# GeoLLM-Engine: A Realistic Environment for Building Geospatial Copilots

Simranjit Singh, Michael Fore, Dimitrios Stamoulis
*CoStrategist* R&D Group, Microsoft Corporation, USA
{simsingh, mifore, stamoulis.dimitrios}@microsoft.com

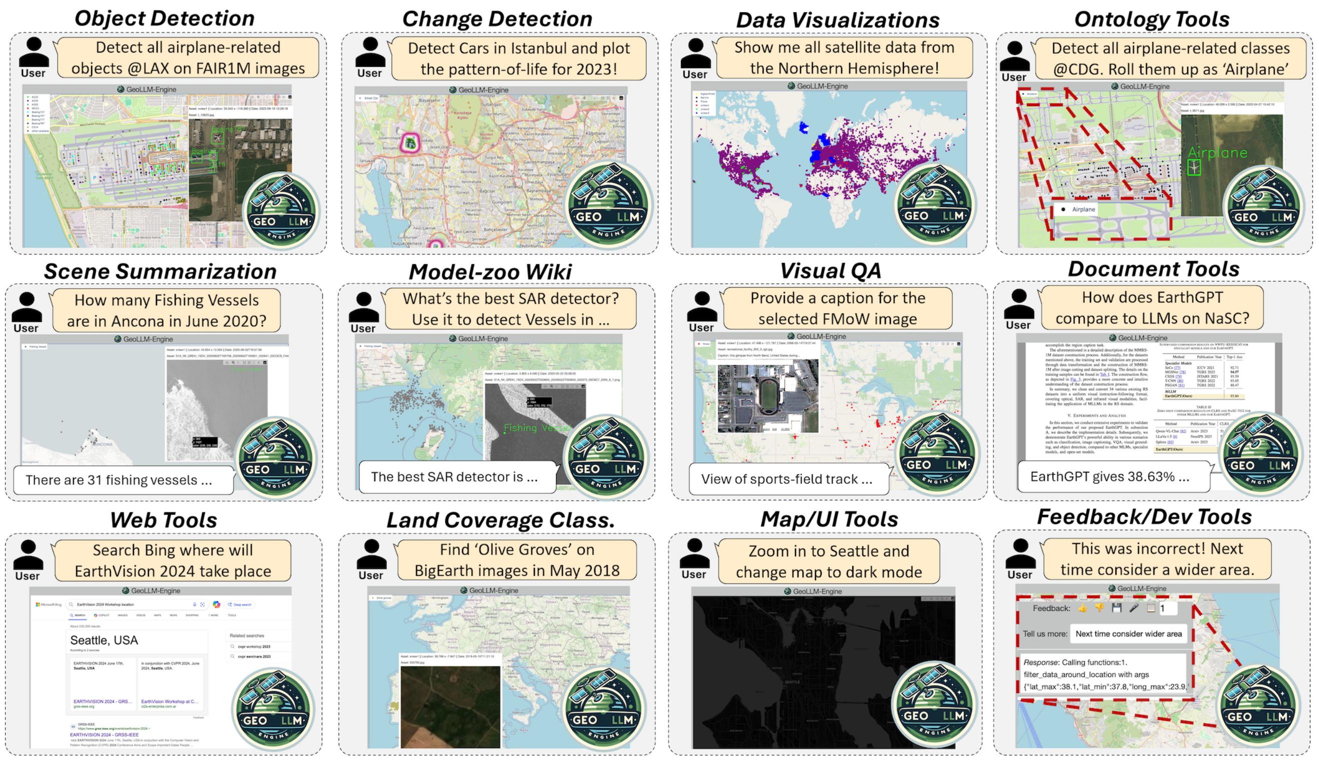
Object Detection
User
Detect all airplane-related objects @LAX on FAIR1M images
Change Detection
User
Detect Cars in Istanbul and plot the pattern-of-life for 2023!
Data Visualizations
User
Show me all satellite data from the Northern Hemisphere!
Ontology Tools
User
Detect all airplane-related classes @CDG. Roll them up as 'Airplane'
Airplane
Scene Summarization
User
How many Fishing Vessels are in Ancona in June 2020?
There are 31 fishing vessels ...
Model-zoo Wiki
User
What's the best SAR detector? Use it to detect Vessels in ...
The best SAR detector is ...
Visual QA
User
Provide a caption for the selected FMoW image
View of sports-field track ...
Document Tools
User
How does EarthGPT compare to LLMs on NaSC?
EarthGPT gives 38.63% ...
Web Tools
User
Search Bing where will EarthVision 2024 take place
Seattle, USA
Land Coverage Class.
User
Find 'Olive Groves' on BigEarth images in May 2018
Map/UI Tools
User
Zoom in to Seattle and change map to dark mode
Feedback/Dev Tools
User
This was incorrect! Next time consider a wider area.
Feedback:
Tell us more: Next time consider wider area
Response: Calling functions:1. filter_data_around_location with args
GeoLLM-Engine
GEO LLM ENGINE

Figure 1. GeoLLM-Engine offers a *realistic* web environment equipped with comprehensive APIs for developing, deploying, and evaluating geospatial Copilots on tasks that authentically reflect the workflows of remote sensing analysts. Unlike existing LLM benchmarks which rely on simplified question-answer pairs, GeoLLM-Engine introduces nuanced, long-horizon natural-language commands over a wide range of tasks - from object detection and land classification to document retrieval and web search.

## Abstract

*Geospatial Copilots unlock unprecedented potential for performing Earth Observation (EO) applications through natural language instructions. However, existing agents rely on overly simplified single tasks and template-based prompts, creating a disconnect with real-world scenarios. In this work, we present* `GeoLLM-Engine`*, an environment for tool-augmented agents with intricate tasks routinely executed by analysts on remote sensing platforms. We enrich our environment with geospatial API tools, dynamic maps/UIs, and external multimodal knowledge bases to properly gauge an agent's proficiency in interpreting* ***realistic*** *high-level natural language commands and its* ***functional*** *correctness in task completions. By alleviating overheads typically associated with human-in-the-loop benchmark curation, we harness our massively parallel engine across 100 GPT-4-Turbo nodes, scaling to over half a million diverse multi-tool tasks and across 1.1 million satellite images. By moving beyond traditional single-task image-caption paradigms, we investigate state-of-the-art agents and prompting techniques against long-horizon prompts.*

## 1. Introduction

With the advent of generative AI, Large Language Models (LLMs) have the potential to significantly enhance Earth Observation (EO) workflows [41, 49] across a broad range of tasks, from detection to learning from spatio-temporal data to analyzing aerial, UAV, and satellite images and videos [7]. However, existing approaches consider predefined low-level template-based prompts that only capture the *textual surface form* of the predicted image-caption pairs [52], while often overlooking the *functional agent correctness* at completing high-level natural language (NL) commands [17]. While the need for more representative benchmarks has been underscored across several generative AI domains [20, 28, 52], their significance is even greater in the geospatial domain, as it involves complex multimodal data across diverse spatial and temporal dimensions.

Such disconnect partly stems from prevailing perceptions that benchmark creation, focused on simplistic single-task prompts, is straightforward. However, the overhead extends beyond merely curating benchmarks with image-caption pairs, a task that can be programmatically accomplished, as evidenced by the rapid release of numerous geospatial benchmarks over recent months [10, 18, 29, 30, 32, 36, 41, 47, 49–51]. Instead, the challenge lies in establishing an environment equipped with the requisite tools, dynamic UIs, and real-world APIs to form the "engine" for developing complex tasks. In this work, our **key insight** is that amidst the abundance of "geospatial benchmarking" works, a subtle refocus is necessary, prioritizing the construction of a robust *engine as the foundation for benchmark creation*, rather than the benchmarks themselves.

We draw inspiration from novel work [28, 52, 53] that introduces environment-based benchmarking suites for comprehensive agent assessment. While these works highlight potential in their domains, adopting them for geospatial applications necessitates overcoming human-in-the-loop bottlenecks, particularly in manual ground-truth verification and template creation. To mitigate these challenges, we employ formal-language-based verification techniques [43], recently introduced to expedite labor-intensive Reinforcement Learning from Human Feedback (RLHF) workflows.

In this work, we introduce `GeoLLM-Engine`, a highly *realistic* environment that captures real-world tasks on EO platforms [13]. Our environment comprises various fully operational APIs and dynamic map/web UIs to execute geospatial tasks via high-level NL prompts. More importantly, we employ model-correctness checker techniques [43, 52] that allow our "back-end" engine to autonomously verify the accuracy of generated benchmarks, requiring only a one-off initial validation of task templates. By reducing the necessity for human intervention, we can massively parallelize our benchmark suite across 100 GPT-4-Turbo nodes to create large-scale benchmarks

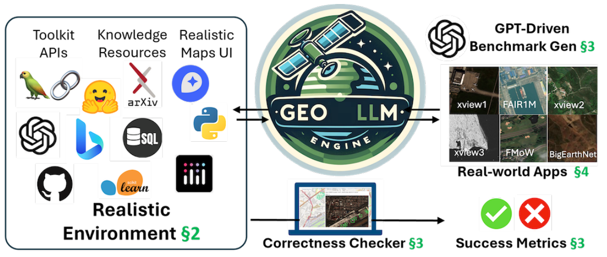

Figure 2. **Paper overview:** `GeoLLM-Engine` is a realistic environment for building and evaluating agents on complex EO tasks. Our GPT-driven benchmark-*engine* and the agent-correctness checker minimize the need for human intervention for benchmark curation. This allows us to develop a large-scale benchmark with more than $500,000$ *long-horizon* geospatial tasks on $1,100,000$ satellite images.

with 100,000 prompts that span half a million tasks over 1.1 million images from open-source EO datasets.

Each `GeoLLM-Engine` prompt exhibits high-level intent that emulates the nuances and abstract language usage patterns typically employed by human operators, as shown in Fig. 1. Using this benchmark, we follow state-of-the-art evaluation schemes with tool-augmented fine-tuning-free agents in zero-/few-shot in-context learning modality [17], powered by the latest GPT-3.5 and GPT-4 `Turbo (0125)` versions. Capturing our key insight, our findings reveal that merely expanding LLM benchmarks with more tasks of uniform complexity (*e.g.*, an excess of visual QA captioning tasks) does not significantly enrich our understanding, as agent performance predictably shows little variation. Conversely, we show that varying levels of complexity better assess agent performance. To this end, we diversify the scope of our benchmark by incorporating various remote sensing (RS) applications over different satellite imagery sources and tasks of escalating intents, ranging from document knowledge retrieval to UI/Web interactions to geospatial data analytics.

## 2. `GeoLLM-Engine` Environment

Our aim is to establish a realistic environment that clearly advances beyond current benchmarks, featuring a self-contained web UI with a varied array of LLM tools and an integrated benchmarking engine (Fig. 2). Built upon open-source libraries and APIs, `GeoLLM-Engine` is designed as a reproducible and scalable platform to support the development and evaluation of geospatial agents.

**Environment - "Front-end":** A key challenge in creating such an environment is the need for reproducibility and comparability across different systems and methodologies. To address this, we leverage a suite of open-source APIs, enabling the seamless integration of a wide array of tools, datasets, and functionalities, while facilitating transparency

| Tool Types | # Tools | Tool Examples |
|---|---|---|
| OpenAI | 5 | `json_mode(),embeddings()` |
| LangChain | 9 | `vectorstores.FAISS()` |
| Vision | 17 | `transformers.SwinModel()` |
| Map | 31 | `scattermapbox()` |
| Geotools | 23 | `rasterio(),geopandas()` |
| Python | 20 | `pyPDF(),PIL()` |
| Web | 15 | `openURL(),bing_search()` |
| Data Tools | 28 | `sklearn.cluster()` |
| Database | 8 | `db_load(),filter_date()` |
| UI | 12 | `display_hover()` |
| Knowledge | 6 | `wiki_RAG(),model_cards()` |

Table 1. `GeoLLM-Engine` module inventory: Tool Space $\mathcal{T}$

and accessibility. We intend to release our codebase and benchmark to foster advancements in geospatial Copilots.

**Tool Space:** We equip our environment with a comprehensive array of open-source Python packages, catering to various functionalities from data analytics to LLM-specific tasks, such as employing `LangChain` for FAISS [8] embeddings in knowledge retrieval applications. The interface leverages `Mapbox` APIs for interactive mapping, while `rasterio` and `geopandas` facilitate advanced manipulation of geospatial data. The complete module inventory, comprising over 175 tools, is presented in Tab. 1. This diverse toolkit enables us to execute complex tasks ranging from satellite imagery analysis to utilizing vector storage for rapid geographic data querying.

**Engine - "Back-end":** To facilitate large-scale experimentation, `GeoLLM-Engine` incorporates a Command-Line Interface (CLI) alongside its UI, providing comprehensive tooling and scripting capabilities. This feature is crucial for conducting extensive investigations, allowing us to efficiently run model-verification checks on over half a million tasks and evaluate baseline agents against our benchmarks. Thanks to this setup, these operations are completed within hours, leveraging hundreds of GPT endpoints. The CLI, in tandem with the UI, provides diverse modalities essential for replicating the intricate demands of geospatial analysis tasks which require the integration of disparate data sources and analytical techniques.

**Environment Formulation:** Our environment is represented as $\mathcal{E} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T} \rangle$, comprising the environment state $\mathcal{S}$, the action space $\mathcal{A}$, and the tool space $\mathcal{T}$ (Tab. 1). To intuitively understand $\mathcal{S}$, consider a user query "*zoom into the Indo-Pacific region and show me the vessels during November 2021*." Upon query completion, $\mathcal{S}$ will encapsulate not just the visual or textual response, but also the altered state of the map, such as its zoom level or position, the loaded database, and the temporal window of displayed data.

While detailed notation of transition functions extends beyond our study's scope, previous research [52] has established these functions as deterministic. This implies a critical property: given the same starting state $\mathcal{S}_0$, any two agents executing an identical sequence of actions or tools will invariably arrive at the same final state $\mathcal{S}^*$. This deterministic nature is vital for our purposes, as it allows for the verifiability of any agent's solution against the benchmark's "ground truth" specifications [43]. We exploit this feature in our model checker (elaborated in Sec. 3) in two principal ways. Firstly, given a "golden" ground truth, we can deterministically assess the functional correctness of any candidate solution that follows the same sequence of actions or tools. Secondly, this principle underpins our approach to benchmark (ground-truth) generation: by evaluating multiple agent solutions, a consensus on the final state among the majority indicates a high likelihood of an accurate ground truth, hence eliminating the need for human inspection.

**User Intent Formulation:** Intuitively, each user intent is encapsulated by four parts: the question $q$ that triggers the agent, the executed tool sequence $\hat{T}$, the agent's textual response $\hat{r}$ to the user, and the concluding environment state $\hat{S}$. Thus, we can express each task as $\{q, \hat{T}, \hat{r}, \hat{S}\}$. The sequence $\hat{T}$ is defined by the set of tool $\hat{T} = \{\hat{t}_1, \hat{t}_2, \dots\}$, where at each step $i$ the agent invokes tool $\hat{t}_i = \{\hat{tool}_i, \hat{args}_i\} \in \mathcal{T}$. As shown later in Sec. 3, by contrasting the task set $\{q, \hat{T}, \hat{r}, \hat{S}\}$ with a gold standard $\{q, T^*, r^*, S^*\}$, we can ascertain the functional correctness across our entire benchmark.

## 3. `GeoLLM-Engine` Benchmark Suite

In this section, we first describe the process for "*grounding*" high-level natural language instructions into a structured set of task templates and user intents that cover all the `GeoLLM-Engine` tools. Next, we discuss our GPT-driven, human-out-of-the-loop ground-truth sampling approach.

**Intent Collection:** The initial phase involves utilizing human annotators (our team members) to craft a small set of user intents, following the qualitative guidelines set forth in [52]. These intents are designed to be both nuanced and high-level, requiring the agent to perform more than one or two actions, and should be decomposable into a series of interchangeable templates. While this step is manual, please note that it represents the primary (and essentially the only) offline step necessitating human involvement. Through this procedure, we generate the foundational intents and templates, serving as the modular components from which benchmark queries are constructed.

The annotators are instructed to input detailed prompts, utilizing zero-shot GPT-4-Turbo with Chain-of-Thought prompting [42] to propose solutions (as highlighted in our Results, GPT-4 demonstrates notable zero-shot capabilities). Utilizing "User Feedback" UI buttons, annotators

| Category | Example | Tools $T$ |
|---|---|---|
| Load→Filter→Plot | Plot on the map the fair1m and xview1 images around Pittsburgh, PA, USA from May 2023 for all vehicle-related categories | `db_load(), filter_coords(), filter_date(), filter_class(),...` |
| | Show me all *golf course* LCC images for FMoW from January 2015 to May 2016 | `db_load(), filter_lccclass(), filter_date(), scatter_plot(),...` |
| | Fetch all Fishing-Vessel detections on xview3 from Belle Île en Mer, France | `db_load(), filter_coords(),...` |
| UI/Web Navigation | Can you search Google for "Foundation models in Earth Observation apps"? | `google_search(), open_url()` |
| | Can you zoom the map to Seattle, WA? | `zoom_map()` |
| Information Seeking | How many distinct semantic tags does SkyScript (arxiv) covers and what percentage of image-tag pairs was *manually* checked for tag accuracy? | `arxiv_search(), docs_RAG(), answer_tools()` |
| | According to our Model wiki, which detector is best-suited for vessels on SAR imagery? | `modelzoo_search(), answer_tools()` |

Table 2. Examples of *realistic* `GeoLLM-Engine` tasks across three primary "Intent" categories. Please note the *long-horizon high-level* natural language commands, which constitutes a novel departure from existing single-task template-based geospatial LLM benchmarks.

identify and confirm correctly executed model responses, hence collecting the "correct" examples along with the corresponding agent actions. By promoting scenarios that necessitate an average of 7-8 tool interactions, we can cover the entire tool space within 250 instantiated queries. Following the granular intent categories as in [52], we classify the various prompts into the categories shown in Tab. 2:

1. **Information Seeking**: Queries for knowledge retrieval aimed at sourcing information from wikis and documents to support EO investigations.
2. **UI/Web Navigation**: Commands designed for UI interaction, such as opening web search results or displaying images corresponding to detections mapped out.
3. **Load-Filter-Plot**: Operations to load data, apply specific filters, and present geospatial findings in an insightful manner, for example, through change-detection heatmaps or land cover classification (LCC) categories.

**Tool Templates:** Given the 250 instantiated queries and GPT's solutions, we programmatically parse all `json` responses and remove duplicates, obtaining a **distinct template for every tool agent-call**, *i.e.*, $\forall t_i \in \mathcal{T}$ we have the GPT `json`-call $\{toolname_i, args_i\}$. These `json` responses furnish us with the "building blocks" that can be used for benchmark sampling in a straightforward intuition: leveraging the recently introduced `json_mode` feature in OpenAI's APIs, we can initiate queries to a standalone GPT model. Within this context, we present the model with a specific tool interaction template and pose inquiries akin to: "*Given your prior action of zooming into X and the corresponding function call, generate the new function call to instead focus on Vienna.*" This allows us to dynamically generate tool-specific commands tailored to new, contextually relevant scenarios.

**GPT-Driven Benchmark Creation:** To efficiently scale our benchmark generation, we capitalize on two key attributes of GPT-4-Turbo agents: the extended input token lengths and their ability to parse extensive databases described through SQL-like schemas. We leverage the expanded token capacity and we incorporate directly into the model's context **all** tool-calling `json` templates for the 174 tools (manually verified in the previous step). Moreover, we compile the metadata (*e.g.*, coordinates, dates, categories, document titles, *etc.*) associated with all satellite images and documents into a SQL table. From this, we randomly select 1,000 entries, collecting their categories, coordinates, and dates sets and providing them as SQL schemas to GPT. Last, we append the 250 manually crafted queries as "successfully sampled" examples, and alongside in-context "benchmark creation" instructions, this entire prompt is fed to GPT-4, prompting it to autonomously generate both a suggested task prompt and the corresponding solution, forming a new query task $\{q, T^*, r^*, S^*\}$.

To circumvent the need for manual verification of each solution's accuracy, we introduce a novel approach that integrates functional model checking with the principle of LLM *self-consistency* [40]. Drawing from the concept where an agent repeatedly solves the same prompt, typically converging on the correct solution, we apply this by having GPT-4 propose multiple solutions to its self-generated (original) prompt. After 10 iterations, we execute these solutions in our engine (in-parallel via our CLI tools) to ascertain the final state $S^*$ for each. If 9 out of 10 solutions converge on the same end state, we consider this to be a verified ground truth and incorporate it into our dataset. Overall, this novel *self-consistency* sampling scheme allows us to streamline the benchmark generation process without extensive human intervention.

**Model-Checker Formulation:** `GeoLLM-Engine` incorporates a rigorous set of model checks to determine both the functional correctness and overall success of an agent's response to a given prompt. As shown in Tab. 2, *correctness* checks the agent's tool usage $\hat{T}$ against the corresponding ground-truth $T^*$, assessing argument accuracy within function calls (*e.g.*, missing a required tool or calling the right tool with wrong parameters). *Success* checks the final system state produced by the agent's sequence of actions $\hat{S}$ against the expected ground-truth state $S^*$. We leverage our engine to "run" both the sequences starting from

| Model Checks | Function | Description | Condition | Check Failed? |
|---|---|---|---|---|
| ***Correctness*** | | | | |
| Tool Calls | $f_{valid}(\hat{tool})$ | Agent selected “real” tool | $tool \in$ tool space $\mathcal{T}$ | Infeasible Action |
| | $f_{needed}(\hat{tool})$ | Tool was required to solve the task | $\hat{tool} \in$ ground-truth tools $T$ | Function Error |
| | $f_{missed}(tool^*)$ | Agent missed the tool from its solution | $tool^* \in T \| \| tool^* \notin$ solution $\hat{T}$ | Missed Function |
| Tool Args | $f_{str}(\hat{args}, args^*)$ | Input variables (string) names matching | $\forall arg_i : \hat{arg}_i == arg_i^*$ | Argument Error |
| | $f_{num}(\hat{args}, args^*)$ | Input variables (float) “matching” | $\forall arg_i : \|\hat{arg}_i - arg_i^*\| < \epsilon$ | |
| ***Success*** | | | | |
| Agent Output | $f_{reply}(\hat{r}, r^*)$ | Response reasonably close to expected answer | $ROUGE(\hat{r}, r^*) > \epsilon_R$ | Unsuccessful Task |
| | $f_{output}(\hat{o}, o^*)$ | Correct output (*e.g.*, detected objects) | | |
| Tool Args | $f_{map}(\hat{S}_{map}, S^*_{map})$ | Correct map view (*e.g.*, zoomed area) | $\hat{S}_{map} == S^*_{map}$ | |
| | $f_{map}(\hat{S}_{UI}, S^*_{UI})$ | Correct UI view (*e.g.*, open web-tabs) | $\hat{S}_{UI} == S^*_{UI}$ | |

Table 3. We define *model-checks* to obtain the agent success and correctness at performing each task.

the same initial state to confirm whether the agent's output and ground-truth match. Note the distinction between the two failure cases; they are not mutually inclusive. For example, an agent may erroneously invoke an unnecessary tool (“Function Error”), yet this may not alter the final state, which could still align with the anticipated result.

**Agent Evaluation Metrics:** Based on the model checks, we can define the appropriate metrics that we subsequently use to evaluate the performance of different agents:

1. *Success rate*: the ratio of successfully completed tasks across the entire benchmark as defined by $f_{success}$ (Tab. 3). This ratio informs us of the degree to which the agent is able to complete tasks, irrespective of whether it took incorrect or unnecessary intermediate steps.
2. *Correctness rate*: the ratio of correct function-call operations across the benchmark as defined by the $f_{correctness}$ error types. Given the total number of errors and ground-truth tools, we compute the correctness ratio $R_{correct} = \max(0, 1 - N_{errors}/N_{tools})$ [28, 53], which captures how likely it is for the agent to invoke the correct functions in the expected order.
3. *ROUGE score*: ROUGE-L recall score [21] to compare final model replies $\hat{r}$ with the ground truth $r^*$.

## 4. Remote Sensing Datasets

We consider several open-source data sources encompassing tasks like object detection, land cover classification, and visual question answering to build a collective dataset of **1,149,612** images. All data sources have coordinates and time metadata which offer us global spatio-temporal coverage in our benchmark prompts (Fig. 3):

1. *xView1* [19]: 846 high-resolution images sourced from WorldView-3 satellites focusing on overhead object detection with 1 million objects and 60 classes.
2. *xView2* [11]: dataset for building damage assessment with 5,598 images sourced from the Maxar/DigitalGlobe Open Data Program before and after 19 natural disasters with 850,736 building annotations.
3. *xView3* [31]: a dataset of 23,432 SAR GRD images from the Sentinel-1 mission annotated for (fishing) vessel detections to study illegal fishing practices.
4. *SARFish* [27]: extends the xView3-SAR GRD dataset by providing products from the Sentinel-1 C-band SAR satellite constellation operated by the European Space Agency's (ESA) Copernicus Program in both real-valued GRD and complex-valued SLC product types.
5. *FAIR1M* [39]: object detection dataset with 24,775 images with over 1 million instances sourced from Gaofen satellites and Google Earth.
6. *Functional Map of the World (FMoW)* [5]: a multi-label LCC dataset for buildings/land use with 727,144 images from over 200 countries.
7. *BigEarthNet* [38]: 344,385 Sentinel-2 LCC images from the CORINE Land Cover database.

## 5. Results

Tab. 4 summarizes the performance of state-of-the-art GPT-based agents with various prompting schemes. In addition to the LLM metrics, we report the agent's performance with respect to the underlying object detection (F1 score), LCC (Recall), and visual question answering (VQA, Rouge-L) tasks in our benchmark. These metrics provide insights into each agent's efficiency, accuracy, and responsiveness in executing geospatial tasks.

**GPT versions:** We observe that there's a marked difference in performance between models based on the GPT-3.5 Turbo and GPT-4 Turbo frameworks, with the latter generally achieving higher scores across all metrics. For instance, both the Chain-of-Thought [42] (CoT) and ReAct [46] methods, especially in their few-shot configura-

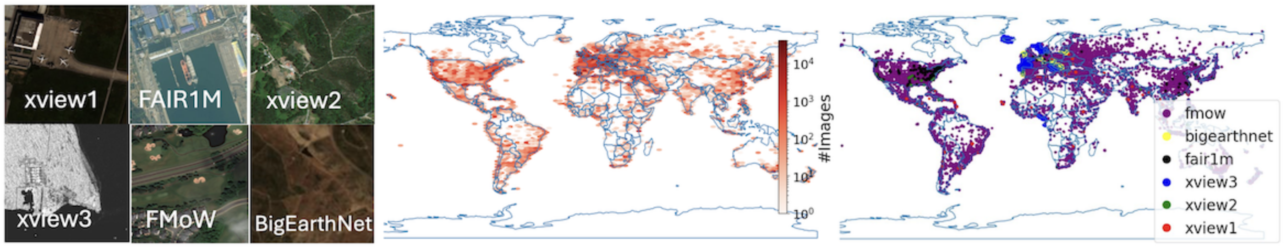

Figure 3. Dataset overview with a wide spatial and temporal range. The satellite imagery serves as *task contexts* for LLM agents to perform function calls (*e.g.*, given image coordinates, we can assess the agents' ability to fetch the proper set of images around a user-specified location) and is not employed for any LLM-finetuning or other downstream tasks, enabling our research-purposes investigation.

| Method | Correctness Rate ↑ | Success Rate ↑ | Obj. Det F1 ↑ | LCC R ↑ | VQA Rouge-L ↑ | Avg. Tokens / Task ↓ |
|---|---|---|---|---|---|---|
| ***GPT-3.5 Turbo (0125)*** | | | | | | |
| Chameleon [25] Zero-Shot | 38.08% | 40.94% | 39.18% | 49.52% | 52.74% | 38.9k |
| Chameleon [25] Few-Shot | 60.31% | 48.31% | 43.85% | 44.29% | 54.62% | 39.7k |
| CoT [42] Zero-Shot | 40.20% | 64.66% | 54.99% | 93.28% | 54.40% | 21.1k |
| CoT [42] Few-Shot | 65.77% | 68.45% | 73.81% | **98.39%** | 56.28% | 26.4k |
| ReAct [46] Zero-Shot | 62.06% | 51.56% | 54.16% | 92.57% | 56.45% | 17.3k |
| ReAct [46] Few-Shot | **68.42%** | **73.47%** | **75.01%** | 97.45% | **65.26%** | 30.9k |
| ***GPT-4 Turbo (0125)*** | | | | | | |
| Chameleon [25] Zero-Shot | 46.31% | 57.10% | 45.57% | 91.20% | 49.76% | 38.3k |
| Chameleon [25] Few-Shot | 67.78% | 59.01% | 55.82% | 67.92% | 49.27% | 40.7k |
| CoT [42] Zero-Shot | 80.88% | 77.35% | 87.99% | 96.56% | 65.29% | 23.6k |
| CoT [42] Few-Shot | 84.01% | 80.00% | 88.40% | **99.89%** | 67.65% | 25.8k |
| ReAct [46] Zero-Shot | 84.27% | 80.03% | **89.34%** | 98.83% | 68.11% | 26.7k |
| ReAct [46] Few-Shot | **84.31%** | **81.11%** | 83.85% | 99.63% | **69.37%** | 32.5k |

Table 4. The overall performance of different methods and prompting strategies on `GeoLLM-Engine-10k`.

tions, demonstrate a significant leap in correctness and success rates when transitioning from GPT-3.5 to GPT-4. This improvement underscores the advancements in model understanding and task execution capabilities. This finding is consistent with most of the work in language guided agents [4]. As anticipated, the few-shot configurations outperform their zero-shot counterparts across all evaluated methods. This trend underscores the value of providing models with a few examples to adapt to specific tasks, significantly enhancing their ability to accurately interpret and respond to complex geospatial queries.

**Prompting Schemes:** We observe that ReAct generally outperforms CoT for both GPT versions, as confirmed across several domains deploying ReAct for multimodal tasks [44]. Moreover, Chameleon's performance is notably lower than that of CoT and ReAct, especially evident with the GPT-4 model, indicating that while Chameleon is a valuable method, its prompting "decomposition" strategy which is primarily multi-step single-tool prompting makes it less suited for long-horizon (multi/single-step multi-tool) geospatial tasks. Please note that our focus is tool-augmented fine-tuning-free agents, so we leave the exploration of fine-tuned LLMs for future work.

**Model Cost:** An interesting observation is the lack of a clear correlation between the tokens consumed and the success or correctness rate. Despite Chameleon consuming the highest number of tokens on average, it does not lead in terms of success or correctness rates. This finding suggests token usage does not directly translate to higher performance, challenging the assumption that more extensive responses might yield better results. These insights collectively highlight the nuanced dynamics of model performance within the `GeoLLM-Engine` benchmark.

**Success Rate vs. Task Complexity:** Delving into the relationship between success rate (SR) and task complexity provides further insights into agent performance. As shown in Fig. 4, there is an inverse relationship between SR and the number of tool calls required to answer a prompt. Specif-

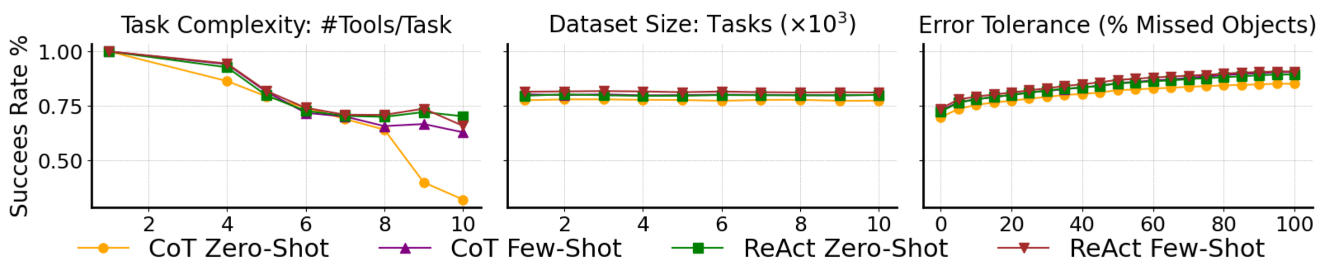

Figure 4. Success Rate vs. Task Complexity (left), Benchmark size (middle), and Error Tolerance (right). All methods use GPT-4-Turbo.

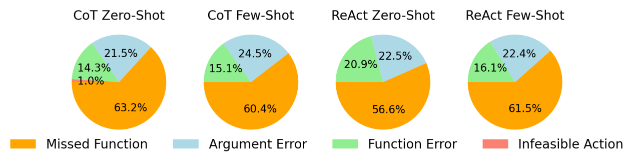

Figure 5. Distribution of different errors in correctness rate for CoT and ReAct methods. All methods use GPT-4

| Benchmark | Number of Queries | Number of Tasks | Success Rate |
|---|---|---|---|
| GeoLLM-Engine-10k | 10,000 | 50,830 | 77.35% |
| GeoLLM-Engine-100k | 100,000 | **521,868** | 76.81% |

Table 5. Success Rate of GPT-4 CoT on massively large-scale benchmark with 100k queries

ically, note how for tasks requiring a single tool call, SR exceeds 95%, while for more complex tasks involving more than eight tool calls SR is below 70% and even dropping below 27% for certain agents. This finding highlights a **critical limitation** of current geospatial agents: while agents can handle simpler tasks with relative ease, their performance degrades as the task complexity increases. We emphasize the need for benchmarks that measure agents' real-world utility against complex geospatial scenarios.

**Success Rate vs. Benchmark size:** Next, we conduct ablations varying dataset size from 500 to 10,000 tasks (Fig. 4): for all methods, the success rates remain relatively unchanged across this range. In a significant escalation of our testing regime, we scaled our GeoLLM-Engine benchmark to 100,000 queries covering over half a million tool calls (Tab. 5). For this experiment, which is the largest in this domain to our knowledge, we leverage the massively parallel nature of our engine over 100 GPT endpoints. Despite the tenfold increase, the success rate of GPT-4 CoT showed relative stability, suggesting that merely increasing the benchmark size does not necessarily challenge the agents more or provide a better assessment of their capabilities. Instead, it is the task complexity within the benchmarks that presents a significant factor in evaluating geospatial performance. These findings critique the prevalent approach in recent works where the focus has been on scaling the benchmark size, while our findings highlight that increasing the complexity of tasks is more essential for assessing agent performance.

**Success Rate vs. Error tolerance:** Last, we capture GeoLLM-Engine's parameterizable nature that allows us to evaluate agents based on varying error thresholds by properly updating the $\epsilon$ in the model-checkers functions. The incremental increase in success rates with the relaxation of error thresholds reflect scenarios allowing for a certain margin of error or applications where absolute precision is often unattainable (*e.g.*, nuanced definition of what a user means when asking "*Show me all car detections in Madrid*"; whether this assumes to include suburban areas or not might vary per user).

**Correctness Rate:** Fig. 5 shows the error types for CoT and ReAct on GPT-4, in both zero-shot and few-shot scenarios. The most common error type is "Missed Function", suggesting that the GPT-4 often omits necessary tool calls regardless of the approach used, accounting for more than half of all errors. These confirm recent findings that dynamic/RAG-augmented [37] prompting further improves agent performance by addressing such failures. Last, the consistent error distribution across different methods implies that these issues are not method-specific but rather inherent to the current capabilities of the underlying GPT-4.

## 6. Related Work

Recent advancements in autonomous agents span from reinforcement learning platforms [3] to web-based applications and benchmarks for web interactions [6, 22, 34, 45, 52]. VisualWebArena [17] represents a milestone of LLM benchmarking, offering realistic tasks to evaluate multimodal web agents towards improving performance on complex web pages. The adoption of multimodal models for EO tasks is gaining momentum (Tab. 6). Innovations like SkyEyeGPT [49] and Remote Sensing ChatGPT [10] showcase advancements in integrating VQA agents and computer vision models with RS imagery for enhanced multimodal responses.

| Geospatial Benchmarks | Dynamic Agent? | Standalone UI Engine? | Diverse Commands? | Functional Correctness? | Number of Images | Number of Queries | Avg. Query Complexity |
|---|---|---|---|---|---|---|---|
| Charting [32] | ✓ | ✓ | ✗ | ✗ | $\sim$ 1,000 | - | 1 task/query |
| GeoChat [18] | ✓ | ✓ | ✓ | ✗ | $\sim$ 150,000 | 318,000 | $\sim$1 task/query$^\dagger$ |
| ChatEarthNet [47] | ✗ | ✗ | ✓ | ✗ | 173,488 | 173,488 | $\sim$1 task/query$^\dagger$ |
| GRAFT [29] | ✗ | ✗ | ✓ | ✗ | 18,9000,000 | - | 1 task/query |
| RSVG [48] | ✗ | ✗ | ✓ | ✗ | 17,402 | 38,320 | 1 task/query |
| RSGPT [12] | ✗ | ✗ | ✗ | ✗ | - | 2,585 | 1 task/query |
| Remote-CLIP [23] | ✗ | ✗ | ✓ | ✗ | - | - | 1 task/query |
| RS-CLIP [35] | ✗ | ✗ | ✗ | ✗ | 45,134 | 225,670 | 1 task/query |
| RSICD [26] | ✗ | ✗ | ✓ | ✗ | 10,921 | 24,333 | 1 task/query |
| RS-ChatGPT [10] | ✓ | ✓ | ✓ | ✗ | - | - | $\sim$1 task/query $^\dagger$ |
| SkyEyeGPT [49] | ✓ | ✓ | ✓ | ✗ | - | 968,000 | $\sim$1 task/query $^\dagger$ |
| SkyScript [41] | ✗ | ✓ | ✓ | ✗ | 2,600,000 | 1,300,000 | 1 task/query |
| RS5M [51] | ✗ | ✗ | ✓ | ✗ | >5,000,000 | - | 1 task/query |
| EarthGPT [50] | ✓ | ✓ | ✓ | ✗ | - | >1,000,000 | $\sim$1 task/query $^\dagger$ |
| RSVQA [24] | ✗ | ✗ | ✓ | ✗ | 1,066,316 | 10,659 | 1 task/query |
| LHRS-Bot [30] | ✗ | ✗ | ✓ | ✗ | 15,000 | 1,150,000 | $\sim$1 task/query$^\dagger$ |
| `GeoLLM-Engine` | ✓ | ✓ | ✓ | ✓ | 1,149,612 | 521,868 | 5.23 tasks/query |

Table 6. Comparison `GeoLLM-Engine` to existing geospatial LLM-related benchmarks. $^\dagger$Note: while these works consider dynamic chatGPT-like prompting capabilities, the underlying agent execution is conducted mainly across multiple steps via conversational prompting of separate tasks (*i.e.*, *multi-step single-task*), rather that multiple tasks (tools) per query (*i.e.*, *single/multi-step* ***multi***-*task*).

However, existing benchmarks often rely on predefined, single-step text-image prompts. `GeoLLM-Engine` allows us to assess agents' ability to execute nuanced EO tasks.

## 7. Limitations and Future Work

We recognize limitations within our framework. First, using GPT-4 for both generating and evaluating ground truths could introduce bias risks, as highlighted by benchmarking work [52]. We are currently enhancing sampling diversity by leveraging hybrid strategies that incorporate both GPT-generated outputs and programmatic elements [53]. Second, while our platform emulates complex tasks reflective of EO analysts' workflows, we emphasized depth (*long-horizon* tasks) over breadth in task complexity. We are actively expanding the capabilities of our engine to incorporate a wider variety of tasks (*e.g.*, from maritime traffic analysis [2] to illegal fishing [1] to damage assessment [33]) leveraging `GeoLLM-Engine`'s flexible APIs. Furthermore, we have focused on assessing finetuning-*free* agents, which is why `GeoLLM-Engine` doesn't specify train-val-test splits. Following recent work on tuning LLMs on EO tasks [49], we are expanding our benchmark to tool-agents training [14]. Last, we have utilized standard LLM APIs without explicitly optimizing for cost (*e.g.*, latency or token usage). We are currently orthogonal optimizations to enhance `GeoLLM-Engine` efficiency, such as LLM compilers [16], dynamic tooling [9], and token compression [15].

## 8. Conclusion

In this work, we introduce `GeoLLM-Engine`, a novel environment for evaluating geospatial Copilots, designed to bridge the gap between simplistic benchmarks and the complex demands of Earth Observation (EO) applications. By leveraging a rich array of geospatial API tools, dynamic interfaces, and a massive parallel processing framework over 100 GPT-4-Turbo nodes, our environment facilitates the execution of over half a million multifaceted tasks across 1.1 million satellite images. This advancement not only highlights the limitations of existing benchmarks but also sets a new standard for the development and evaluation of AI agents in the geospatial domain. Looking forward, `GeoLLM-Engine` paves the way for future research to explore sophisticated EO tasks, promising significant strides toward realizing the full potential of geospatial Copilots.

## References

[1] David J Agnew, John Pearce, Ganapathiraju Pramod, Tom Peatman, Reg Watson, John R Beddington, and Tony J Pitcher. Estimating the worldwide extent of illegal fishing. PloS one, 4(2):e4570, 2009. 8

[2] Favyen Bastani, Piper Wolters, Ritwik Gupta, Joe Ferdinando, and Aniruddha Kembhavi. Satlaspretrain: A large-scale dataset for remote sensing image understanding. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 16772–16782, 2023. 8

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas

Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. CoRR, abs/1606.01540, 2016. 7

[4] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023. 6

[5] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6172–6180, 2018. 5

[6] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023. 7

[7] Department of Economic and Social Affairs, United Nations. Sustainable development goals. https://sdgs.un.org/goals, 2023. Accessed: March-2024. 2

[8] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2024. 3

[9] Michael Fore, Simranjit Singh, and Dimitrios Stamoulis. Geckopt: Llm system efficiency via intent-based tool selection, 2024. 8

[10] Haonan Guo, Xin Su, Chen Wu, Bo Du, Liangpei Zhang, and Deren Li. Remote sensing chatgpt: Solving remote sensing tasks with chatgpt and visual models, 2024. 2, 7, 8

[11] Ritwik Gupta, Richard Hosfelt, Sandra Sajeev, Nirav Patel, Bryce Goodman, Jigar Doshi, Eric Heim, Howie Choset, and Matthew Gaston. xbd: A dataset for assessing building damage from satellite imagery, 2019. 5

[12] Yuan Hu, Jianlong Yuan, Congcong Wen, Xiaonan Lu, and Xiang Li. Rsgpt: A remote sensing vision language model and benchmark, 2023. 8

[13] Johannes Jakubik, Sujit Roy, C. E. Phillips, Paolo Fraccaro, Denys Godwin, Bianca Zadrozny, Daniela Szwarcman, Carlos Gomes, Gabby Nyirjesy, Blair Edwards, Daiki Kimura, Naomi Simumba, Linsong Chu, S. Karthik Mukkavilli, Devyani Lambhate, Kamal Das, Ranjini Bangalore, Dario Oliveira, Michal Muszynski, Kumar Ankur, Muthukumaran Ramasubramanian, Iksha Gurung, Sam Khallaghi, Hanxi, Li, Michael Cecil, Maryam Ahmadi, Fatemeh Kordi, Hamed Alemohammad, Manil Maskey, Raghu Ganti, Kommy Weldemariam, and Rahul Ramachandran. Foundation models for generalist geospatial artificial intelligence, 2023. 2

[14] Yanan Jian, Fuxun Yu, Simranjit Singh, and Dimitrios Stamoulis. Stable diffusion for aerial object detection. In NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI, 2023. 8

[15] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models, 2023. 8

[16] Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An llm compiler for parallel function calling, 2024. 8

[17] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks, 2024. 2, 7

[18] Kartik Kuckreja, Muhammad Sohail Danish, Muzammal Naseer, Abhijit Das, Salman Khan, and Fahad Shahbaz Khan. Geochat: Grounded large vision-language model for remote sensing, 2023. 2, 8

[19] Darius Lam, Richard Kuzma, Kevin McGee, Samuel Dooley, Michael Laielli, Matthew Klaric, Yaroslav Bulatov, and Brendan McCord. xview: Objects in context in overhead imagery, 2018. 5

[20] Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. Open Review, 62(1), 2022. 2

[21] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. 2004. 5

[22] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. CoRR, abs/1802.08802, 2018. 7

[23] Fan Liu, Delong Chen, Zhangqingyun Guan, Xiaocong Zhou, Jiale Zhu, Qiaolin Ye, Liyong Fu, and Jun Zhou. Remoteclip: A vision language foundation model for remote sensing, 2024. 8

[24] Sylvain Lobry, Diego Marcos, Jesse Murray, and Devis Tuia. Rsvqa: Visual question answering for remote sensing data. IEEE Transactions on Geoscience and Remote Sensing, 58 (12):8555–8566, 2020. 8

[25] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models, 2023. 6

[26] Xiaoqiang Lu, Binqiang Wang, Xiangtao Zheng, and Xuelong Li. Exploring models and data for remote sensing image caption generation. IEEE Transactions on Geoscience and Remote Sensing, 56(4):2183–2195, 2018. 8

[27] Connor Luckett, Benjamin McCarthy, Tri-Tan Cao, and Antonio Robles-Kelly. The sarfish dataset and challenge. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops, pages 752–761, 2024. 5

[28] Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. Tofu: A task of fictitious unlearning for llms, 2024. 2, 5

[29] Utkarsh Mall, Cheng Perng Phoo, Meilin Kelsey Liu, Carl Vondrick, Bharath Hariharan, and Kavita Bala. Remote sensing vision-language foundation models without annotations via ground remote alignment, 2023. 2, 8

[30] Dilxat Muhtar, Zhenshi Li, Feng Gu, Xueliang Zhang, and Pengfeng Xiao. Lhrs-bot: Empowering remote sensing with vgi-enhanced large multimodal language model, 2024. 2, 8

[31] Fernando Paolo, Tsu ting Tim Lin, Ritwik Gupta, Bryce Goodman, Nirav Patel, Daniel Kuster, David Kroodsma, and Jared Dunnmon. xview3-sar: Detecting dark fishing activity using synthetic aperture radar imagery, 2022. 5

[32] Jonathan Roberts, Timo Lüddecke, Rehan Sheikh, Kai Han, and Samuel Albanie. Charting new territories: Exploring the geographic and geospatial capabilities of multimodal llms, 2024. 2, 8

[33] Caleb Robinson, Simone Fobi Nsutezo, Anthony Ortiz, Tina Sederholm, Rahul Dodhia, Cameron Birge, Kasie Richards, Kris Pitcher, Paulo Duarte, and Juan M Lavista Ferres. Rapid building damage assessment workflow: An implementation for the 2023 rolling fork, mississippi tornado event. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 3760–3764, 2023. 8

[34] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In Proceedings of the 34th International Conference on Machine Learning, pages 3135–3144. PMLR, 2017. 7

[35] João Daniel Silva, João Magalhães, Devis Tuia, and Bruno Martins. Large language models for captioning and retrieving remote sensing images, 2024. 8

[36] Simranjit Singh, Michael Fore, and Dimitrios Stamoulis. Evaluating tool-augmented agents in remote sensing platforms, 2024. 2

[37] Venkat Krishna Srinivasan, Zhen Dong, Banghua Zhu, Brian Yu, Hanzi Mao, Damon Mosk-Aoyama, Kurt Keutzer, Jiantao Jiao, and Jian Zhang. Nexusraven: a commercially-permissive language model for function calling. In NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following, 2023. 7

[38] Gencer Sumbul, Marcela Charfuelan, Begüm Demir, and Volker Markl. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. In IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium, pages 5901–5904. IEEE, 2019. 5

[39] Xian Sun, Peijin Wang, Zhiyuan Yan, Feng Xu, Ruiping Wang, Wenhui Diao, Jin Chen, Jihao Li, Yingchao Feng, Tao Xu, et al. Fair1m: A benchmark dataset for fine-grained object recognition in high-resolution remote sensing imagery. ISPRS Journal of Photogrammetry and Remote Sensing, 184:116–130, 2022. 5

[40] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. 4

[41] Zhecheng Wang, Rajanie Prabha, Tianyuan Huang, Jiajun Wu, and Ram Rajagopal. Skyscript: A large and semantically diverse vision-language dataset for remote sensing, 2023. 2, 8

[42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. 3, 5, 6

[43] Yunhao Yang, Neel P. Bhatt, Tyler Ingebrand, William Ward, Steven Carr, Zhangyang Wang, and Ufuk Topcu. Fine-tuning language models using formal methods feedback, 2023. 2, 3

[44] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action, 2023. 6

[45] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Advances in Neural Information Processing Systems, pages 20744–20757. Curran Associates, Inc., 2022. 7

[46] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. 5, 6

[47] Zhenghang Yuan, Zhitong Xiong, Lichao Mou, and Xiao Xiang Zhu. Chatearthnet: A global-scale image-text dataset empowering vision-language geo-foundation models, 2024. 2, 8

[48] Yang Zhan, Zhitong Xiong, and Yuan Yuan. Rsvg: Exploring data and models for visual grounding on remote sensing data. IEEE Transactions on Geoscience and Remote Sensing, 61:1–13, 2023. 8

[49] Yang Zhan, Zhitong Xiong, and Yuan Yuan. Skyeyegpt: Unifying remote sensing vision-language tasks via instruction tuning with large language model, 2024. 2, 7, 8

[50] Wei Zhang, Miaoxin Cai, Tong Zhang, Yin Zhuang, and Xuerui Mao. Earthgpt: A universal multi-modal large language model for multi-sensor image comprehension in remote sensing domain, 2024. 8

[51] Zilun Zhang, Tiancheng Zhao, Yulong Guo, and Jianwei Yin. Rs5m and georsclip: A large scale vision-language dataset and a large vision-language model for remote sensing, 2024. 2, 8

[52] 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. Webarena: A realistic web environment for building autonomous agents, 2023. 2, 3, 4, 7, 8

[53] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for llm question answering with external tools, 2023. 2, 5, 8