Geo-OLM: Enabling Sustainable Earth Observation Studies with Cost-Efficient Open Language Models & State-Driven Workflows

Dimitrios Stamoulis dstamoulis@utexas.com Chandra Family Department of Electrical and Computer Engineering The University of Texas at Austin Austin, TX, USA Diana Marculescu dianam@utexas.edu Chandra Family Department of Electrical and Computer Engineering The University of Texas at Austin Austin, TX, USA

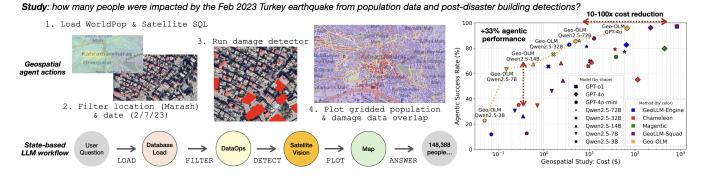


Figure 1: Earth observation studies, such as post-disaster societal impact assessments [38], involve complex workflows with multiple tools, data sources, and analytical steps. Automating such tasks with off-the-shelf Copilots – particularly agents using proprietary models such as GPT-40 – can rapidly escalate costs, making large-scale spatiotemporal assessments unsustainable for practitioners and researchers. We introduce Geo-OLM, a geospatial agentic framework based on state-driven prompting, where reasoning is structured into distinct stages. By decoupling task progression from tool calling, Geo-OLM enables competitive agentic performance with low-resource open language models while substantially reducing costs. [Project repo]

Abstract

Geospatial Copilots hold immense potential for automating Earth observation (EO) and climate monitoring workflows, yet their reliance on large-scale models such as GPT-40 introduces a paradox: tools intended for sustainability studies often incur unsustainable costs. Using agentic AI frameworks in geospatial applications can amass thousands of dollars in API charges or requires expensive, powerintensive GPUs for deployment, creating barriers for researchers, policymakers, and NGOs. Unfortunately, when geospatial Copilots are deployed with open language models (OLMs), performance often degrades due to their dependence on GPT-optimized logic. In this paper, we present Geo-OLM, a tool-augmented geospatial agent that leverages the novel paradigm of state-driven LLM reasoning to decouple task progression from tool calling. By alleviating the workflow reasoning burden, our approach enables low-resource OLMs to complete geospatial tasks more effectively. When downsizing to small models below 7B parameters, Geo-OLM outperforms the strongest prior geospatial baselines by 32.8% in successful query completion rates. Our method performs comparably to proprietary models achieving results within 10% of GPT-40, while reducing inference costs by two orders of magnitude from \$500-\$1000 to under \$10. We present an in-depth analysis with geospatial downstream benchmarks, providing key insights to help practitioners effectively deploy OLMs for EO applications.

1 Introduction

Agentic AI has emerged as a powerful tool in geospatial sciences, where AI Copilots have the potential to transform Earth observation (EO) workflows [4]. EO-based studies, such as climate and environmental monitoring, are inherently complex, requiring the processing of vast datasets, integration of multiple geospatial APIs, visualization through mapping systems [11, 13], and implementation of analytical techniques to identify long-term spatiotemporal trends [36]. These tasks demand domain expertise in geosciences and are critical for sustainability research, making them prime candidates for automation through agentic AI [43].

Geospatial agents – tool-augmented Large Language Models (LLMs) – have demonstrated their ability to support climate studies and agriculture studies [25, 53], urban planning [2, 55], and disaster response [14] by automating data processing [4], visualization [56], and analysis. However, a fundamental **paradox** arises: while these tools are positioned as key enablers of sustainability research, their reliance on large-scale proprietary models incurs significant computational and financial costs, ultimately compromising their own sustainability. This challenge arises from prevailing agentic AI methodologies, which often prioritize maximizing Copilot performance over computational efficiency and cost-effectiveness [12].

To manage the complexity of geospatial applications, advanced LLM reasoning frameworks with multi-agent orchestration [12, 29] rely on large models like GPT-40, whose usage is prohibitively expensive. Conducting real-world geoscientific studies over satellite imagery spanning multiple terabytes – requiring iterative querying, multi-step prompting, and extended user interactions – can quickly accumulate costs in the thousands of dollars. Deploying open-source LLMs also presents significant challenges, as applying these agentic schemes to open language models (OLMs) results in severe performance degradation [24]. Moreover, scaling with large OLMs (e.g., LLaMA-3.3 70B) demands A100-class GPUs, which themselves are costly and have a high carbon footprint. This raises critical questions about accessibility, particularly for researchers, NGOs, and policymakers – stakeholders who stand to benefit most from geospatial AI but might be least equipped to bear these costs.

To address the need for sustainable agentic EO applications, our goal is to enable geospatial agents that operate effectively with small and mid-sized OLMs, typically models with 14B parameters or fewer [1]. However, at this smaller scale, OLMs struggle with the complex reasoning required for geospatial tasks. To this end, we draw the attention to an aspect often underleveraged in agentic AI design: unlike generalist Copilots designed for broader tasks like web browsing or document summarization [59], geospatial workflows follow a structured, sequential progression [40]. Consider a geoscientist: they must first load and preprocess data before performing filtering, and similarly, data analysis must be completed before generating map visualizations. This inherent structure presents us with a key opportunity: by explicitly modeling this progression, we can design more efficient agentic prompting that align with geospatial task dependencies and is better suited for smaller OLMs.

In this work, we present Geo-OLM, geospatial agents that leverage the recently proposed state-driven LLM prompting paradigm [50] to alleviate the reasoning burden in low-resource OLMs. Our approach models geospatial workflows as discrete states and transitions, decoupling process grounding and task execution from tool calling and sub-task solving. Each state is described by its expected functionalities and supplemented with few-shot examples to facilitate tool selection, so transitions can be guided by lightweight LLM-based logic [50]. This mitigates the computational overhead of multi-round agentic orchestration logic, enabling smaller OLMs to achieve higher task completion rates at significantly lower cost.

We evaluate our approach on state-of-the-art geospatial agentic benchmarks, GeoLLM-Engine [41] and GeoLLM-Squad [24], which encompass single- and multi-agent EO workflows, respectively. We conduct a comprehensive evaluation across various prompting schemes and LLM APIs, including proprietary GPT models and open LLM families (e.g., LLaMA [10], Phi [1], Qwen [52]). Our analysis spans multiple API endpoints such as OpenAI and Ollama [34]. To our knowledge, this is the first work to leverage state-driven prompting within OLMs for geospatial tasks.

Our results show that, when tested on proprietary models like GPT-40, Geo-OLM achieves performance within 1% of state-of-the-art methods. When applied to smaller OLMs, our approach maintains robust performance at a fraction of the cost, reducing expenses from over \$1,000 to less than \$10 on EO benchmarks – a 100x decrease. More importantly, at that downsized scale with models of 14B parameters or less (e.g., Qwen-2.5-7B [52]), existing geospatial solutions experience a performance collapse by up to 60% in completion rates, while Geo-OLM remains within 10-20% of the

large baselines. Overall, state-driven reasoning offers substantial cost benefits, achieving near-GPT performance. We demonstrate the generalizability of Geo-OLM beyond benchmark scenarios through a case study on a real-world environmental analysis of the 2023 Turkey earthquake [38].

Our main contributions are as follows: (1) We introduce Geo-OLM, a geospatial prompting scheme that models EO workflows as state machines, enabling enhanced control and efficiency for EO tasks using smaller OLMs. (2) We conduct an extensive study across major LLM serving runtimes (e.g., Ollama, OpenAI) and model families (Phi, Qwen, LLaMA), offering practitioner guidelines for optimizing geospatial agentic prompting. (3) We evaluate our approach on both single- and multi-agent tasks and report on both monetary/cloud costs and task success rates, providing a holistic view of agentic performance. We will release our code and prompts upon acceptance to facilitate further research and adoption. We hope that our method will serve as a valuable resource for geoscientists and practitioners seeking cost-efficient and sustainable agentic practices for EOw studies.

2 Related work

Geospatial Foundation Models (GFMs). There is rapid progress in applying generative AI advances to EO tasks [43], with recent works demonstrating their potential across various domains, including satellite vision [6, 18, 19, 26, 30, 39, 58], urban analysis [2, 55], map-based applications [56], agriculture[3, 22, 25, 32, 53], and forestry [51, 60]. However, these approaches typically focus on a narrowly defined scope, where a fine-tuned LLM is trained for a specific task within a predetermined workflow [18, 58]. While effective, this approach is akin to deploying task-specific computer vision models, requiring extensive labeled data and computational resources for fine-tuning. In contrast, there is growing interest in more generalizable, fine-tuning-free solutions with LLMs.

Geospatial agents: A promising approach for geospatial AI is function-calling via LLMs, where an agent (tool-augmented LLM) interprets natural language user queries and selects the appropriate tools to complete a task. Recent geospatial copilots [4, 41] extend this capability to multi-scope EO tasks, integrating various APIs to support complex, long-horizon workflows. However, these systems centralize the prompting, decision-making, reasoning, and tool execution logic, making them reliant on large-scale proprietary models to function effectively [24]. This issue is even more pronounced in generalist frameworks, where the focus is on maximizing performance at the expense of efficiency. For instance, compositionbased [29] and ledger-based scheduling [12] paradigms require extensive message passing and iterative reasoning across multiple LLM calls to converge on a task. These computationally expensive techniques often make state-of-the-art methods cost-prohibitive for deployment in sustainability and climate studies.

Open Language Models. In the last few weeks alone, we have witnessed rapid advances in smaller open and open-weights models that train faster, are optimized for low latency [33], and increasingly match larger proprietary models in coding tasks and tool-calling workflows [52]. Breakthroughs such as the Phi model family [1] and DeepSeek models [15] demonstrate impressive potential toward

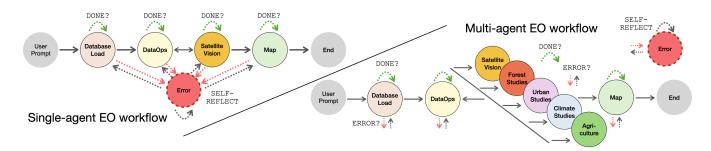


Figure 2: State-driven LLM reasoning with Geo-OLM. Earth observation (EO) workflows exhibit inherent progression logic, which we encapsulate using the StateFlow paradigm [50]. We illustrate this first in a single-agent scenario for satellite detection in remote sensing (RS) tasks (left) and extend it flexibly to a multi-agent setup incorporating diverse EO applications (right). By structuring reasoning into explicit states, Geo-OLM reduces the cognitive burden on low-resource OLMs, allowing them to better reflect on task progression, correctly handle errors, and prevent premature function termination.

closing the performance-cost gap. Similarly, the open-source ecosystem for OLM deployment is maturing rapidly, with community-driven initiatives like vLLM [21, 45] and more commercially focused platforms like LangChain [23] and Ollama [34].

However, these platforms remain primarily focused on generalist tasks, making them not directly applicable to workflows requiring domain expertise. Additionally, the lack of a widely established testbed for geospatial agentic tasks makes it difficult to compare model families or track improvements between older and newer models [57]. Similar discrepancies in evaluation methodologies have been noted in other domains, such as biomedical agentic applications [28]. In this work, we consider these challenges by conducting an in-depth evaluation across a wide range of the latest OLM variants from all major families.

3 Methodology

3.1 Background: State-driven LLM reasoning

In state-based LLM reasoning, the task-solving process is conceptualized as a finite state machine (FSM) [46], albeit a simplified version that depends on context history rather than an explicit input tape [50]. In LLMs, this context is inherently provided by the chat history of agent-user interactions (i.e., the messages = [. .] variable in LLM APIs) which accumulates as the execution progresses. By adapting an FSM-like notation, a workflow can be formulated as a tuple $\langle S, \delta, \Gamma \rangle$, where: (1) S represents the set of states, each corresponding to a distinct stage in the workflow, typically linked to executing a specific sub-task (e.g., loading data, applying filters, generating visualizations). (2) δ defines state transitions, determining the next state based on the current execution context and message history. (3) Γ denotes the sequence of messages exchanged during execution, including prompts, tool definitions, invocations, and responses.

More concretely, over n LLM API calls, Γ consists of the initial task prompt P, agent instructions $I = \{I_1, I_2, \ldots, I_n\}$, tool definitions $T = \{T_1, T_2, \ldots, T_n\}$ specifying the tools available at each stage, LLM responses $R = \{R_1, R_2, \ldots, R_n\}$, and tool return messages $O = \{O_1, O_2, \ldots, O_n\}$. In chat-based LLM APIs, these components have dedicated roles: system (task prompt P), user (instructions I),

tools argument (tool definitions *T*), assistant (LLM responses *R*), and the recently introduced tool/ipython (tool outputs *O*).

State transitions δ operate as a mapping function $(\delta: S \times \Gamma \to S)$, determining the next state based on the current state and accumulated message history. In practice, the prompt engineer defines the states, as well as the available tools per execution states, the expected actions, and tool invocations directly into LLM prompts. Execution logic is implemented as a combination of simple heuristics (e.g., programmatically verifying SQL query execution before proceeding) and LLM-based reasoning.

3.2 State-driven EO workflows

We begin by identifying a representative workflow for EO tasks, specifically defining the states (S) and state transitions (δ). We use Remote sensing (RS) tasks from the GeoLLM-Engine benchmark to illustrate the process (we generalize to other EO tasks in the following Subsection). GeoLLM-Engine consists of multi-round tasks, such as: "From the xView1 satellite imagery, run vessel detection and plot all ships from the Port of Istanbul in May 2020."

Overall flow. As shown in Figure 2 (left), the process in such an EO workflow begins with loading the specified satellite product from the SQL database. Next, the data is filtered based on user criteria (e.g., temporal and spatial constraints such as date ranges and geographic coordinates). The agent then executes the detection model on the satellite imagery. Depending on the task requirements, additional refining steps may follow – for instance, filtering detections based on specific object categories. Finally, the system renders the detections onto a map and returns the results to the user.

States. To structure execution within the LLM, we define the state sequence for single-task EO queries as Init \rightarrow Load \rightarrow Filter \rightarrow Detect \rightarrow Map \rightarrow End. In each state, the model is instructed to perform a set of recommended actions corresponding to its designated stage in the workflow.

LLM prompting. The LLM is first provided with an explicit definition of the workflow, outlining the structured progression of states. Then, at each interaction step, the model is instructed to infer and explicitly output its current execution stage (*S*) based on the accumulated messages history. This is implemented by requiring the

LLM to append CURRENT_STAGE = ... to its response, ensuring that the model continuously self-identifies its position in the workflow.

Per-state tools. At each state, we associate the set of tools that are relevant to completing the corresponding step. Tool definitions are appended to the model input (via the tools API argument) and the LLM responds with the selected function calls to execute. While this follows the standard LLM function-calling mechanism, our state-driven approach introduces a key efficiency gain: since the LLM explicitly tracks its execution state, we can dynamically suggest tools relevant for each stage. For example, if the CURRENT_STAGE is identified as Load, the agent is programmatically provided with few-shot examples related to database query functions. This selective prompting significantly reduces decision complexity, as also motivated in more elaborate dynamic tooling schemes [27, 37, 42], as it is beneficial for smaller OLMs which might struggle when presented with a large set of tool choices.

Transitions. As in standard LLM execution, transitions between workflow stages are LLM-driven, where the model infers the appropriate next step based on the accumulated message history and updates CURRENT_STAGE accordingly. However, conceptualizing the workflow as a state machine introduces two key advantages that can further reduce the reasoning burden on smaller OLMs: structured error handling and task completion validation.

For *error* handling, we introduce a dedicated Error state to manage failures. Instead of treating execution errors as unstructured chat history (as in conventional function-calling pipelines), statedriven reasoning enables programmatic transitions to the Error state when an exception is detected. Upon entering this state, the system issues a standalone LLM call with explicit SELF-REFLECT instructions, prompting the model to reconsider its previous actions and identify potential errors. To assist in debugging, the LLM is provided with details of the last function call, execution results, and the corresponding tool definitions, allowing it to *self-correct* minor mistakes such as typos or incorrect parameter usage.

For task *completion*, we introduce a termination safeguard. This is done by instructing the LLM to append the keyword TERMINATE when it determines that the task is complete. This signal is programmatically parsed, and upon detection, a follow-up validation step is issued: the model is explicitly asked to reason and confirm whether execution has indeed concluded by summarizing its current output. This mechanism helps prevent premature exits or infinite loops due to incorrect state reasoning or erroneous function calling.

3.3 Expanding to other EO workflows

State-based reasoning extends naturally to multi-agent flows [50], enabling broader geospatial applications beyond the single-task RS examples discussed above. We expand the state sequence definitions to enumerate the different possible agentic functionalities in the LLM prompt – as different paths shown in Figure 2 (right). This is implemented by requiring the LLM to interpret the original user task and append USER_INTENT = ... to its response. The list of possible intents is flexibly adapted to the workflow of interest. In our study, we consider GeoLLM-Squad [24], a multi-scope geospatial designed for diverse satellite-based analytics for urban monitoring, agriculture, forest management, and climate analysis, and satellite detections (see next Section for benchmark details). To this end, we

ask the LLM to reason and select from Vision (Detect), Forest, Urban, Climate, and Agriculture.

4 Geo-OLM framework

Our Geo-OLM implementation consists of two main components: the agentic frontend, which handles all interactions between the user and the agent, and the LLM API serving backend, responsible for executing model queries.

Agentic frontend. We build our frontend by customizing agentic routines from AutoGen [49]. As is common in agentic AI evaluations, we adopt a command-line sandbox implementation to ensure controlled execution, particularly when interacting with external tools such as web APIs or SQL databases. In this setting, the system records agent actions and executes the corresponding API calls in isolation, rather than requiring a full UI-driven workflow [7].

This design significantly simplifies implementation, allowing us to focus on developing the API toolset necessary to solve GeoLLM benchmarks without UI integration overhead, while maintaining full execution fidelity [20]. For function calling, we compile the list of functionalities and respective API tools required to solve the considered benchmarks [24, 41]. We reimplement the logic (the specific tools, e.g., loading xview1 data) and we confirm correctness by closely matching metrics reported in the baselines used in our comparisons.

LLM serving backend. A key component of our implementation is the LLM runtime engine, which enables local execution of open LMs – an aspect often overlooked in existing geospatial agents that rely solely on GPT-based APIs. We integrate the Ollama [34] LLM serving framework to support both text-based and chat-based model inference. For models that do not natively support tool execution within these APIs, we follow the official schema recommendations, specifying regex-based parsing to extract function calls based on tool role definitions. Otherwise, we leverage the built-in API schemas directly.

Benchmarks. To evaluate Geo-OLM, we conduct experiments on two benchmarks, namely GeoLLM-Engine [41] and GeoLLM-Squad [24], which cover realistic EO tasks and different user workflows over millions of satellite images and large-scale remote sensing products: (1) GeoLLM-Engine consists of single-tasks, multistep workflows for object detection, land cover classification, and visual question answering (VQA) with satellite imagery. The benchmark integrates open-source satellite imagery datasets with globally distributed spatiotemporal metadata, requiring agents to reason over specific locations and time periods. (2) GeoLLM-Squad covers a wider scope with multi-round EO workflows requiring interactions across different data modalities based on satellite products for urban analysis, agriculture and forest monitoring, and climate-related retrievals. Both benchmarks involve open-source datasets such as built-up area and population density mapping from Sentinel-2 imagery [16], and vegetation indices or surface temperature data from NASA MODIS Terra products [11, 44, 47]. We standardize all products, including HDF and GeoTIFF formats, as DataFrame variables, facilitating seamless integration into our framework.

Case study. Unlike existing works that focus exclusively on curated benchmarks, we extend our evaluation to real-world scenarios. To evaluate how Geo-OLMs generalize beyond predefined datasets,

we replicate the analytical workflow of the February 2023 Turkey earthquake study [38]. Using the open-source satellite-based building damage assessments and population estimates, we prompt Geo-OLMs to follow the original work's methodology via state-driven reasoning, assessing their agentic ability to reach the originally reported findings within reasonable fidelity.

5 Experimental Setup

Model families. To our knowledge, this is the first geospatial agentic study to systematically profile both proprietary and open LLMs to this extent, as we consider numerous major tool-supporting APIs available in Ollama as of January 2025. For proprietary models, we use OpenAI's latest GPT-40, GPT-40-mini, and GPT-01, while for open models we report results for the LLaMA-3.3 and 3.2 series (70B, 3B, 1B) [10], the Qwen-2.5 family (72B down to 0.5B) [52], Mistral (Small 14B) [33], and the latest Phi-4 (14B) [1].

LLM baselines. We evaluate both generic and geospatial-specific LLM prompting methods. For comparison purposes, we reimplement the GeoLLM-Engine and GeoLLM-Squad solvers within our framework, which use ReAct [54] and Orchestrator [12] prompting, respectively. Moreover, we compare with Chameleon [29], a compositional scheduling method, and Magentic, a generalist multi-agent framework [12]. We confirm our implementations closely match reported values from prior work.

Metrics. For performance on EO tasks, we report metrics as in the baseline benchmarks, e.g., F1 scores for object detection tasks. To assess agentic execution, we adopt widely-used evaluation practices from WebArena [9, 20, 59], computing *success* and *correctness* rates. Success rate measures the proportion of fully completed tasks, regardless of intermediate errors, while correctness rate quantifies the fraction of correct function calls with the correct parameters executed in the expected order.

Testbed. GPT experiments use the OpenAI API, while all OLM evaluations are run on a dedicated GPU cluster equipped with 8 A series GPUs (40GB each). To ensure consistent cost measurements and reduce potential variability from network delays, we store all datasets locally on disk. We compute the average time and tokens per query (from initial user question to completion) over the entire benchmark (2k). All software dependencies use the latest available versions as of January 2025. For Ollama, we use the default Q4 quantized models.

Cost model. We report the total cost of executing the full benchmark (2K queries). For GPT models, we log the exact breakdown of input, cached, and output tokens from the OpenAI API and compute costs using official pricing rates [35]. For OLMs running locally, we estimate costs based on market-rate GPU pricing. Our setup is closest to the Azure NC48ads series [31], with an estimated rate of \$10 per hour. For representative comparison with a real-world deployment (where the requests will be batched over a server), we amortize costs by the maximum number of models that can fit concurrently in the GPUs based on Ollama memory usage. We compute OLM costs by multiplying this amortized hourly rate by the profiled per-query inference time.

6 Results

GPT Models. To assess the reasoning effectiveness of state-driven prompting, we first compare Geo-OLM against existing GPT-based geospatial methods with the latest GPT-40 and o1 variants (Table 1). Among the GPT-40 models, Geo-OLM achieves agentic rates within 1% from the strongest prior work, GeoLLM-Squad, and within 2-4% of the near-oracle GPT-01, while reducing costs by an order of magnitude (\$77.02 vs. \$833.91). Similar trends hold for GPT-40-mini, where Geo-OLM achieves an 85.4% success rate, within 2.3% of GeoLLM-Squad, but at a 2.5 × lower cost. In terms of EO metrics, Geo-OLM GPT-40 has lower error rates across most applications, while performance with GPT-40-mini is stronger for satellite detections and within 2% of GeoLLM-Engine ϵ rates. Last, confirming our implementation, the correctness rates for the baseline methods with GPT-40-mini are within 2% of previously reported values [24].

OLMs - Qwen family. Starting with the latest Qwen-2.5 variants (Table 2), we evaluate agentic performance beyond proprietary LLMs. We observe that Geo-OLM achieves a better performance-cost trade-off across different models sizes (from 72B down to 3B). For the stronger variant (Qwen-2.5-72B), Geo-OLM achieves rates within 1–2% of its GPT-4o-mini counterpart while running at comparable cost, demonstrating the feasibility of sustained geospatial analysis without reliance on opaque proprietary models.

Prior geospatial agents with OLMs. From Table 2, we note how quickly agentic performance deteriorates as we move to low-resource OLMs. Both GeoLLM-Engine and GeoLLM-Squad experience a 30–40% drop in completion rates at 7B, while Geo-OLM remains consistently higher by up to 33%. Since we have previously verified our implementations by matching reported GPT-based metrics, we hypothesize that this degradation in existing geospatial agents stems from how prior work handles function calling with OLMs. Specifically, both methods report direct integration with text-based API logic rather than utilizing chat-based support (tools argument in HuggingFace/Ollama).

To investigate further, we conduct an in-depth failure analysis. We find that lower success rates in existing works primarily result from models exiting prematurely, while correctness errors often arise from incorrect tool argument formatting. For example, function calls with OLMs frequently fail due to minor syntax errors, such as passing startdate instead of start_date. Additionally, the reliance on text-based logic introduces JSON parsing schemas on top of the model output, contributing further to typos and misformatting errors. These findings indicate that addressing these failures should improve performance.

Prior work with our self-reflect logic. To this end, we augment our baseline implementations by applying our chat-based logic (accumulating messages over task progression and chat APIs) along with error correction and termination checks. We refer to these improved baselines as "our +ERR/TRM" in Table 2. These enhancements incorporate all the prompting improvements of our method except for state-driven reasoning. Indeed, we observe that across nearly all models, these refinements lead to higher success rates, bringing performance closer to Geo-OLM. However, GeoLLM-Engine continues to lag, while GeoLLM-Squad remains more expensive due to its complex orchestration logic.

Model	Geospatial Agent	Chat API	Correct. Rate%	Success Rate%	Tokens Avg (k)	Total Cost	$\begin{array}{c} \textbf{Agro} \\ \epsilon \% \end{array}$	$\begin{array}{c} \textbf{Climate} \\ \epsilon\% \end{array}$	Urban $\epsilon \%$	Forest $\epsilon\%$	Vision R%
GPT-o1	GeoLLM-Squad	✓	90.97	97.2	77.36	\$833.91	0.14	0.58	1.39	0.56	98.71
	GeoLLM-QA	✓	52.49	62.3	22.78	\$68.23	5.34	3.97	6.33	6.73	72.80
	GeoLLM-Engine	\checkmark	74.11	82.6	22.25	\$64.72	5.05	2.26	8.31	7.85	87.71
CDT 4a	Chameleon	\checkmark	21.87	55.1	36.19	\$115.08	12.51	9.14	9.62	10.77	50.42
GPT-40	Magentic	\checkmark	25.40	79.7	117.65	\$447.55	83.89	80.63	80.86	83.13	97.25
	GeoLLM-Squad	\checkmark	85.32	96.2	77.49	\$219.67	3.70	2.02	1.99	3.33	97.59
	Geo-OLMs	✓	86.03	95.1	30.44	\$77.02	3.44	1.17	4.40	2.51	96.26
	GeoLLM-QA	✓	37.41	53.1	21.55	\$3.99	13.05	10.29	14.61	12.82	69.16
	GeoLLM-Engine	\checkmark	53.56	82.8	20.09	\$3.63	11.45	8.69	8.88	8.59	60.33
GPT-4o-mini	Chameleon	\checkmark	34.63	68.7	60.10	\$11.17	12.25	8.05	8.85	8.84	67.42
	Magentic	\checkmark	18.85	73.1	187.45	\$38.33	13.44	11.45	11.54	12.29	75.51
	GeoLLM-Squad	\checkmark	59.46	87.7	72.98	\$12.63	16.42	9.70	10.22	10.19	70.58
	Geo-OLMs	√	58.75	85.4	28.49	\$4.96	15.66	10.45	9.75	8.63	75.49

Table 1: GPT models: Agentic correctness and success rates with cost and downstream EO task performance [24].

Other generalist baselines. Chameleon remains reasonably competitive in task completion rates, but suffers from high execution costs due to its compositional orchestration overhead and multiple cross-agent communication rounds. Meanwhile, Magentic fails entirely, with completion rates dropping below 5% for all OLMs, preventing us from collecting meaningful performance metrics. This is consistent with cautionary notes from its developers, who tested it primarily against GPT-40 [12, 49]. This highlights a broader limitation of applying generalist approaches to geospatial and sustainability studies – while effective for proprietary LLMs, they often fail to generalize as plug-and-play solutions for more complex and nuanced tasks, particularly with smaller OLMs.

OLM vs. GPT trends. While performance remains competitive at larger scales, the trade-offs become more pronounced as we scale down below 32B. In the 7B–14B range, success rates drop by approximately 20%, but at a 10× and 100× cost reduction compared to GPT-4o-mini and GPT-4o, respectively. As a helpful consideration for practitioners, we note that all models collapse below 3B parameters – a trend consistent across model families beyond Owen.

Correctness vs. Tokens Finally, an interesting trend emerges with respect to token consumption: smaller models often use more tokens per task than larger ones. This is a direct result of lower correctness rates, causing agents to repeatedly attempt function calls, leading to excessive retries and inflated token costs. Across all methods and model families, we observe a correlation between lower correctness rates and higher per-task token usage. As motivation for future work, we hypothesize that improving reasoning for better correctness rates (e.g., tool selection) would reduce token overhead, thereby lowering latencies and further reducing costs.

Other OLM families - Llama. We present results for the latest variants of LLaMA, Mistral, and Phi, with most models introduced in recent weeks (Table 3). For LLaMA 70B, we observe that adding more logic sometimes has the opposite effect, especially with the self-reflection mechanism which surprisingly causes the model

to question itself and abandon tasks prematurely. Even more surprisingly, the model performs better without chat API prompting, instead favoring direct text-based parsing, which aligns with reported community findings (e.g., vLLM library [45]).

Mistral and Phi. The recently released Mistral appears to handle chat-based and less complex logic more effectively, with both chat API usage and direct text-based parsing working comparably well. However, even with error correction, it struggles with the orchestration burden from GeoLLM-Squad. Meanwhile, Phi demonstrates robust performance but exhibits higher overall token consumption, with noticeably more verbose outputs – likely due to its reasoning-focused development and pretraining [1].

OLM and EO tasks. As expected, we observe a correlation between better agentic performance and improved EO metrics (lower error rates and higher detection recall). Across nearly all model families – with the exception of smaller 3B variants and Mistral – Geo-OLMs sustain among the best EO metrics. Finally, we note that detection tasks tend to be more robust to errors, whereas lower success rates have a greater impact on EO product predictions.

Discussion. As captured in Figure 3, we compare geospatial agent performance across a wide range of proprietary and open language models, spanning scales from 72B down to 3B parameters. Geo-OLMs consistently achieve better trade-offs between performance and cost, maintaining competitive agentic success and correctness rates at significantly lower computational costs. Unlike existing methods, which degrade sharply on smaller OLMs, Geo-OLMs leverage structured state-driven reasoning to mitigating failures such as erroneous tool selection or premature task exits. These results demonstrate that state-driven prompting is an effective strategy for adapting lightweight, community-accessible models to complex geospatial workflows without the prohibitive costs of proprietary LLMs.

Cost reduction considerations. Lower dollar costs directly correspond to reduced runtime per query. That is, within the same model family, a cost reduction by a certain factor corresponds to the

Table 2: Qwen-2.5 models: Agentic correctness and success rates with cost and downstream EO task performance [24].

Model	Geospatial Agent	Chat API	Correct. Rate%	Success Rate%	Tokens Avg (k)	Total Cost	Agro $\epsilon\%$	Climate $\epsilon\%$	Urban ε %	Forest $\epsilon\%$	Vision R%
	GeoLLM-Engine (GE)		45.06	81.3	26.43	\$8.69	22.08	20.73	20.05	19.14	96.24
	GE+Our ERR/TRM	✓	51.32	84.9	29.39	\$5.79	13.06	11.65	8.80	8.93	85.69
0	Chameleon	✓	25.74	79.2	99.36	\$36.48	16.13	12.89	11.52	10.45	46.39
Qwen-2.5-72B	GeoLLM-Squad (GS)		45.36	77.2	85.28	\$58.83	32.44	29.28	29.93	28.78	96.68
	GS+Our ERR/TRM	\checkmark	56.39	85.4	82.34	\$53.40	11.28	11.01	10.96	9.68	92.67
	Geo-OLMs	✓	57.52	85.9	33.22	\$6.01	11.84	11.27	10.65	9.71	94.53
	GeoLLM-Engine (GE)		46.48	65.6	27.11	\$1.26	13.77	9.84	10.36	10.70	91.24
	GE+Our ERR/TRM	\checkmark	48.24	73.6	25.34	\$1.71	14.06	10.67	9.91	9.16	87.60
O	Chameleon	\checkmark	23.77	65.9	99.13	\$9.45	15.37	10.27	10.00	8.45	75.96
Qwen-2.5-32B	GeoLLM-Squad (GS)		52.40	69.7	76.91	\$10.39	13.83	10.00	10.69	9.46	88.75
	GS+Our ERR/TRM	\checkmark	56.03	75.4	81.58	\$13.64	13.78	10.15	9.95	8.74	88.28
	Geo-OLMs	✓	55.91	75.1	29.71	\$1.63	13.63	10.62	10.21	9.25	86.67
	GeoLLM-Engine (GE)		31.93	26.2	29.68	\$0.35	39.32	35.83	31.71	29.73	82.13
	GE+Our ERR/TRM	\checkmark	37.97	63.8	31.37	\$0.46	17.92	13.33	11.84	11.29	80.26
Owen-2.5-14B	Chameleon	\checkmark	22.65	68.2	106.13	\$2.38	16.21	12.90	11.38	11.12	86.86
Qweii-2.5-14D	GeoLLM-Squad (GS)		42.99	54.4	86.22	\$2.63	39.91	33.88	29.93	28.89	90.41
	GS+Our ERR/TRM	✓	44.09	67.4	89.43	\$3.47	19.90	15.13	13.56	13.90	81.71
	Geo-OLMs	\checkmark	42.79	67.4	37.08	\$0.58	19.43	15.05	13.51	14.16	87.86
	GeoLLM-Engine (GE)		19.84	30.5	55.52	\$0.23	50.44	45.62	43.39	41.66	72.16
	GE+Our ERR/TRM	\checkmark	26.75	56.2	51.02	\$0.23	19.42	15.13	14.39	13.90	48.27
Owen-2.5-7B	Chameleon	\checkmark	14.64	34.6	99.92	\$0.72	98.37	94.69	94.77	93.57	97.59
Qwen-2.5-7B	GeoLLM-Squad (GS)		33.96	46.2	149.54	\$1.49	31.00	27.50	25.82	22.96	51.91
	GS+Our ERR/TRM	\checkmark	36.63	60.5	153.21	\$2.74	15.86	13.00	12.64	11.22	66.36
	Geo-OLMs	✓	40.90	63.3	40.77	\$0.15	15.79	13.00	12.59	10.50	59.98
	GeoLLM-Engine (GE)		9.42	11.8	52.01	\$0.07	103.08	97.25	89.65	87.08	89.55
	GE+Our ERR/TRM	\checkmark	19.26	24.9	45.84	\$0.06	23.59	20.49	19.36	18.96	49.85
Owen 25 2D	Chameleon	\checkmark	9.09	35.1	108.50	\$0.28	17.43	13.82	12.05	12.21	52.58
Qwen-2.5-3B	GeoLLM-Squad (GS)		7.94	12.6	146.31	\$0.42	92.05	87.29	82.95	82.28	92.92
	GS+Our ERR/TRM	\checkmark	25.63	19.5	149.85	\$0.48	23.33	19.47	18.72	18.46	74.72
	Geo-OLMs	✓	26.26	22.8	50.87	\$0.05	28.97	25.62	24.10	23.91	80.89

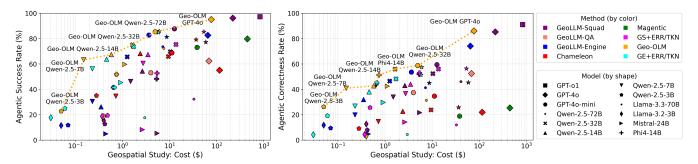


Figure 3: Agentic success and correctness rates vs. benchmarking cost (for a typical benchmark size of 2K user queries) across various models (distinguished by markers) and prompting techniques (distinguished by colors). Overall, Geo-OLM achieves better performance-cost trade-offs across model scales from 72B down to 3B parameters, sustaining agentic performance at reduced computational expenses (best viewed in color).

Table 3: LLaMA, Mistral, and Phi models: Agentic correctness and success rates with cost and downstream EO performance [24].

Model	Geospatial Agent	Chat API	Correct. Rate%	Success Rate%	Tokens Avg (k)	Total Cost	$\begin{matrix} \textbf{Agro} \\ \epsilon\% \end{matrix}$	Climate $\epsilon\%$	Urban ε %	Forest $\epsilon\%$	Vision R%
	GeoLLM-Engine (GE)		41.33	73.6	28.45	\$7.19	15.13	13.12	12.28	11.62	82.03
	GE+Our ERR/TRM	\checkmark	4.37	17.7	39.24	\$2.97	60.26	54.45	47.30	44.85	63.95
I lama 2 2 70P	Chameleon	\checkmark	10.54	10.0	_	_	_	_	_	_	_
Llama-3.3-70B	GeoLLM-Squad (GS)		46.02	73.3	36.60	\$40.79	24.54	21.38	21.27	20.67	88.09
	GS+Our ERR/TRM	\checkmark	11.95	32.3	126.60	\$32.80	26.50	22.86	21.35	21.09	33.33
	Geo-OLMs	✓	31.30	72.3	81.73	\$7.57	13.70	8.92	11.17	9.94	84.00
	GeoLLM-Engine (GE)		11.78	11.5	40.84	\$0.05	32.78	29.43	27.76	27.38	48.84
	GE+Our ERR/TRM	\checkmark	4.26	17.7	30.75	\$0.03	22.31	19.02	17.48	16.51	17.86
Llama-3.2-3B	Chameleon	\checkmark	0.70	5.9	_	_	-	_	_	_	_
Liailia-3.2-3D	GeoLLM-Squad (GS)		12.65	15.4	68.80	\$0.40	41.03	36.79	33.77	33.48	61.48
	GS+Our ERR/TRM	\checkmark	4.42	18.7	72.80	\$0.37	31.54	28.21	26.67	25.84	37.21
	Geo-OLMs	✓	3.17	18.5	75.18	\$0.40	30.26	27.18	25.64	26.41	39.47
	GeoLLM-Engine (GE)		4.73	4.9	13.67	\$0.44	75.17	70.78	66.21	65.00	86.54
	GE+Our ERR/TRM	\checkmark	29.71	45.4	40.97	\$0.92	25.75	22.49	21.44	22.36	61.25
Mistral-24B	GeoLLM-Squad (GS)		21.60	5.4	98.69	\$5.44	30.00	25.79	24.21	25.00	83.62
	GS+Our ERR/TRM	✓	14.16	8.2	68.54	\$2.63	23.85	20.62	19.33	19.85	59.81
	Geo-OLMs	✓	42.55	59.7	61.29	\$1.12	11.94	9.90	9.97	10.35	29.12
	GeoLLM-Engine (GE)		43.73	41.5	36.49	\$0.64	16.88	13.62	12.56	13.00	94.14
	GE+Our ERR/TRM	\checkmark	45.25	41.5	41.09	\$0.67	18.91	15.74	15.18	13.91	91.80
Phi4-14B	GeoLLM-Squad (GS)		51.93	48.2	101.91	\$5.32	21.93	18.97	18.27	18.07	89.72
	GS+Our ERR/TRM	\checkmark	52.88	52.6	121.73	\$5.43	18.93	15.80	14.31	14.47	85.26
	Geo-OLMs	✓	51.30	51.8	54.72	\$0.75	21.79	18.47	16.97	17.00	85.19

runtime being lower by the same factor, reflecting more efficient execution. Across different model families, smaller models benefit from higher parallelization and lower inference latency. We note that it is meaningful to include more hardware-specific criteria [17], such as latency constraints (e.g., data retrieval delays from disk or cloud storage) as well as energy and power efficiency. As an important direction for future work, we motivate detailed hardware profiling beyond monetary cost, particularly across GPU clusters, edge devices, and local-remote execution setups. A simple extension we are currently investigating is replacing Q4-quantized variants with their full-precision counterparts (as used in Hugging Face baselines [48]). For instance, Q4, Q8, Q16, and full-precision variants are readily available in Ollama [34], allowing us to flexibly populate the performance-cost trade-off front.

Geo-OLMs in Remote sensing. In Table 4 we present the performance comparison with agentic results and downstream task metrics in remote sensing (RS) satellite-detection workflows [41]. Overall, the trends observed in the EO baseline benchmark persist. Geo-OLM maintains strong overall performance. In larger models, detection F1 scores remain particularly high, ranging from 80–90%, with success rates remaining high and exceeding 80%. As expected, downsizing models leads to gradual performance degradation, impacting both agentic reasoning and downstream detection accuracy. Geo-OLM remains competitive across most scales, with the same issues of lower performance persisting for Llama and Mistral models (with success rates dropping to less than 40%).

7 Case Study: Earthquake Impact

A possible limitation of state-driven prompting is its reliance on explicitly defined workflow states, raising an important question: does improving OLM performance come at the cost of generalizability? To evaluate this, we conduct a case study simulating how a geospatial analyst could apply our method to a real-world disaster impact assessment. We focus on the February 2023 Turkey earthquake, leveraging a previously released post-disaster satellite and population analysis study [38]. This study is an ideal test case for agentic reasoning, as it explicitly describes an analytical workflow with a step-by-step methodology. Crucially, it provides structured datasets (satellite-detected building damage and population density estimates) and tools [5], enabling programmatic verification of agentic performance with high fidelity by matching its findings.

Methodology. To enable agentic evaluation, we convert the study's datasets and documentation into a vector database, allowing for retrieval-augmented prompting of relevant information. Using GPT, we generate 120 representative question variants that a geospatial analyst might ask based on the study's findings. Consider a sample question: "How many buildings were damaged in Nurdăği on February 9?" or "What was the estimated impacted population in Islahiye on February 7?" These can be programmatically answered from the dataset, providing a direct ground truth for evaluating agentic correctness. This allows us to establish a gold-standard benchmark, by deriving expected answers and matching directly with the study's conclusions.

Table 4: Agentic performance in RS tasks [41] across different model families, evaluating correctness, success rates, and downstream task accuracy (Land-coverage classification accuracy and detection F1 scores).

Model	Geospatial Agent	Correctness Rate (%)	Success Rate (%)	LCC Acc (%)	Det F1 (%)
	GeoLLM-Engine	59.06	81.97	95.45	75.37
GPT-40	GeoLLM-Squad	86.04	84.70	96.28	86.90
	Geo-OLM	92.14	95.63	95.54	97.44
	GeoLLM-Engine	59.98	84.70	74.49	62.10
GPT-40-mini	GeoLLM-Squad	42.89	87.29	90.54	69.43
	Geo-OLM	64.54	86.89	85.30	76.06
	GeoLLM-Engine	45.75	85.12	94.72	95.96
Qwen-2.5-72B	GeoLLM-Squad	42.60	87.50	95.26	96.45
	Geo-OLM	54.59	84.70	91.25	92.28
	GeoLLM-Engine	54.13	62.84	95.14	90.67
Qwen-2.5-32B	GeoLLM-Squad	53.65	65.57	91.81	88.01
	Geo-OLM	55.64	73.22	94.80	85.80
	GeoLLM-Engine	35.84	33.33	96.36	81.23
Qwen-2.5-14B	GeoLLM-Squad	46.77	68.35	94.01	91.38
	GeoLLM-Squad	49.53	74.42	90.01	89.11
	Geo-OLM	41.94	65.75	92.61	88.89
	GeoLLM-Engine	16.51	50.38	79.70	69.93
Qwen-2.5-7B	GeoLLM-Squad	34.09	58.02	67.51	51.11
	Geo-OLM	37.25	59.89	67.79	71.91
	GeoLLM-Engine	7.25	20.45	70.83	63.75
Llama-3.3-70B	GeoLLM-Squad	40.59	68.48	97.20	88.45
	Geo-OLM	27.09	61.54	48.78	81.28
	GeoLLM-Engine	9.32	21.52	94.47	89.17
Llama-3.2-3B	GeoLLM-Squad	5.65	28.57	86.37	93.88
	Geo-OLM	22.40	25.93	95.05	80.95
	GeoLLM-Engine	8.56	15.24	83.16	85.87
Mistral-24B	GeoLLM-Squad	30.02	10.56	93.71	84.73
	Geo-OLM	34.26	36.61	59.95	88.85
	GeoLLM-Engine	48.05	52.20	93.45	94.72
Phi4-14B	GeoLLM-Squad	49.53	74.42	90.01	89.11
	Geo-OLM	48.96	72.41	93.22	85.20

Next, we generate agentic ground truths, formatted as sequences of tool-calling steps. We follow the Tool-QA methodology [61] and use GPT-40 as an oracular solver, providing both the sample question and the programmatically derived solution. The oracle executes API calls to retrieve results, recording step-by-step function calls and argument selections. We discard agentic solutions that do not match the programmatic solution, allowing for a 10% variance to account for rounding differences in tool calls (typically coordinates).

Finally, we formulate the task as a state-driven workflow tailored to this study, defining states as: Init \rightarrow Load \rightarrow Filter \rightarrow Detect \rightarrow Correlate (Building/Population) \rightarrow Plot/Answer \rightarrow End. The Correlation step reflects the study's implementation, where population estimates are spatially correlated with detected building

damage to quantify affected populations. We expand Geo-OLM with the tools for satellite-based damage detection and urban population estimation [5]. We convert the study data to DataFrames for frontend integration with our framework. For evaluation, we adopt the same success rate criteria as in the baseline benchmarks.

Results. We observe that trends persist in this case study (Table 5), with Geo-OLM maintaining good performance-cost trade-off across nearly all model versions. As model size decreases, correctness rates drop even for this simpler task, likely due to increased typos and argument formatting errors. However, success rates remain consistently high, reaching 100% in larger models, as these tasks follow a simpler single-scope workflow, unlike the more complex multi-agent, multi-scope flows in the baseline benchmark.

Table 5: Agentic performance across model families towards replicating the February 2023 Turkey earthquake impact study [38], evaluating correctness, success rates, and cost.

Model	Geospatial	Correct.	Success	Total
Model	Agent	Rate%	Rate%	Cost (120)
	GE+ERR/TRM	62.42%	100.0%	\$2.14
GPT-40	Chameleon	88.52%	100.0%	\$12.14
Gr 1-40	GS+ERR/TRM	95.92%	100.0%	\$9.99
	Geo-OLM	90.52%	100.0%	\$2.59
	GE+ERR/TRM	58.43%	100.0%	\$0.11
GPT-40-mini	GS+ERR/TRM	89.44%	100.0%	\$0.59
	Geo-OLM	80.72%	100.0%	\$0.28
	GE+ERR/TRM	58.12%	100.0%	\$0.24
Overage 2 5 72P	Chameleon	68.66%	98.3%	\$3.16
Qwen-2.5-72B	GS+ERR/TRM	80.94%	100.0%	\$2.31
	Geo-OLM	78.67%	100.0%	\$0.29
	GE+ERR/TRM	43.65%	96.7%	\$0.06
Owen-2.5-32B	Chameleon	44.56%	66.7%	\$0.54
Qweii-2.5-52b	GS+ERR/TRM	62.04%	96.7%	\$0.26
	Geo-OLM	62.59%	98.3%	\$0.08
	GE+ERR/TRM	40.11%	45.0%	\$0.02
Qwen-2.5-14B	GS+ERR/TRM	49.22%	98.3%	\$0.15
	Geo-OLM	43.16%	90.4%	\$0.06
	GE+ERR/TRM	35.67%	30.0%	\$0.01
Qwen-2.5-7B	GS+ERR/TRM	46.00%	36.7%	\$0.08
	Geo-OLM	50.40%	38.3%	\$0.02
	GeoLLM-Engine (GE)	38.23%	95.0%	\$0.20
Llama-3.3-70B	GeoLLM-Squad (GS)	75.35%	100.0%	\$1.49
	Geo-OLM	70.58%	100.0%	\$0.34
DI :4 4 4 D	GS+ERR/TRM	66.38%	50.0%	\$0.19
Phi4-14B	Geo-OLM	41.84%	65.0%	\$0.02

8 Limitations and Future Work

One key limitation of state-driven reasoning is its reliance on domain expertise to construct structured workflows. Designing effective prompts requires a clear understanding of the task and careful engineering of workflow states [50], which may introduce a barrier to adoption. However, flow-based tooling is already a common practice in LLM deployment, particularly within UI-driven frameworks such as AutoGen Studio [7, 49] and Azure Prompt Flow [8]. From this perspective, state-driven reasoning can be seen as an extension of existing deployment paradigms rather than an entirely new requirement. This also presents an opportunity for future work to explore human-computer interaction (HCI) practices for integrating structured reasoning into interactive agent development and enhancing interfaces beyond tool invocation.

Another consideration is the rapid improvement of smaller language models, which may eventually narrow the performance-cost gap. While this trend is undeniable, our results show that even commercial solutions do not yet achieve human-like performance in complex geospatial workflows. Since both proprietary and smaller open models are primarily optimized for general-purpose reasoning, there will continue to be a need for domain-specialized agents in fields such as geosciences [43] and biosciences [28]. In this context,

state-driven prompting remains valuable as a method to enhance execution reliability and reasoning efficiency, even as cost differences diminish over time.

9 Conclusion

We introduced Geo-OLM, a geospatial agentic framework based on state-driven prompting, which explicitly separates reasoning from tool execution. Through comprehensive evaluations on geospatial benchmarks and a domain-specific case study, we demonstrated that Geo-OLMs achieve competitive performance, staying within 10-20% of proprietary models while reducing costs by $10-100 \times 100$. These results highlight the viability of structured agentic workflows for cost-efficient, scalable geospatial AI.

References

- Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. 2024. Phi-4 technical report. arXiv preprint arXiv:2412.08905 (2024).
- [2] Prabin Bhandari, Antonios Anastasopoulos, and Dieter Pfoser. 2024. Urban Mobility Assessment Using LLMs. In Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems. 67–79.
- [3] Nikolaos Ioannis Bountos, Arthur Ouaknine, and David Rolnick. 2023. FoMo-Bench: a multi-modal, multi-scale and multi-task Forest Monitoring Benchmark for remote sensing foundation models. arXiv preprint arXiv:2312.10114 (2023).
- [4] Yuxing Chen, Weijie Wang, Sylvain Lobry, and Camille Kurtz. 2024. An Ilm agent for automatic geospatial data analysis. arXiv preprint arXiv:2410.18792 (2024).
- [5] Microsoft Corp. 2025. Global ML Building Footprints. https://github.com/microsoft/GlobalMLBuildingFootprints Accessed: 2025-01-30.
- [6] Philipe Dias, Aristeidis Tsaris, Jordan Bowman, Abhishek Potnis, Jacob Arndt, H Lexie Yang, and Dalton Lunga. 2024. OReole-FM: successes and challenges toward billion-parameter foundation models for high-resolution satellite imagery. In Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems. 597–600.
- [7] Victor Dibia, Jingya Chen, Gagan Bansal, Suff Syed, Adam Fourney, Erkang Zhu, Chi Wang, and Saleema Amershi. 2024. Autogen studio: A no-code developer tool for building and debugging multi-agent systems. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. 72–79.
- [8] Azure Documentation. 2024. What is Azure Machine Learning prompt flow? https://learn.microsoft.com/en-us/azure/machine-learning/prompt-flow/ overview-what-is-prompt-flow?view=azureml-api-2 Accessed: 2025-01-30.
- [9] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How Capable are Web Agents at Solving Common Knowledge Work Tasks?. In Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235). PMLR, 11642– 11662. https://proceedings.mlr.press/v235/drouin24a.html
- [10] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024).
- [11] Earth Science Data and Information System (ESDIS). 2025. NASA Earthdata Search. https://search.earthdata.nasa.gov. Accessed 2025-01-07.
- [12] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. 2024. Magentic-one: A generalist multi-agent system for solving complex tasks. arXiv preprint arXiv:2411.04468 (2024).
- [13] Global Fishing Watch. 2025. Global Fishing Watch. https://globalfishingwatch. org/. Accessed 2025-01-30.
- [14] Vinicius G Goecks and Nicholas R Waytowich. 2023. Disasterresponsegpt: Large language models for accelerated plan of action development in disaster response scenarios. arXiv preprint arXiv:2306.17271 (2023).
- [15] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025).
- [16] Matthew C Hansen, Peter V Potapov, Rebecca Moore, Matt Hancher, Svetlana A Turubanova, Alexandra Tyukavina, David Thau, Stephen V Stehman, Scott J Goetz, Thomas R Loveland, et al. 2013. High-resolution global maps of 21st-century forest cover change. science 342, 6160 (2013), 850–853.
- [17] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024.

- ROUTERBENCH: A Benchmark for Multi-LLM Routing System. arXiv preprint arXiv:2403.12031 (2024).
- [18] Yuan Hu, Jianlong Yuan, Congcong Wen, Xiaonan Lu, and Xiang Li. 2023. Rsgpt: A remote sensing vision language model and benchmark. arXiv preprint arXiv:2307.15266 (2023).
- [19] Yanan Jian, Fuxun Yu, Simranjit Singh, and Dimitrios Stamoulis. 2023. Stable diffusion for aerial object detection. arXiv preprint arXiv:2311.12345 (2023).
- [20] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In ICLR 2024 Workshop on Large Language Model (LLM) Agents.
- [21] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles. 611–626.
- [22] Alexandre Lacoste, Nils Lehmann, Pau Rodriguez, Evan Sherwin, Hannah Kerner, Björn Lütjens, Jeremy Irvin, David Dao, Hamed Alemohammad, Alexandre Drouin, et al. 2024. Geo-bench: Toward foundation models for earth monitoring. Advances in Neural Information Processing Systems 36 (2024).
- [23] LangChain. 2024. LangChain Documentation. https://python.langchain.com/ docs/introduction/ Accessed: 2025-01-30.
- [24] Chaehong Lee, Varatheepan Paramanayakam, Andreas Karatzas, Yanan Jian, Michael Fore, Heming Liao, Fuxun Yu, Ruopu Li, Iraklis Anagnostopoulos, and Dimitrios Stamoulis. 2025. Multi-Agent Geospatial Copilots for Remote Sensing Workflows. arXiv:2501.16254
- [25] Jiajia Li, Kyle Lammers, Xunyuan Yin, Xiang Yin, Long He, Renfu Lu, and Zhaojian Li. 2024. MetaFruit Meets Foundation Models: Leveraging a Comprehensive Multi-Fruit Dataset for Advancing Agricultural Foundation Models. arXiv preprint arXiv:2407.04711 (2024).
- [26] Fan Liu, Delong Chen, Zhangqingyun Guan, Xiaocong Zhou, Jiale Zhu, Qiaolin Ye, Liyong Fu, and Jun Zhou. 2024. RemoteCLIP: A Vision Language Foundation Model for Remote Sensing. IEEE Transactions on Geoscience and Remote Sensing 62 (2024), 1–16.
- [27] Yanming Liu, Xinyue Peng, Yuwei Zhang, Jiannan Cao, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. 2024. Tool-Planner: Dynamic Solution Tree Planning for Large Language Model with Tool Clustering. arXiv preprint arXiv:2406.03807 (2024).
- [28] Simon Lobentanzer, Shuhan Feng, Nicole Bruderer, et al. 2025. A platform for the biomedical application of large language models. *Nature Biotechnology* (2025). doi:10.1038/s41587-024-02534-3
- [29] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2024. Chameleon: Plug-and-play compositional reasoning with large language models. Advances in Neural Information Processing Systems 36 (2024).
- [30] Utkarsh Mall, Cheng Perng Phoo, Meilin Kelsey Liu, Carl Vondrick, Bharath Hariharan, and Kavita Bala. 2024. Remote Sensing Vision-Language Foundation Models without Annotations via Ground Remote Alignment. In The Twelfth International Conference on Learning Representations.
- [31] Microsoft Azure. 2025. Azure Virtual Machines ND A100 v4 Series. https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/gpu-accelerated/nd-family#nd a100 v4-series Accessed: 2025-01-30.
- [32] Microsoft Industry Clouds. 2023. Evolving Microsoft Azure Data Manager for Agriculture to transform data into intuitive insights. https://azure.microsoft.com/en-us/blog/evolving-microsoft-azure-data-manager-for-agriculture. Accessed: 2025-01-07.
- [33] Mistral AI Team. 2025. Mistral Small 3. https://mistral.ai/news/mistral-small-3/ Accessed 2025-01-30.
- [34] Ollama. 2025. Get up and running with large language models. https://github.com/ollama/ollama. Accessed 2025-01-30.
- [35] OpenAI. 2025. OpenAI API Pricing. https://openai.com/api/pricing/ Accessed: 2025-01-30.
- [36] Fernando S Paolo, David Kroodsma, Jennifer Raynor, Tim Hochberg, Pete Davis, Jesse Cleary, Luca Marsaglia, Sara Orofino, Christian Thomas, and Patrick Halpin. 2024. Satellite mapping reveals extensive industrial activity at sea. *Nature* 625, 7993 (2024), 85–91.
- [37] Varatheepan Paramanayakam, Andreas Karatzas, Iraklis Anagnostopoulos, and Dimitrios Stamoulis. 2024. Less is more: Optimizing function calling for llm execution on edge devices. arXiv preprint arXiv:2411.15399 (2024).
- [38] C Robinson, R Gupta, S Fobi Nsutezo, E Pound, A Ortiz, M Rosa, K White, R Dodhia, A Zolli, C Birge, et al. 2023. Turkey building damage assessment. *Microsoft Corporation* (2023).
- [39] João Daniel Silva, João Magalhães, Devis Tuia, and Bruno Martins. 2024. Large language models for captioning and retrieving remote sensing images. arXiv preprint arXiv:2402.06475 (2024).
- [40] Simranjit Singh, Michael Fore, and Dimitrios Stamoulis. 2024. Evaluating Tool-Augmented Agents in Remote Sensing Platforms. arXiv preprint arXiv:2405.00709 (2024).

- [41] Simranjit Singh, Michael Fore, and Dimitrios Stamoulis. 2024. GeoLLM-Engine: A Realistic Environment for Building Geospatial Copilots. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. 585–594.
- [42] Venkat Krishna Srinivasan, Zhen Dong, Banghua Zhu, Brian Yu, Damon Mosk-Aoyama, Kurt Keutzer, Jiantao Jiao, and Jian Zhang. 2023. Nexusraven: a commercially-permissive language model for function calling. In NeurIPS 2023 Foundation Models for Decision Making Workshop.
- [43] Daniela Szwarcman, Sujit Roy, Paolo Fraccaro, Porsteinn Elí Gíslason, Benedikt Blumenstiel, Rinki Ghosal, Pedro Henrique de Oliveira, Joao Lucas de Sousa Almeida, Rocco Sedona, Yanghui Kang, et al. 2024. Prithvi-EO-2.0: A Versatile Multi-Temporal Foundation Model for Earth Observation Applications. arXiv preprint arXiv:2412.02732 (2024).
- [44] E. Vermote, C. Justice, M. Claverie, and B. Franch. 2015. MOD09GA MODIS/Terra Surface Reflectance Daily L2G Global 1km SIN Grid V006. doi:10.5067/MODIS/ MOD09GA.006 Accessed 2025-01-07.
- [45] vLLM Team. 2025. vLLM Documentation. https://docs.vllm.ai/en/stable/. Accessed: January 15, 2025.
- [46] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, and Peter Wolstenholme. 2006. Modeling software with finite state machines: a practical approach. Auerbach Publications.
- [47] Z. Wan, S. Hook, and G. Hulley. 2021. MODIS/Terra Land Surface Temperature/Emissivity 8-Day L3 Global 1km SIN Grid V061. https://doi.org/10.5067/ MODIS/MYD11A2.061. Accessed 2025-01-07.
- [48] T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019).
- [49] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. arXiv preprint arXiv:2308.08155 (2023).
- [50] Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. 2024. StateFlow: Enhancing LLM Task-Solving through State-Driven Workflows. arXiv preprint arXiv:2403.11322 (2024).
- [51] Zhitong Xiong, Yi Wang, Fahong Zhang, Adam J Stewart, Joëlle Hanna, Damian Borth, Ioannis Papoutsis, Bertrand Le Saux, Gustau Camps-Valls, and Xiao Xiang Zhu. 2024. Neural plasticity-inspired foundation model for observing the Earth crossing modalities. arXiv e-prints (2024), arXiv-2403.
- [52] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 Technical Report. arXiv preprint arXiv:2412.15115 (2024).
- [53] Guofeng Yang, Yu Li, Yong He, Zhenjiang Zhou, Lingzhen Ye, Hui Fang, Yiqi Luo, and Xuping Feng. 2024. Multimodal large language model for wheat breeding: a new exploration of smart breeding. arXiv preprint arXiv:2411.15203 (2024).
- [54] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629 (2022).
- [55] Chenyang Yu, Xinpeng Xie, Yan Huang, and Chenxi Qiu. 2024. Harnessing llms for cross-city od flow prediction. In Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems. 384–395.
- [56] Chiqun Zhang, Anirudh Sriram, Kuo-Han Hung, Renzhong Wang, and Dragomir Yankov. 2024. Context-aware Conversational Map Search with LLM. In Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems. 485–488.
- [57] Chenhui Zhang and Sherrie Wang. 2024. Good at captioning, bad at counting: Benchmarking gpt-4v on earth observation data. arXiv preprint arXiv:2401.17600 (2024)
- [58] Wei Zhang, Miaoxin Cai, Tong Zhang, Yin Zhuang, and Xuerui Mao. 2024. Earth-GPT: A Universal Multimodal Large Language Model for Multisensor Image Comprehension in Remote Sensing Domain. IEEE Transactions on Geoscience and Remote Sensing 62 (2024), 1–20.
- [59] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A Realistic Web Environment for Building Autonomous Agents. In The Twelfth International Conference on Learning Representations.
- [60] Xiao Xiang Zhu, Zhitong Xiong, Yi Wang, Adam J Stewart, Konrad Heidler, Yuanyuan Wang, Zhenghang Yuan, Thomas Dujardin, Qingsong Xu, and Yilei Shi. 2024. On the Foundations of Earth and Climate Foundation Models. arXiv preprint arXiv:2405.04285 (2024).
- [61] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. Advances in Neural Information Processing Systems 36 (2023), 50117–50143.