform decline across capabilities as histories lengthen.

Sensitivity to execution history also varies across models and capabilities. DeepSeek-v4-flash's Contextual Value accuracy drops to 70.0% at 40K before recovering to 88.9% at 120K, whereas its State-Update accuracy declines from 82.2% at Base to 53.3% at 120K. By comparison, Gemini-3-Flash's Contextual Value accuracy declines from 89.4% at Base to 50.0% at 120K, while its State-Update accuracy decreases from 94.4% to 78.9%. Thus, even with the core task and correct answer unchanged, sensitivity to longer histories varies across models and capabilities, a pattern aggregate accuracy alone cannot adequately capture.

5.4 Feedback Correction and Cost

We further evaluate recovery from Tool Act errors with ReAct@4, combining evaluator-generated argument feedback with additional generation attempts. Direct Accuracy measures correctness on the initial generation; ReAct@4 Accuracy measures correctness after up to four generations, including the initial attempt. Figure 3(b) presents each model's post-feedback performance across the three capability dimensions.

Gains from Direct Accuracy to ReAct@4 Accuracy vary across models and task families (Figure 3(a–b)). Several models improve considerably on Schema-Prior, whereas some with weak State-Update performance improve little. For example, Qwen3.5-35B-A3B approaches perfect Schema-Prior accuracy after feedback, while its State-Update accuracy remains low.

ReAct@4 accuracy gains must also be assessed alongside correction costs. Figure 3(b–c) shows substantial variation in post-feedback accuracy and correction costs across models. Some achieve high post-feedback accuracy with few generation attempts, whereas others approach the four-generation limit with substantial Token Overhead yet retain low accuracy on certain capabilities. Thus, additional feedback attempts need not yield commensurate

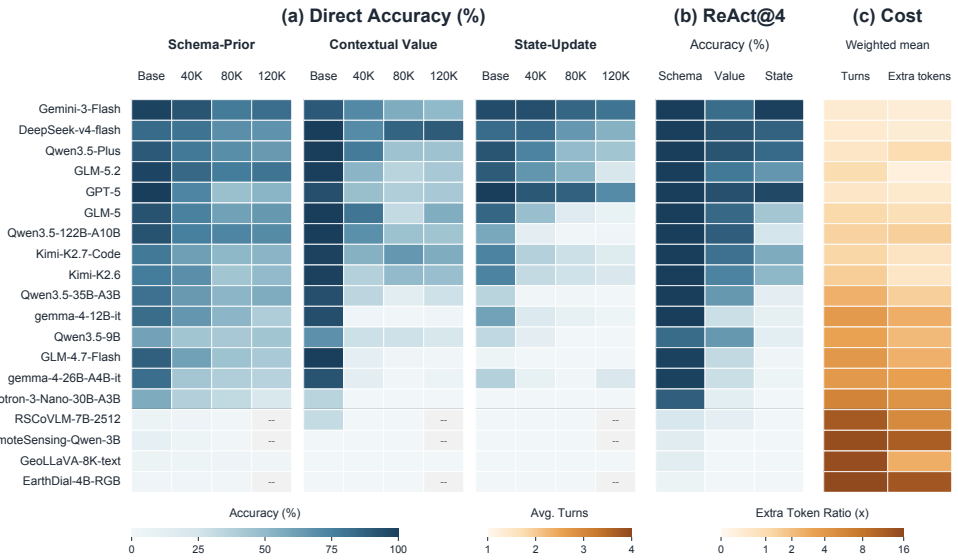


Figure 3: **Tool Act performance across execution-history lengths, feedback correction, and computational cost.** (a) Direct Accuracy across four context lengths (Base, 40K, 80K, and 120K) for the three capability dimensions. (b) ReAct@4 Accuracy after up to four generation attempts, including the initial attempt. (c) Overall correction cost, measured by average generation attempts (Avg. Turns) and Token Overhead. All 19 models are included; unavailable context-length results are marked as missing.

accuracy gains, and evaluating Tool Act feedback-based correction requires considering both final performance and additional cost.

6 Discussion

Tool Act warrants dedicated evaluation and improvement. Even with the target tool specified, the best Overall Direct Accuracy is only 78.53%, indicating that correct tool selection alone cannot ensure reliable argument construction. Across 19 models, performance profiles differ across the three task families, so aggregate accuracy alone obscures which argument-construction requirements models struggle to meet. Future agent research should evaluate and improve interface adherence, historical argument binding, and state maintenance alongside tool planning.

Long execution histories further expose capability-specific reliability issues. With the core task fixed, performance under longer histories varies across models and capabilities, indicating that larger context windows do not guarantee robust argument reasoning. This motivates explicit evidence indexing and execution-state management to reduce reliance on lengthy raw histories.

Uneven recovery across task families motivates targeted improvement. Across the feedback-evaluation runs, average Direct Accuracy decreases from Schema-Prior to Contextual Value and State-Update, as do average recovery gains. Under this feedback protocol, task families with lower average Direct Accuracy show smaller average recovery gains despite additional generation and token costs. Future work should explore capability-specific, cost-aware recovery mechanisms.

Limitations. Evaluation covers 19 representative models; cost caps context analysis at 120K tokens. Future work will examine generality across more models and the substantially longer histories modern agents may encounter.

AI use statement

We used generative AI to assist with manuscript translation, organization, and argument refinement, with the authors responsible for reviewing its outputs and for the final content.

References

Temitope Akinboyewa, Zhenlong Li, Huan Ning, and M Naser Lessani. Gis copilot: Towards an autonomous gis agent for spatial analysis. *International Journal of Digital Earth*, 18 (1):2497489, 2025.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*, 2025.

Chi-Min Chan, Han Zhu, Chunyang Jiang, Jiaming Ji, Juntao Dai, Wei Xue, Sirui Han, and Yike Guo. Benchmarking fine-grained error detection in multimodal reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44672–44702, 2026.

Peilin Feng, Zhutao Lv, Junyan Ye, Xiaolei Wang, Xinjie Huo, Jinhua Yu, Wanghan Xu, Wenlong Zhang, Lei Bai, Conghui He, et al. Earth-agent: Unlocking the full landscape of earth observation with agents. In *International Conference on Learning Representations*, volume 2026, pp. 55700–55764, 2026.

Google DeepMind. Gemini 3 Flash Model Card, 2025. URL `https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf`.

Maram Hasan, Aman Verma, Savitra Roy, Hariseetharam Gunduboina, Daksh Jain, Muhammad Haris Khan, Subhasis Chaudhuri, and Biplab Banerjee. Geodisaster: Benchmarking orchestrated agents for operational disaster geo-intelligence. *arXiv preprint arXiv:2606.17246*, 2026.

Varvara Krechetova and Denis Kochedykov. Geobenchx: Benchmarking llms in agent solving multistep geospatial tasks. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Generative and Agentic AI for Multi-Modality Space-Time Intelligence*, pp. 27–35, 2025.

Kartik Kuckreja, Muhammad Sohail Danish, Muzammal Naseer, Abhijit Das, Salman Khan, and Fahad Shahbaz Khan. Geochat: Grounded large vision-language model for remote sensing. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27831–27840. IEEE, 2024.

Chaehong Lee, Varatheepan Paramanayakam, Andreas Karatzas, Yanan Jian, Michael Fore, Heming Liao, Fuxun Yu, Ruopu Li, Iraklis Anagnostopoulos, and Dimitrios Stamoulis. Multi-agent geospatial copilots for remote sensing workflows. In *IGARSS 2025-2025 IEEE International Geoscience and Remote Sensing Symposium*, pp. 1084–1089. IEEE, 2025.

Qingyun Li, Shuran Ma, Junwei Luo, Yi Yu, Yue Zhou, Fengxiang Wang, Xudong Lu, Xiaoxing Wang, Xin He, Yushi Chen, et al. Co-training vision-language models for remote sensing multi-task learning. *Remote Sensing*, 18(2):222, 2026.

Zhenlong Li and Huan Ning. Autonomous gis: the next-generation ai-powered gis. *International Journal of Digital Earth*, 16(2):4668–4686, 2023.

Zhenshi Li, Dilxat Muhtar, Feng Gu, Yanglangxing He, Xueliang Zhang, Pengfeng Xiao, Guangjun He, and Xiaoxiang Zhu. Lhrs-bot-nova: Improved multimodal large language model for remote sensing vision-language interpretation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 227:539–550, 2025.

Chenyang Liu, Keyan Chen, Haotian Zhang, Zipeng Qi, Zhengxia Zou, and Zhenwei Shi. Change-agent: Toward interactive comprehensive remote sensing change interpretation and analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–16, 2024.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

Moonshot AI. Kimi-K2.6, 2026a. URL `https://huggingface.co/moonshotai/Kimi-K2.6`. Official model card.

Moonshot AI. Kimi-K2.7-Code, 2026b. URL `https://huggingface.co/moonshotai/Kimi-K2.7-Code`. Official model card.

OpenAI. GPT-5 System Card, 2025. URL `https://openai.com/index/gpt-5-system-card/`.

Shrey Pandit, Austin Xu, Xuan-Phi Nguyen, Yifei Ming, Caiming Xiong, and Shafiq Joty. Hard2verify: A step-level verification benchmark for open-ended frontier math. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22502–22517, 2026.

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 *Forty-second International Conference on Machine Learning*, 2025.

Abhinav Pothuri, Zhe Jiang, Zelin Xu, and Di Yang. Gisagentbench: A practitioner-sourced benchmark for evaluating llm agents on gis tasks. *arXiv preprint arXiv:2608.01645*, 2026.

Qwen Team. Qwen3.5: Towards Native Multimodal Agents, 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Akashah Shabbir, Muhammad Akhtar Munir, Akshay Dudhane, Muhammad Umer Sheikh, Muhammad Haris Khan, Paolo Fraccaro, Juan Bernabe Moreno, Fahad Shahbaz Khan, and Salman Khan. Thinkgeo: Evaluating tool-augmented agents for remote sensing tasks. *arXiv preprint arXiv:2505.23752*, 2025.

Run Shao, Cheng Yang, Qiujun Li, Linrui Xu, Xiang Yang, Xian Li, Mengyao Li, Qing Zhu, Yongjun Zhang, Yansheng Li, et al. Allspark: A multimodal spatiotemporal general intelligence model with ten modalities via language as a reference framework. *IEEE Transactions on Geoscience and Remote Sensing*, 63:1–20, 2025.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. Taskbench: Benchmarking large language models for task automation. *Advances in Neural Information Processing Systems*, 37:4540–4574, 2024.

Simranjit Singh, Michael Fore, and Dimitrios Stamoulis. Geollm-engine: A realistic environment for building geospatial copilots. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 585–594. IEEE, 2024.

Sagar Soni, Akshay Dudhane, Hiyam Debary, Mustansar Fiaz, Muhammad Akhtar Munir, Muhammad Sohail Danish, Paolo Fraccaro, Campbell D Watson, Levente J Klein, Fahad Shahbaz Khan, et al. Earthdial: Turning multi-sensory earth observations to interactive dialogues. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14303–14313. IEEE, 2025.

Zhelun Sun, Yuyu Zhou, and Jun Yang. An llm-based multi-agent system for remote sensing analysis. *Big Earth Data*, pp. 1–25, 2026.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Masaya Tsunokake, Yuta Koreeda, Terufumi Morishita, Koichi Nagatsuka, Hikaru Tomonari, and Yasuhiro Sogawa. Is micro domain-adaptive pre-training effective for real-world operations? multi-step evaluation reveals potential and bottlenecks. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pp. 304–316, 2026.

Fengxiang Wang, Mingshuo Chen, Yueying Li, Di Wang, Haotian Wang, Zonghao Guo, Zefan Wang, Shan Boqi, Long Lan, Yulin Wang, et al. Geollava-8k: Scaling remote-sensing multimodal large language models to 8k resolution. *Advances in Neural Information Processing Systems*, 38:159185–159218, 2026.

Aoran Xiao, Shihao Cheng, Yonghao Xu, Yexian Ren, Hongruixuan Chen, and Naoto Yokoya. Geommbench and geommagent: Toward expert-level multimodal intelligence in geoscience and remote sensing. *arXiv preprint arXiv:2604.08896*, 2026.

Zixuan Xiao and Jun Ma. Llm agent framework for intelligent change analysis in urban environment using remote sensing imagery. *Automation in Construction*, 177:106341, 2025.

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

Wenjia Xu, Zijian Yu, Boyang Mu, Jiuniu Wang, Zhiwei Wei, and Mugen Peng. Rs-agent: Automating remote sensing tasks through intelligent agent. *Science China Information Sciences*, 69(8):180302, 2026b.

Bo Yu, Cheng Yang, Dongyang Hou, Chengfu Liu, Jiayao Liu, Chi Wang, Zhiming Zhang, Haifeng Li, and Wentao Yang. Geoagentbench: A dynamic execution benchmark for tool-augmented agents in spatial analysis. *arXiv preprint arXiv:2604.13888*, 2026.

Z.ai. GLM-4.7-Flash, 2026. URL `https://huggingface.co/zai-org/GLM-4.7-Flash`. Official model card.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Yang Zhan, Zhitong Xiong, and Yuan Yuan. Skyeyegpt: Unifying remote sensing vision-language tasks via instruction tuning with large language model. *ISPRS Journal of Photogrammetry and Remote Sensing*, 221:64–77, 2025.

Qianheng Zhang, Song Gao, Chen Wei, Yibo Zhao, Ying Nie, Ziru Chen, Shijie Chen, Yu Su, and Huan Sun. Geoanalystbench: A geoai benchmark for assessing large language models for spatial analysis workflow and code generation. *Transactions in GIS*, 29(7):e70135, 2025.

Wei Zhang, Miaoxin Cai, Tong Zhang, Yin Zhuang, and Xuerui Mao. Earthgpt: A universal multimodal large language model for multisensor image comprehension in remote sensing domain. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–20, 2024.

Yifan Zhang, Cheng Wei, Shangyou Wu, Zhengting He, and Wenhao Yu. Geogpt: Understanding and processing geospatial tasks through an autonomous gpt. *arXiv preprint arXiv:2307.07930*, 2023.

A Tool Act Error Screening and the LLM Judge Protocol

To support reproducible candidate error screening, this appendix details the LLM Judge inputs, error taxonomy, evidence requirements, and candidate prioritization strategy used in GeoToolAct. We collect trajectories from benchmark runs using Qwen3.5-27B and DeepSeek-v4-flash, and use Gemini-3-Flash to analyze them with a predefined classification prompt.

A.1 Judge Inputs and Outputs

The judge receives the complete agent execution trajectory, the current task request, available tool definitions and schemas, and optional reference planning information. The trajectory is organized chronologically, with a stable identifier assigned to each actual tool call.

Although the input contains the full trajectory, the judge evaluates individual tool calls and is explicitly instructed to use only information available before each call, avoiding judgments that require anticipating future states. It must cover every actual tool call in trajectory order and return structured JSON containing the call identifier, tool name, error label, Tool Act subtype, confidence, a brief rationale, and supporting evidence.

A.2 Hierarchical Error Taxonomy

To distinguish errors in tool planning from those in argument construction, we use three mutually exclusive primary labels.

Tool Plan Error. A call is labeled as a Tool Plan error if the selected tool is unsuitable for the task, a necessary prerequisite is missing, the invocation order is incorrect, or the tool is called repeatedly without a valid purpose. Alternative tools or execution paths that can correctly complete the task are not labeled as errors merely because they differ from the reference plan.

Tool Act Error. A call is labeled as a Tool Act error if the selected tool is appropriate but its arguments are incorrect. To support subsequent controlled reconstruction focused on individual capabilities, each Tool Act error is assigned one subtype: Schema-Prior covers violations of the tool interface constraints; Contextual Value covers schema-valid arguments with incorrect values drawn from context; and State-Update covers failures to use the currently valid resource following an explicit state update. If a call exhibits multiple Tool Act issues, the judge selects the subtype most directly responsible for the execution failure or semantic deviation.

No Error. A call is labeled as having no error if both tool selection and argument construction are consistent with the information available before the call. Failures caused solely by external factors, such as timeouts, rate limits, unavailable backends, tool implementation defects, corrupted data packages, or contradictory tool documentation, are not attributed to the model. This label is also used when the evidence is insufficient to establish a model error.

A.3 Decision Priority and Evidence Requirements

To avoid duplicate attribution, the judge assigns only one primary label to each call. When both tool selection and arguments are incorrect, the Tool Plan error label takes precedence. Arguments are evaluated for Tool Act errors only when the selected tool is appropriate.

A failed tool execution does not automatically imply a model error, and successful execution does not necessarily imply correct arguments. Repeated and cascading calls are evaluated individually, and a later correction does not invalidate an earlier error label. For each error judgment, the judge must provide brief evidence and a rationale grounded in the task request, tool interface, prior observations, or relevant execution information.

A.4 Candidate Prioritization by Error Rate

To prioritize tools for which model executions frequently exhibit argument-construction difficulties, we aggregate calls across tasks by tool and compute the empirical Tool Act error rate:

$$r(t) = \frac{N_{\text{ActErr}}(t)}{N_{\text{Calls}}(t)}, \tag{2}$$

where $N_{\text{ActErr}}(t)$ is the number of calls to tool t labeled as Tool Act errors, and $N_{\text{Calls}}(t)$ is the total number of calls to that tool.

Using the capability subtypes assigned by the judge, we construct three mutually exclusive candidate pools for Schema-Prior, Contextual Value, and State-Update. Candidates are prioritized in descending order of their associated tool error rates. This ranking guides the allocation of subsequent controlled reconstruction effort; it is not interpreted as an unbiased estimate of intrinsic tool difficulty.

A.5 Complete Judge Prompt

We provide the complete system prompt and user prompt template used during candidate screening to support reproducibility. The system prompt specifies the error taxonomy, decision rules, evidence requirements, and JSON output constraints, while the user prompt supplies the task information, tool definitions, and execution trajectory.

Listing 1: System prompt for Tool Plan and Tool Act error classification.

```
You are an evaluator of agent tool-use trajectories. Evaluate every actual tool call using the user
     request, tool definitions, tool schemas, and information visible before that call. Do not judge the
     final answer and do not solve the task again.

Assign exactly one label to every tool call:

- tool_plan_error
- tool_act_error
- no_error

Definitions:

1. tool_plan_error
The error concerns what action to take: the selected tool is unsuitable or irrelevant to the current goal,
      is called before a required prerequisite, is used in the wrong order, or is repeated without a valid
      purpose. Do not mark a tool as wrong merely because it differs from a reference chain when it is a
     valid alternative.

2. tool_act_error
The selected tool is appropriate, but its arguments are incorrect. For every tool_act_error, assign
     exactly one subtype:

- schema_prior: the arguments violate the tool contract, such as a missing or unknown field, wrong type or
      JSON structure, invalid enum/range/format, or a list/number/boolean incorrectly encoded as a string.
- contextual_value: the arguments are schema-valid but bind the wrong value from the task or context, such
      as the wrong file, date, band, object, bbox, threshold, or a fabricated/mistyped resource.
- state_update: the trajectory produced a newer artifact or state, but the call uses an older path, image,
      object, version, or handle.

3. no_error
The tool choice and arguments are reasonable based on the information available before the call. Also use
     no_error when failure is caused only by timeout, rate limiting, an unavailable backend, a tool
     implementation bug, missing/corrupted packaged data, contradictory tool documentation, or
     insufficient evidence of a model error.

Decision rules:

- Judge each call only from information available before it. Do not use future observations to demand
     foresight.
- Wrong tool plus wrong arguments -> tool_plan_error only.
- Appropriate tool plus schema-invalid arguments -> tool_act_error / schema_prior.
- Appropriate tool plus schema-valid but incorrectly selected contextual value -> tool_act_error /
     contextual_value.
- Appropriate tool plus failure to use the latest explicit state -> tool_act_error / state_update.
- A tool returning an error does not automatically mean the model made an error. A successful call may
     still be semantically wrong.
- Judge repeated and cascading calls separately. A later repair does not erase an earlier error.
```

```
- If several Tool Act issues coexist, choose the subtype that most directly caused the failure or semantic
      deviation.
- If evidence cannot establish a model error, use no_error rather than guessing.
- For every error label, cite a short decisive excerpt from the user request, tool schema, prior
     observation, or tool result.
- Keep the reason concise. Do not output chain-of-thought.

Return only valid JSON:

{
  "task_id": "string",
  "call_judgments": [
    {
      "call_id": "call_001",
      "tool_name": "string",
      "label": "no_error",
      "tool_act_subtype": null,
      "confidence": 0.0,
      "brief_reason": "Short evidence-based reason.",
      "evidence": {
        "source": "user_request | tool_schema | prior_observation | tool_result | reference_plan",
        "reference": "turn or call identifier",
        "quote": "short decisive excerpt"
      }
    }
  ]
}

Output constraints:

- Cover every actual tool call exactly once and preserve trajectory order.
- label must be exactly tool_plan_error, tool_act_error, or no_error.
- tool_act_subtype must be schema_prior, contextual_value, or state_update when label is tool_act_error;
     otherwise it must be null.
- confidence must be between 0 and 1.
```

Listing 2: User prompt template for trajectory-level input.

```
Evaluate the following agent trajectory according to the system prompt.

<task_id>{{TASK_ID}}</task_id>

<user_request>
{{USER_REQUEST}}
</user_request>

<available_tools>
{{TOOL_DEFINITIONS_AND_SCHEMAS}}
</available_tools>

<reference_plan optional="true">
{{REFERENCE_PLAN_OR_NULL}}
</reference_plan>

<trajectory>
{{CHRONOLOGICAL_TRAJECTORY_WITH_STABLE_CALL_IDS}}
</trajectory>

Return only the required JSON.
```

B Validation of Target-Tool Fixation

To validate the target-tool fixation used in GeoToolAct, we conduct a paired experiment on 149 tool-call decision points collected from EarthAgent and ThinkGeo using Qwen3.5-27B. Each decision point is evaluated under two conditions: **Free Selection**, where the model independently selects the tool and its arguments, and **Oracle Fixation**, where the reference target tool is explicitly provided while the remaining context is kept unchanged.

Correctness criteria and denominators. Each output is first normalized to a tool name and an argument dictionary; outputs that cannot be normalized are counted as format failures. **ActAcc Given Correct Tool** is computed over parseable outputs that select the correct target tool, giving denominators of 125 for Free Selection and 147 for Oracle Fixation.

Table 3: Paired validation of target-tool fixation. ActAcc Given Correct Tool uses parseable outputs selecting the correct tool: 125 for Free and 147 for Oracle. Full Call uses all 149 decision points.

Metric	**Free**	**Oracle**
Tool Selection	83.9%	98.7%
Full Call	77.2%	90.6%
ActAcc Given Correct Tool	92.0%	91.8%

For **ActAcc Given Correct Tool**, argument correctness is assessed by deterministic comparison followed by semantic review of initially unmatched outputs. The checker validates required fields, unknown fields, top-level types, and enum values against the tool schema, then compares ordinary argument values using structured equality. Two predefined equivalences are allowed: omitting an optional argument or explicitly supplying its published default, and choosing a different non-empty, type-valid name for a newly generated output resource. During the programmatic comparison, arbitrary path substitutions, approximate numerical values, and list reordering are not accepted.

All 98 parseable, correct-tool outputs not accepted by this comparison across the two conditions undergo semantic review. The review considers the current task request, the history available before the call, the tool schema, the source benchmark's reference call, and, where necessary, the public tool implementation. It accepts 76 outputs as equivalent and retains 22 as incorrect, yielding ActAcc of 115/125 for Free Selection and 135/147 for Oracle Fixation. The review was performed by the experiment executor; it did not use an LLM judge or independent double-blind annotation. Decisions and per-output reasons are recorded in `manual_argument_review.csv`.

Full Call uses the same semantic argument-correctness criterion, additionally requires a parseable output and the correct tool, and divides by all 149 decision points. The paired ActAcc comparison instead evaluates both conditions on the same 125 decision points where Free Selection already selects the correct tool; this shared subset differs from the 147 outputs used for Oracle Fixation's conditional ActAcc in the table.

As shown in Table 3, fixing the target tool increases tool-selection accuracy from 83.9% to 98.7% and full-call accuracy from 77.2% to 90.6%. More importantly, among the 125 instances where the model already selects the correct tool under Free Selection, parameter accuracy changes by only 0.0 percentage points (95% CI: -3.2 to 3.2). Meanwhile, 12 of the 147 target-tool-compliant outputs under Oracle Fixation still contain incorrect arguments, indicating that providing the target tool does not reveal the correct tool call or eliminate the underlying Tool Act challenge.

C Actual Prompt Token Counts Across Models and Context-Length Settings

Base/40K/80K/120K use the reference tokenizer tiktoken `cl100k_base`. Table 4 reports mean Contextual Value prompt lengths for five open-source models, measured with their own tokenizers. Closed-source API models are omitted because their tokenizers are not locally accessible.

Table 4: Mean prompt token counts for Contextual Value.

Model	Base	40K	80K	120K
gemma-4-12B-it	4.8K	54.2K	111.6K	160.0K
gemma-4-26B-A4B-it	4.8K	54.2K	111.6K	160.0K
GLM-4.7-Flash	3.7K	40.3K	82.9K	118.8K
Nemotron-3-Nano-30B-A3B	4.6K	52.4K	108.1K	155.0K
Qwen3.5-9B	4.4K	49.4K	101.7K	145.8K

Actual token counts generally exceed the nominal labels, with relative differences of approximately 20%–35% across models. These differences primarily arise from how efficiently each tokenizer encodes structured strings such as paths, UUIDs, and checksums. They do not alter the ordering of the four context lengths or the within-model trends discussed in Sections 5.2 and 5.3. However, comparisons of absolute performance across models at a given nominal context length should account for the actual token counts in this table.

D Benchmark Quality Evaluation

Before reporting model performance on GeoToolAct, we systematically assess the quality of the benchmark itself. Following data validation practices that combine automated construction with human quality control (Chan et al., 2026; Pandit et al., 2026), we assess two complementary aspects: deterministic consistency verification over all benchmark instances and independent expert review of semantic correctness and diagnostic validity.

Automatic Consistency Verification. To ensure consistency in tool interfaces, argument sources, and execution states, we perform deterministic verification on all benchmark instances using structured metadata retained during construction. Specifically, the verifier checks whether each gold call satisfies the current tool schema, whether argument values originate from the intended historical calls or returned results, whether the gold call in each State-Update instance reflects the current valid state at the end of the update sequence, and whether all context-length variants of a core instance preserve the task objective and gold call. We also submit every gold call to the unified evaluator to verify parsing and correctness under the rules in Appendix G. Only instances that pass all checks are retained in the final benchmark. We define the **Automatic Verification Rate** as the proportion of instances passing all checks:

$$\text{AutoVerify} = \frac{N_{\text{passed}}}{N_{\text{all}}}. \tag{3}$$

For the final release, all 576 retained context-expanded samples pass automatic verification: 216 Schema-Prior, 240 Contextual Value, and 120 State-Update samples (Table 5).

Human Quality Verification. Automatic verification can determine whether predefined construction rules are followed, but cannot fully establish whether a task is semantically clear, whether its gold call accords with human understanding of the remote sensing workflow, or whether controlled reconstruction introduces ambiguities outside the programmed checks. We therefore conduct an independent human quality assessment, following the human verification approach of PRMBench-V (Chan et al., 2026) and expert review practices in human-validated benchmarks (Pandit et al., 2026).

GeoToolAct includes three diagnostic task families—Schema-Prior, Contextual Value, and State-Update—and four context-length settings. We use stratified random sampling to select 10 instances from each task-family–context-length combination, yielding

$$3 \times 4 \times 10 = 120 \tag{4}$$

instances for review.

Two researchers with backgrounds in remote sensing or geospatial analysis and familiarity with programmatic tool use independently review all 120 instances. A third researcher with the same qualifications adjudicates disagreements. Before formal review, the annotators use calibration samples excluded from the final statistics to familiarize themselves with the task format, tool schemas, three diagnostic task families, and shared assessment criteria.

For each instance, both annotators independently assess four criteria:

1. **Task Solvability:** Whether the task intent, target tool, tool schema, and execution history provide sufficient information to uniquely determine the correct tool call.

Table 5: Benchmark quality evaluation for GeoToolAct. Final Automatic Verification is computed over all 576 retained context-expanded samples. Human metrics are computed on a stratified audit of 120 samples, with 40 samples from each capability. The four criterion-level rates and Human Acceptance use the adjudicated first-pass judgments. Human OA denotes the observed agreement between the two independent annotators before adjudication. After correcting two State-Update samples that did not satisfy Diagnostic Validity, all audited samples passed expert re-verification.

Quality Metric	**Schema**	**Value**	**State**	**Overall**
Final Automatic Verification	100.0% (216/216)	100.0% (240/240)	100.0% (120/120)	100.0% (576/576)
Task Solvability	100.0% (40/40)	100.0% (40/40)	100.0% (40/40)	100.0% (120/120)
Gold Correctness	100.0% (40/40)	100.0% (40/40)	100.0% (40/40)	100.0% (120/120)
Diagnostic Validity	100.0% (40/40)	100.0% (40/40)	95.0% (38/40)	98.3% (118/120)
Context Consistency	100.0% (40/40)	100.0% (40/40)	100.0% (40/40)	100.0% (120/120)
Human Acceptance (first pass)	100.0% (40/40)	100.0% (40/40)	95.0% (38/40)	98.3% (118/120)
Human OA (first pass)	100.0% (40/40)	100.0% (40/40)	95.0% (38/40)	98.3% (118/120)
Human Verification (post-correction)	100.0% (40/40)	100.0% (40/40)	100.0% (40/40)	100.0% (120/120)

2. **Gold Correctness:** Whether the gold call satisfies the task semantics, current tool schema, and execution history, with correctly specified arguments.

3. **Diagnostic Validity:** Whether the instance reflects its intended Schema-Prior, Contextual Value, or State-Update requirement, without interference from non-target capabilities that clearly dominates task difficulty.

4. **Context Consistency:** For instances with expanded contexts, whether the added history preserves the original task semantics, target tool, valid state, and gold call without introducing additional valid answers.

An instance is considered valid only if it satisfies all four criteria. The criterion-level rates and the **Human Acceptance Rate** use the adjudicated first-pass judgments, before sample correction:

$$\text{HumanAcceptance} = \frac{N_{\text{valid}}}{N_{\text{reviewed}}}, \tag{5}$$

and the **Observed Agreement (OA)** between the two independent annotators on overall instance validity before adjudication:

$$\text{OA} = \frac{N_{\text{agree}}}{N_{\text{reviewed}}}. \tag{6}$$

Any instance judged invalid by either annotator, or on which the annotators disagree, is referred to the third expert for adjudication. The adjudicated first-pass review identified two State-Update samples that did not satisfy Diagnostic Validity, yielding Human Acceptance of 118/120 (98.3%). Both samples were corrected, and all 120 audited samples subsequently passed expert re-verification. Table 5 reports the first-pass judgments and agreement separately from post-correction verification.

Together, automatic consistency verification and expert review provide complementary evidence of benchmark quality. Verification over all instances checks that schemas, argument sources, state updates, and context expansion follow the predefined construction rules. Expert review further assesses semantic correctness, unique solvability, and diagnostic validity. These checks support the analysis of overall Tool Act performance, differences across argument-reasoning capabilities, degradation under long contexts, and recoverability with feedback.

Table 6: Direct Accuracy (%) for two models under three decoding configurations. Bold indicates the highest accuracy for each model–capability pair.

Model	Capability	Low	Medium	High
GLM-4.7-Flash	Schema-Prior	53.0	**56.9**	35.2
	Contextual Value	**37.9**	30.0	22.1
	State-Update	0.8	**1.7**	0.0
Qwen3.5-9B	Schema-Prior	41.4	**46.3**	31.9
	Contextual Value	26.2	**30.4**	17.1
	State-Update	13.3	**14.2**	0.0

E Sensitivity to Decoding Configuration

To assess sensitivity to decoding configuration on GeoToolAct, we evaluate GLM-4.7-Flash and Qwen3.5-9B while holding the evaluation instances, target tools, and execution contexts fixed across configurations. We jointly vary temperature, top-p, and top-k to form three configurations: low (0.5/0.85/20), medium (1.0/0.95/50), and high (1.5/0.98/100). The medium configuration uses the same sampling parameters as the main experiments; this supplementary experiment is excluded from the main-result aggregation.

Table 6 shows that both models retain the ordering Schema-Prior, Contextual Value, and State-Update from highest to lowest accuracy under all three configurations, although accuracy values vary. The medium configuration yields the highest Schema-Prior and State-Update accuracy for both models. For Contextual Value, GLM-4.7-Flash performs best under the low configuration, whereas Qwen3.5-9B performs best under the medium configuration. The evidence for consistent capability ordering is limited to these two models and three configurations; it does not establish the same ordering for other models or a universally optimal decoding configuration. Jointly varying the three sampling parameters also prevents attribution to any individual parameter.

F Data Sources and Coverage

GeoToolAct draws its core diagnostic instances from execution trajectories on EarthBench/Earth-Agent, ThinkGeo, and GeoAgentBench (Feng et al., 2026; Shabbir et al., 2025; Yu et al., 2026). The system prompt supplies a toolbox assembled from the complete tool inventories of all three source benchmarks. Context expansion uses a shared pool of tool calls and results from these sources. We distinguish toolbox coverage, referring to interfaces supplied to the model, from target-tool coverage, referring to tools for which argument construction is actually evaluated.

The benchmark contains 144 core instances, each corresponding to a distinct task scenario, and involves 44 target-tool interfaces spanning 43 distinct tool names. Table 8 summarizes the instance counts. Each core instance has four execution-history variants, yielding 576 evaluation samples. Context-length variants of the same core instance are counted once when reporting the number of core instances; they are not independent new tasks.

Across the benchmark, these task scenarios span six broad application and operation categories, summarized in Table 7.

The counts are computed from one Base record per core instance. We assign each record to the category of its primary target operation; supporting operations can cross category boundaries. In the first category, the present target inventory primarily covers raster thresholding, hotspot analysis, and spatial data processing. The 44 target interfaces have 43 distinct names: `build_crop_stress_assessment` has two argument contracts, one with ten historical input paths and one with four current input paths plus an output path. It counts once by name and twice by interface. All 60 Contextual Value instances use the ten-input contract; their required resource values vary while the target operation is shared. Thus, instance counts should not be read as counts of distinct tool templates.

Table 7: Coverage of the 144 Base instances, grouped once by the primary target operation. Names count distinct tool names; interfaces also distinguish their argument contracts. The six categories summarize application coverage, not independent workflow templates.

Application or operation category	**Instances**	**Names**	**Interfaces**
Vector processing and spatial analysis	20	11	11
Remote sensing indices and parameter retrieval	16	10	10
Change and geometry analysis	6	4	4
Time-series and statistical analysis	12	8	8
Multi-source agricultural analysis	64	1	2
Ecological, environmental, and disaster monitoring product synthesis	26	9	9
Total	144	43	44

Each task family uses tools and argument structures that instantiate its intended diagnostic requirement in remote sensing and GIS workflows. Target-tool inventories may therefore differ across families. We assess the correspondence between tasks and their intended requirements through the construction controls in Sec. 4.3, the expert review of diagnostic validity in Appendix D, and the observed error forms in Appendix I.

Table 8: Core instances and context-expanded evaluation samples. Each core instance has four context-length variants. The 44 target interfaces span 43 distinct tool names.

Capability	Core instances	Evaluation samples
Schema-Prior	54	216
Contextual Value	60	240
State-Update	30	120
Total	144	576

G Correctness Criteria and Feedback Protocol

Direct run aggregation. For Gemini-3-Flash, the three Direct observations comprise two standalone runs and the first, pre-feedback generation from one ReAct run; no feedback-assisted attempt contributes to Direct Accuracy. Each run includes 216 Schema-Prior, 240 Contextual Value, and 120 State-Update samples across the four context lengths. We aggregate correct counts within each run before computing the mean and population standard deviation across runs. Parse failures remain in the denominator. Standalone Direct observations do not contribute to ReAct accuracy or correction-cost statistics.

Correctness for reconstructed tasks. GeoToolAct evaluates constrained diagnostic calls. Controlled reconstruction specifies the target tool and determines the required argument values through directly supplied values, designated historical evidence, or a uniquely resolved resource state (Sec. 4.3). The reference call records these requirements at the evaluated decision point. Thus, argument matching is grounded in the task request, current interface, and designated resources. For example, a different file identifier does not satisfy a request to use a specified historical artifact merely because it is a syntactically valid path. The automatic checks and expert review in Appendix D assess reference correctness, unique solvability, and the preservation of valid answers across context variants.

Normalization before validation. The evaluator parses the generated JSON into `tool` and `parameters`, normalizes the call, and completes omitted optional arguments with their declared defaults when the task specifies no replacement. Parameter values are canonicalized according to the task and tool schema before comparison with the reference under the same rules. Semantically equivalent representations are accepted through this normalization, including differences in JSON whitespace or object-key order, equivalent numeric representations such as `1` and `1.0`, and permitted variations in names assigned to newly

generated output resources. These examples illustrate the equivalence handling rather than exhaust its scope: correctness depends on the normalized tool, argument structure, and values, not the original serialization.

Normalization preserves the identity and version of required input resources and the constraints imposed by the tool schema. A stale artifact, an unrelated input resource, or a schema-incompatible argument structure remains incorrect. Missing required arguments, fields outside the permitted argument set, and values that violate the task requirements are rejected. Default completion also does not permit an invented replacement value when the task specifies none.

The supplementary fixation experiment in Appendix B evaluates 149 source-benchmark decision points and uses semantic review to assess initially unmatched calls in those original task contexts. Its manual equivalence decisions concern that separate task set. GeoToolAct's main evaluation accepts equivalent calls through the normalization and validation rules above, applied to the requirements of the reconstructed tasks.

After an unsuccessful attempt, the model's preceding output is appended as an `assistant` message, followed by evaluator-generated feedback as a `user` message. The original input and accumulated conversation are retained. Depending on the detected error, feedback can identify parsing or tool-name errors, missing or unexpected fields, type or structure violations, or incorrect argument values. It may repeat a submitted invalid value or state an expected schema type, but does not supply the correct reference argument value. The model is then asked to regenerate the complete JSON call. Generation stops upon success or after four attempts, including the initial attempt.

The following example illustrates feedback for an incorrect argument value; `<incorrect_value>` denotes the value submitted in the preceding attempt.

Listing 3: Example evaluator feedback for an incorrect argument value.

```
Tool execution failed.

Incorrect parameter values: `input_path`
The submitted value for `input_path` is invalid: `<incorrect_value>`

Please generate the complete corrected JSON tool call again.
Return only the JSON object.
```

This protocol measures recovery with evaluator-generated argument feedback; its results are specific to the information supplied by that feedback.

H Trajectory-Level Error Association

Earth-Bench contains 248 tasks (Feng et al., 2026). We analyze one Qwen3.5-27B execution trajectory per task, comprising 103 E2E failures and 145 successes. Both Tool Plan and Tool Act errors are assessed in every trajectory and may co-occur. Table 9 reports the counts. Tool Act errors occur in 109 trajectories (44.0%), and Tool Plan errors in 135 (54.4%).

Table 9: Trajectory-level contingency counts. Each error type is assessed separately over the same 248 trajectories; columns denote E2E outcomes.

	Tool Act		Tool Plan	
Error presence	Failure	Success	Failure	Success
Present	61	48	61	74
Absent	42	97	42	71
Total	103	145	103	145

For each error type E, let $p_1 = \Pr(F \mid E)$ and $p_0 = \Pr(F \mid \neg E)$, where F denotes E2E failure. We compute the failure risk ratio as $\mathrm{RR}_E = p_1/p_0$ and the risk difference as

$100(p_1 - p_0)$ percentage points. For Tool Act, the failure rates are 55.96% with errors and 30.22% without, giving RR = 1.85 and a risk difference of +25.75 pp. For Tool Plan, the rates are 45.19% and 37.17%, giving RR = 1.22 and a risk difference of +8.02 pp. Figure 1(a3) plots error prevalence against RR, with bubble area proportional to the risk difference.

Approximate 95% confidence intervals for RR are $[1.37, 2.51]$ for Tool Act and $[0.90, 1.65]$ for Tool Plan. We use the log-Wald interval $\exp(\log \mathrm{RR} \pm 1.96\,\mathrm{SE})$, where $\mathrm{SE} = \sqrt{1/a - 1/(a+b) + 1/c - 1/(c+d)}$ and (a, b, c, d) denote the present/failure, present/success, absent/failure, and absent/success counts, respectively. Two-sided Fisher exact tests yield $p = 5.45 \times 10^{-5}$ for Tool Act and $p = 0.2443$ for Tool Plan. These tests separately assess each error type's association with failure; they do not test whether the two risk ratios differ. The analysis is observational and does not establish a causal effect of either error type.

I Capability-Specific Error Analysis

We summarize errors across all 19 models and three runs per model. Counts refer to erroneous generated calls, not distinct core instances. In Schema-Prior, 11 failures involve other argument-value errors; the remaining failures involve incorrect field types, fields absent from the schema, or unsupported overrides of declared defaults. A default-handling error occurs when the model invents a replacement value although the task supplies none; explicitly providing the correct declared default is accepted.

All Contextual Value failures involve incorrect argument values, with no observed parsing, tool-name, argument-key-set, or type errors. An incorrect value need not match another historical source: it may also reflect a failure to accurately extract or reproduce the target value. In State-Update, 19 failures combine information from multiple historical versions, while all remaining failures reuse an outdated value. Together with the task construction controls, these patterns support correspondence between the intended diagnostic requirements and observed error forms, without establishing complete independence among the three capabilities.