GeoX: Mastering Geospatial Reasoning Through Self-Play and Verifiable Rewards

Kyeongjin Ahn
MPI-SWS
Kaiserslautern, Germany
kahn@mpi-sws.org

Seungeon Lee*
MPI-SWS
Kaiserslautern, Germany
selee@mpi-sws.org

Krishna P. Gummadi
MPI-SWS
Kaiserslautern, Germany
gummadi@mpi-sws.org

Meeyoung Cha
MPI-SP & KAIST
Germany & South Korea
mia.cha@mpi-sp.org

Abstract

Geospatial reasoning requires solving image-grounded problems over the complex spatial structure of a scene. However, developing this capability is hindered by the cost of annotating a vast and combinatorial question space. We propose GeoX, a self-play framework that acquires spatial logic through executable programs that yield verifiable rewards without relying on human-curated question-answer pairs. Given a satellite or aerial image, our framework employs a single multimodal policy that proposes spatial problems as executable programs and solves them under three reasoning modes—abduction, deduction, and induction—over spatial primitives and an image understanding tool. A verifier executes each program to convert reward signals that jointly optimize the two roles via reinforcement learning. GeoX consistently improves its base VLMs by up to 5.5 points on average, matching or exceeding conventional baselines trained on millions of curated data. Alongside the proposed method, we release a benchmark for geospatial understanding accumulated through self-play.

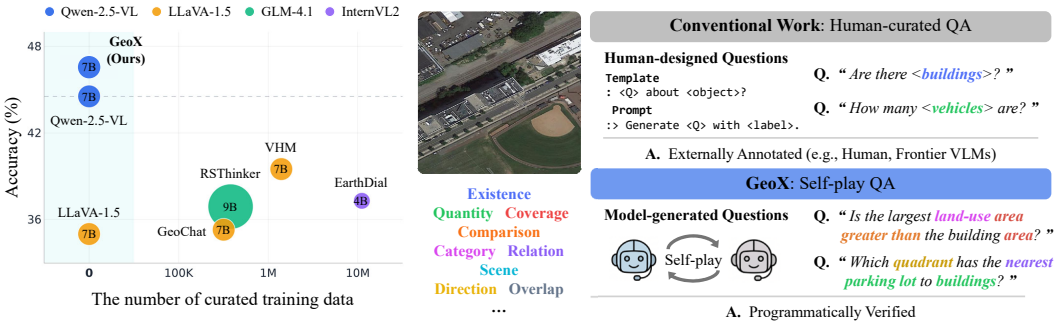
1 Introduction

Interpreting the complex Earth's surface from high-resolution satellite and aerial imagery is a central challenge in remote sensing, with broad implications for urban planning [1, 2, 4, 12], environmental monitoring [27, 21], and disaster response [9, 3]. As such imagery proliferates at an unprecedented scale, these capabilities are becoming a strategic necessity. Many practical questions like *"which buildings lie closest to the planned metro station?"* cannot be answered by recognizing *what* is present in an image alone. They instead demand a structured understanding of *how* objects interact, including their arrangement, adjacency, and relations across the scene. Standard tasks such as scene classification, object detection, and image captioning characterize the *contents*, but not the underlying *spatial structure* of a scene. *Geospatial reasoning* over this structure is thus a core capability for real-world deployment.

Specialized vision-language models (VLMs) for remote sensing have advanced significantly, but their geospatial understanding is bottlenecked by their supervision regime [11, 10, 30]. These models depend on fine-tuning over human-curated question–answer pairs, while the space of spatial questions explodes combinatorially with the many objects in each overhead image. For example, valid questions

*Corresponding author.

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



(a) Performance on VQA benchmarks (b) Paradigm comparison

Figure 1: **Standing and motivation. (a)** GeoX outperforms prior VLMs on VQA benchmarks while using zero curated training data; **(b)** Conventional work derives questions from human-designed templates with answers externally annotated, confining them to predefined patterns. Our framework replaces this paradigm with autonomous self-play, where a single model proposes questions whose answers are programmatically verified, broadening them to compositional patterns.

within an urban image already grow jointly with the target, the reference, and the property among its hundreds of objects, placing comprehensive coverage beyond the reach of manual annotation. As a result, existing datasets span only a small fraction of this space, leaving the models bounded by the scope of human curation (Figure 1a).

Our key insight is that imagery itself encodes the spatial structure that enables its own interpretation. Tasks involving area, centroid, distance, or adjacency reduce to discrete geometric and topological operations whose answers are fixed by the image rather than left to model judgment. Composed into executable programs, these operations admit uniquely determined answers recoverable from the image alone. The supervision required for geospatial reasoning is therefore latent in the scene and accessible without external annotation.

We propose a self-play framework GeoX for geospatial understanding (Figure 1b). A single model alternates between two roles: a *proposer* that constructs an executable problem over the input image, and a *solver* that finds its solution. Each problem specifies an executable program whose execution converts to a verifiable reward that jointly optimizes both roles via reinforcement learning, grounding the learning signal in the spatial structure of the image. To master a scene rather than merely answer questions about it, each problem is recast under three reasoning modes: *abduction*, *deduction*, and *induction*.

Our contributions are as follows.[1]

- **Self-play with verifiable rewards for geospatial reasoning.** We adapt the self-play paradigm using a proposer and solver for remote sensing imagery by routing perception into executable programs composed of spatial primitives and an image understanding tool. A verifier turns each problem into a verifiable reward signal by program execution, letting the model internalize the geometric and topological structure of the Earth's surface without curated data.
- **One proposal seen through three modes of reasoning.** Each problem is recast under *abduction*, *deduction*, and *induction*, which mask a different element of the problem and elicit a different mode of interrogation: inferring the cause from observed evidence, predicting outcomes from a known cause, and synthesizing the procedure that connects them. One proposal can afford three complementary views of the same scene.
- **Strongest gains where curated supervision falls short.** Across remote sensing benchmarks for visual question answering and object counting, our method matches or surpasses prior specialized VLMs, with the largest gains on counting and spatial relation tasks, precisely where the cost of curation becomes prohibitive.
- **A self-grown benchmark verified by program execution.** From the self-play process, we automatically accumulate problems with execution-verified answers, ranging from simple to

[1]Code is publicly available at `https://github.com/kyeongjin0110/GeoX`.

compositional relations. The resulting suite broadens existing remote sensing benchmarks, which skew toward presence detection and scene classification.

This framework is a path toward learning the spatial physics of remote sensing imagery. By grounding optimization in executable programs, the framework moves beyond rote prediction toward structural representations that capture the inherent properties of the physical world. This aligns with emerging efforts to develop methods that acquire physical laws via self-improving.

2 Related Work

2.1 Vision-Language Models for remote sensing

Recent progress has extended vision-language models (VLMs) to satellite and aerial imagery. RSVQA [16] framed visual question answering (VQA) as a core task for overhead imagery. RS-GPT [10] and LHRS-Bot [18] then released larger multimodal datasets alongside companion benchmarks, each training its own VLM on the proposed data. EarthGPT [30] and EarthDial [22] further extend coverage to multi-sensory [30] and multi-temporal [22] imagery. A parallel line of work targets structural understanding of overhead scenes. EarthVQA [25] contributes a dataset for relational reasoning, and GEOBench-VLM [6] compiles a broad task suite for evaluating specialized geospatial capabilities. Despite their growing scale and scope, these datasets all depend on manual annotation.

Subsequent VLMs go beyond scaling data to refining the supervision itself. GeoChat [11] unifies grounded conversation, VHM [19] enforces factual honesty with deceptive instructions, and RS-Thinker [15] elicits Chain-of-Thought rationales. However, these efforts likewise inherit human supervision, hindering scalability and risking error propagation. GeoX removes this dependence by deriving its supervision from the imagery itself.

2.2 Self-improvement in Foundation Models

A growing body of work scales learning by letting a model generate and learn from its own outputs. STaR [28] introduced this idea in the language domain, R^3V [5] extended it to the vision-language domain, and STIC [7] redirected it to image descriptions for visual comprehension. These approaches push supervision beyond manual curation, yet the self-generated signals they rely on are difficult to verify independently, accumulating noise across iterations.

Self-play with verifiable rewards allows models to use automatically checked outputs as reinforcement signals. Absolute Zero [32] is a representative example that employs a proposer-solver framework for self-play through executable verification in code and mathematics. While these domains offer clear-cut correctness, open-ended visual tasks typically lack reliable ways to verify model-generated supervision. We close this gap for remote sensing imagery by routing perception through executable spatial programs whose outputs serve as verifiable rewards.

3 The GeoX paradigm

3.1 Preliminaries

We establish the core mechanisms of reinforcement learning through self-play and verifiable rewards.

Reinforcement Learning with Verifiable Rewards (RLVR). In RLVR, a model generates a response y for a question x and scores it against a reference answer $y^\star$ using a verifier via exact-match or program execution. Formally, it maximizes the expected reward over a dataset of question-answer pairs $\mathcal{D}_{\text{RLVR}} = \{(x_n, y_n^\star)\}_{n=1}^N$:

$$\mathcal{J}_{\text{RLVR}}(\theta) = \mathbb{E}_{(x,y^\star)\sim\mathcal{D}_{\text{RLVR}},\, y\sim\pi_\theta(\cdot|x)}\big[r(y, y^\star)\big], \tag{1}$$

where π_θ is the policy and r is a verifier reward (e.g., $\mathbb{I}[y = y^\star]$).

This approach bypasses human preference annotations and learned reward models, delivering a direct and stable feedback signal. Whereas supervised fine-tuning (SFT) teaches a model to imitate every token of a reference response, RLVR constrains it on only the final answer, leaving the intermediate reasoning trajectory free to be explored and refined.

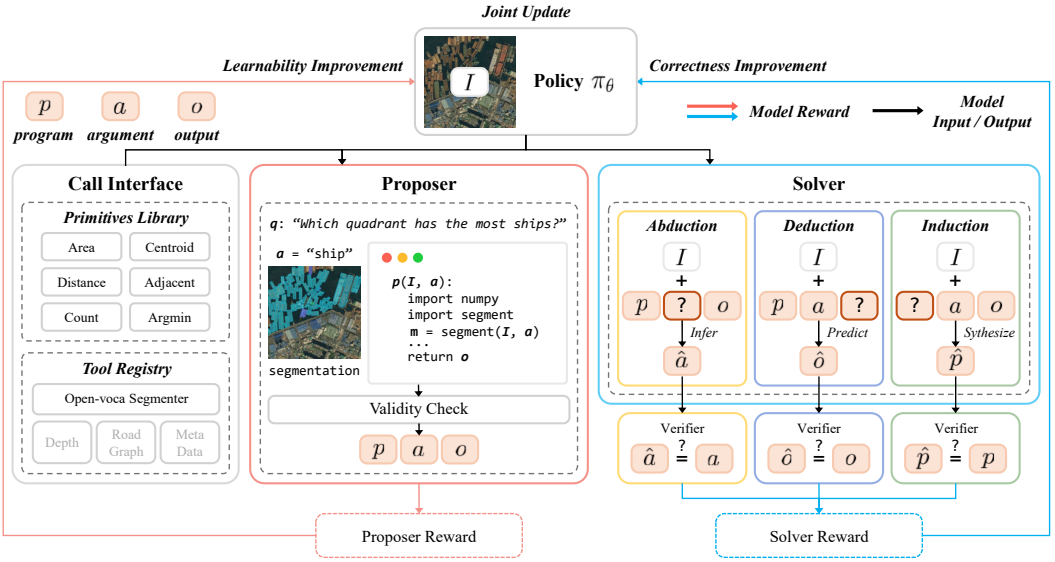


Figure 2: **Method overview.** A single multimodal policy π_θ alternates between a *proposer* (middle) and a *solver* (right) that share a *call interface* (left) of spatial primitives $\mathcal{F}$ and tools $\mathcal{T}$ (instantiated here with an open-vocabulary segmenter; greyed entries left for future work). Conditioned on image I, the proposer construct a problem by composing calls into an executable program p paired with an argument a, forming (p, a, o) with $o = p(a;\, I)$. The solver then recovers a masked element of the problem under three reasoning modes: *abduction* infers $\hat{a}$, *deduction* predicts $\hat{o}$, and *induction* synthesizes $\hat{p}$. The verifier calculates a **proposer reward** for problem learnability and a **solver reward** for answer correctness, which jointly update π_θ.

Self-play between proposer and solver. Self-play enables a model to improve by posing and addressing its own problems. In one such setup, a **proposer** and a **solver** share parameters to form a closed learning cycle [32]. The proposer constructs an executable problem, the solver attempts to solve it, and a verifier executes the program to provide a reward that optimizes both roles. Each problem is represented as a triplet:

$$(p,\, a,\, o), \qquad o = p(a), \tag{2}$$

where p is an executable program, a is its input argument, and o is its output. We use three distinct reasoning modes by masking different elements of the triplet: In *abduction*, the solver infers a from (p, o). In *deduction*, it predicts o from (p, a). In *induction*, it synthesizes p from input-output examples $\{(a_t, o_t)\}_t$.

For example, p is a program that counts ships of a specified type in a geospatial context. Here, *abduction* hypothesizes the type from an observed count with p ($o = 7 \rightarrow a = \texttt{cargo}$); s ($a = \texttt{cargo} \rightarrow o = 7$); *induction* generalizes p from input-output pairs ($\{(\texttt{cargo}, 7), (\texttt{fishing}, 3)\} \rightarrow p$). One proposal thus affords three complementary perspectives on the same scene, each engaging a particular facet of spatial reasoning.

3.2 Problem setting

We refer to the proposed method as GeoX, which trains a single multimodal policy π_θ on remote sensing imagery through self-play and verifiable rewards via program execution, as illustrated in Figure 2. The policy first acts as a **proposer**, constructing an executable problem that realizes a spatial question over an input image. It then acts as a **solver**, attempting to find the solution for constructed problems. Both roles are driven by verifiable rewards: the proposer is rewarded for devising challenging yet learnable problems, while the solver is rewarded for answering correctly under the three reasoning modes.

Let $\mathcal{I} = \{I_m\}_{m=1}^{M}$ be a collection of remote sensing images without annotations. The policy π_θ returns an answer y to a spatial question q on an image $I \in \mathcal{I}$. Each question is expressed as an

image-grounded executable problem:

$$(p,\, a,\, o,\, I), \qquad o = p(a;\, I), \tag{3}$$

where p is a program, a is its argument, and o is the output. For each image I, the proposer generates (p, a), and its execution determines o. The solver then completes a masked element of the problem from partial observations, depending on the reasoning mode.

Proposer. We maintain three problem banks $\mathcal{D}_{\text{abd}}$, $\mathcal{D}_{\text{ded}}$, and $\mathcal{D}_{\text{ind}}$ corresponding to each reasoning mode. These banks offer in-context references for the proposer.

In *abduction* and *deduction*, the proposer samples two image batches $\mathcal{B}_{\text{abd}} = \{I_b^{\text{abd}}\}_{b=1}^{B}$ and $\mathcal{B}_{\text{ded}} = \{I_b^{\text{ded}}\}_{b=1}^{B}$. It constructs one executable problem per image I using K in-context references drawn from $\mathcal{D}_{\text{abd}}$ for abduction or $\mathcal{D}_{\text{ded}}$ for deduction. Each problem consists of a question q, program p, and argument a, with output o obtained by $p(a; I)$. Only valid constructions are used for growing the respective bank, while failed (e.g., syntax or runtime errors) or non-deterministic ones (e.g., randomness, file I/O) are discarded.

In *induction*, the proposer samples B programs from $\mathcal{D}_{\text{abd}} \cup \mathcal{D}_{\text{ded}}$ and generates N_{io} input-output pairs $\{(a_t, o_t)\}_{t=1}^{N_{\text{io}}}$ for each program using references from $\mathcal{D}_{\text{ind}}$. The same grow-and-discard rule is applied.

Solver. The solver processes $3B$ total problems for three reasoning modes, where samples from each bank fill any shortfall in valid proposals. For each problem, the solver makes R attempts, yielding $3BR$ outputs per training step. In *abduction*, the solver infers the missing argument $\hat{a}$ from (I, p, o). In *deduction*, it predicts the expected output $\hat{o}$ from (I, p, a). In *induction*, it discovers the underlying program $\hat{p}$ from $(I, \{(a_t, o_t)\}_{t=1}^{N_{\text{io}}})$. The verifier scores each result in a mode-specific manner:

$$\text{Abduction:} \quad \pi_\theta^{\text{solve}}\big(\hat{a} \mid I,\, p,\, o\big), \qquad o \overset{?}{=} p(\hat{a}; I), \tag{4}$$

$$\text{Deduction:} \quad \pi_\theta^{\text{solve}}\big(\hat{o} \mid I,\, p,\, a\big), \qquad \hat{o} \overset{?}{=} p(a; I), \tag{5}$$

$$\text{Induction:} \quad \pi_\theta^{\text{solve}}\big(\hat{p} \mid I,\, \{(a_t, o_t)\}_{t\in\mathcal{V}}\big), \qquad \frac{1}{|\mathcal{U}|}\sum_{t\in\mathcal{U}} \mathbb{I}\big[o_t \overset{?}{=} \hat{p}(a_t; I)\big]. \tag{6}$$

Here, for induction, the N_{io} input-output pairs are split into a visible subset $\mathcal{V}$ and a held-out subset $\mathcal{U}$, where $\mathcal{V} \cup \mathcal{U} = \{1, \ldots, N_{\text{io}}\}$. This requires the solver to reconstruct the program rather than memorize seen pairs.

Seed initialization. To initialize this process, we define a simple hand-written seed template that calls a segmentation function to check the presence of an object (e.g., `"Is there a building?"`), as shown in Figure 3. The template is instantiated by pairing random object phrases with images, populating each bank with N_{seed} problems. From this warm start, the proposer moves toward increasingly compositional problems, such as comparing areas across object categories, without further human intervention.

SEED PROGRAM

QUESTION q

```
"Is there a building?"
```

ARGUMENT a

```
"building"
```

PROGRAM p

```
def f(I, a):
    from tools import segment
    import numpy as np
    mask = segment(I, a)
    return bool(np.any(mask))
```

OUTPUT o

```
True
```

Figure 3: **Seed problem.** A segmenter call with the phrase `"building"`, followed by a presence check on the returned mask.

3.3 Programs over image-grounded interface

The executor exposes two sets of callables: a primitives library $\mathcal{F}$ and a tool registry $\mathcal{T}$, with $\mathcal{E} = \mathcal{F} \cup \mathcal{T}$ as the full call interface. The primitives library $\mathcal{F}$ contains geometric operators (e.g., `area`, `centroid`, `orientation`) and topological operators (e.g., `distance`, `adjacent`, `overlaps`), along with aggregation operators (e.g., `count`, `argmin`, `compare`), implemented on top of NumPy, SciPy, and scikit-image.

The tool registry $\mathcal{T}$ contains specialized modules for extracting structured information from the image. A spatial program composes callables from $\mathcal{E}$, allowing primitives and tools to be interleaved as needed.

A minimal tool registry. For the experiments in this paper, we equip $\mathcal{T}$ with a single tool: an open-vocabulary segmenter that maps an image paired with a natural-language phrase (e.g., `"red-roofed building"`) to a set of segmentation masks. This minimal choice is motivated by three considerations. (1) *Spatial sufficiency*. Many spatial tasks considered in this work can be computed directly from segmentation masks using geometric, topological, and aggregation operators. (2) *Open-vocabulary flexibility*. This segmenter accepts arbitrary phrases generated by the policy, so a single tool can support diverse spatial questions without committing to a fixed class vocabulary. (3) *Methodological isolation*. The single-tool usage isolates self-play's contribution from that of a richer toolset, establishing a clean baseline against which future extensions can be evaluated.

Intermediate values are hidden from the policy. The policy π_θ accesses only I and the available calls in $\mathcal{F}$ and $\mathcal{T}$.

All intermediate values produced during execution remain internal to the executor and are inaccessible to the policy. For example, when the policy issues a segmenter call with the phrase `"solar panel array"`, neither the resulting mask array nor any values derived from it in subsequent calls, such as `count` or `area`, are exposed to the policy. This design prevents the policy from exploiting execution traces as shortcuts, forcing it to reason from the image and program specification instead.

3.4 Reward design

The proposer and solver are optimized jointly through a unified policy π_θ with mutually informative rewards.

Solver reward. The solver is rewarded for the *correctness* of its answers by a verifier. For abduction and induction, where the output is fully determined by program execution, the verifier is reduced to an execution-consistency condition that the returned output under the solver's result exactly matches o:

$$r^{\text{solve}}_{\text{abd}}(\hat{a}) = \mathbb{I}\big[p(\hat{a}; I) = o\big], \quad r^{\text{solve}}_{\text{ind}}(\hat{p}) = \frac{1}{|\mathcal{U}|}\sum_{t\in\mathcal{U}} \mathbb{I}\big[\hat{p}(a_t; I) = o_t\big]. \tag{7}$$

For deduction, by contrast, where the output may take various types across problems, the type-aware verifier is adopted in place of a binary exact-match rule:

$$r^{\text{solve}}_{\text{ded}}(\hat{o}) = \begin{cases} \max\big(0,\, 1 - |\hat{o} - o| / \max(|o|, 1)\big) & \text{numeric,} \\ \mathbb{I}[\text{norm}(\hat{o}) = \text{norm}(o)] & \text{string,} \\ \mathbb{I}[\hat{o} = o] & \text{boolean,} \\ \text{IoU}(\hat{o}, o) & \text{bounding box.} \end{cases} \tag{8}$$

Proposer reward. The proposer is rewarded for the *learnability* [32] of its proposals. The learnability of a problem is estimated from R solver rollouts, by averaging their solver rewards:

$$\bar{r}^{\text{solve}} \;=\; \frac{1}{R}\sum_{i=1}^{R} r^{\text{solve}}_{(i)}, \tag{9}$$

which then defines the proposer reward:

$$r^{\text{prop}} \;=\; \mathbb{I}\big[\bar{r}^{\text{solve}} > 0\big] \cdot \big(1 - \bar{r}^{\text{solve}}\big)\,. \tag{10}$$

The reward vanishes at both extremes ($\bar{r}^{\text{solve}} \in \{0, 1\}$) and peaks on the hardest solvable problems. An adaptive curriculum thus emerges, in which the proposer favors problems at the frontier of the solver's ability, advancing as the solver improves.

Joint optimization. The shared parameters θ are trained with a joint objective:

$$\mathcal{J}^{\text{GeoX}}(\theta) \;=\; \lambda\, \mathcal{J}^{\text{prop}}(\theta) \;+\; \mathcal{J}^{\text{solve}}(\theta), \tag{11}$$

where $\mathcal{J}^{\text{prop}}$ and $\mathcal{J}^{\text{solve}}$ are the expected proposer and solver rewards under their problem distributions. We set $\lambda = 1$ in all experiments, weighting the two roles equally.

4 Experiments

4.1 Evaluation setup

Datasets. Geospatial understanding is evaluated on three VQA benchmarks: RSVQA-HR [16], EarthVQA [25], and GEOBench-VLM [6]. From GEOBench-VLM, we use the 17 out of 31 tasks that target spatial understanding.

Baseline models. We compare our method against two groups of baselines. *(i) Base*: general-purpose VLMs evaluated zero-shot, namely Qwen-2.5-VL-7B-Instruct [26] (Qwen) and LLaVA-1.5-7B [14] (LLaVA). *(ii) Conventional*: domain-specific VLMs fine-tuned on human-curated remote sensing data, namely GeoChat [11], VHM [19], EarthDial [22], and RSThinker [15].

Evaluation procedure. Performance is measured using accuracy (%). Closed-form answers, including the five-way multiple-choice questions in GEOBench-VLM, are scored by exact match, while open-ended answers in RSVQA-HR and EarthVQA are scored by an LLM-as-judge (Qwen2.5-7B-Instruct). The judge determines whether each prediction is semantically equivalent to the reference answer using the following prompt:

```
Question: {question}
Ground truth: {gt}
Model answer: {pred}
Is the model answer equivalent to the ground-truth?
Answer with one word: "yes" or "no".
```

For each question, we generate $N_{\text{eval}} = 32$ responses and aggregate them by majority voting. All baselines are re-evaluated under a common evaluation protocol.

Additional details on datasets, baselines, implementation, and self-play prompts are provided in Appendices A, B, C, and I, respectively. Further analyses on training dynamics, object counting, data overlap, segmentation tool, number of evaluation responses, confidence intervals, and performance gain are presented in Appendix D. Full results for Sections 4.2 and 4.3 are reported in Appendix H.

4.2 Main results

Table 1 reports the performance of GeoX against strong zero-shot and conventional remote sensing VLMs. Without any curated data, GeoX improves over its base models on average (+5.5 points for LLaVA, +1.9 points for Qwen). Notably, GeoX_{Q} attains the highest average on both EarthVQA (51.2) and GEOBench-VLM (43.3), surpassing EarthDial trained on 11.1M curated QA pairs (31.9 and 31.4).

These average gains trace back to a specific group of tasks that fit naturally within our verifiable environment. This is clear in counting and measurement tasks, where GeoX consistently ranks first across all three benchmarks (Count on RSVQA-HR, Basic and Reasoning-based Counting on EarthVQA, and Object Localization & Counting on GEOBench-VLM) . The effect is also apparent in relational and measurement tasks, where GeoX tops all baselines on the relational task (Spatial Relation Classification on GEOBench-VLM (see Appendix H)) and nearly matches the best curated baseline on the measurement task (Area on RSVQA-HR) . Together, these results indicate that the current call interface is effective for tasks grounded in geometric and topological structure. It also suggests that enriching the tool registry $\mathcal{T}$ is a promising direction for closing the remaining gaps.

4.3 Ablation study

For ablation of the reasoning modes, we contrast the full model against *Singleton variants* (one mode kept) and *Drop-one variants* (one mode removed). To examine the role of self-play, we further consider two *training variants*: (i) BaseGen, which replaces problems from self-play with those from the base model (Qwen-2.5-VL-7B-Instruct) and (ii) SolvOnly, which maintains the self-play mechanism but ignores the proposer reward signals and relies solely on solver signals. The former asks whether image data alone suffices without an evolving curriculum, while the latter asks whether self-play helps without an explicit incentive for a learnable problem.

Table 2 presents performance on GEOBench-VLM with varying numbers of active reasoning modes. Among singleton variants, keeping only *Deduction* performs the best, revealing that its alignment

Table 1: **Main results on VQA benchmarks.** Performance of GeoX, instantiated from two base VLMs (LLaVA and Qwen), is compared against six baselines on RSVQA-HR, EarthVQA, and the four task categories of GEOBench-VLM. For each model, the number of human-curated QA pairs used for fine-tuning is listed; baseline models fine-tuned on fully curated data are grey shaded. The best result in each row is shown in **bold** and our results are highlighted in blue when they surpass the corresponding base (LLaVA or Qwen).

	Base		*Conventional*				*Self-play*	
Benchmark / Task	LLaVA	Qwen	GeoChat	VHM	EarthDial	RSThinker	**GeoX$_{L}$**	**GeoX$_{Q}$**
# Curated	0	0	318K	1.4M	11.1M	380K	0	0
RSVQA-HR								
Presence	57.9	60.1	57.9	**72.3**	60.0	55.4	61.9	63.2
Count	30.3	35.2	25.5	13.5	25.9	22.1	30.5	**36.6**
Area	2.7	10.3	**41.6**	13.5	38.8	22.1	40.3	16.5
Comparison	52.5	69.7	74.8	70.7	**78.9**	68.7	57.5	72.0
Average	35.9	43.8	50.0	42.5	**50.9**	42.1	47.6	47.1
EarthVQA								
Basic Judging	78.6	78.0	66.6	79.0	76.2	69.4	**82.2**	78.2
Reasoning-based Judging	55.8	**73.0**	37.0	71.8	41.4	42.6	53.8	68.6
Reasoning-based Counting	24.8	23.4	16.6	12.8	5.6	15.4	**33.8**	26.6
Basic Counting	60.6	62.4	26.0	39.6	53.0	45.8	**70.2**	69.4
Object Situation Analysis	28.2	29.4	2.2	23.2	4.0	25.2	**30.4**	30.2
Comprehensive Analysis	19.6	33.6	7.8	24.6	11.0	19.0	19.5	**34.4**
Average	44.6	50.0	26.0	41.8	31.9	36.2	48.3	**51.2**
GEOBench-VLM								
Object Localization & Counting	20.6	37.3	22.2	27.0	24.8	24.7	22.2	**39.5**
Scene Understanding	46.6	54.8	49.6	55.1	**60.5**	49.6	47.1	53.9
Object Classification	35.2	43.8	34.5	36.3	35.9	36.2	36.5	**47.5**
Event Detection	14.7	**45.3**	37.8	47.0	16.7	20.9	14.3	42.4
Average	26.2	42.1	30.3	35.4	31.4	30.0	27.4	**43.3**

Table 2: **Ablation study results on GEOBench-VLM.** Performance is reported across the four task categories of GEOBench-VLM. Columns specify which solver modes are active: A (Abduction), D (Deduction), I (Induction). *Singleton Variants* keeps one mode; *Drop-one Variants* removes one; *Training Variants* use all three reasoning modes. *Full* denotes our default configuration. The best result in each row is shown in **bold** and the second-best is shown in underline.

	Singleton Variants			*Drop-one Variants*			*Training Variants*		*Full*
Config.	Abd	Ded	Ind	−Abd	−Ded	−Ind	BaseGen	SolvOnly	GeoX
Active Modes	A	D	I	D,I	A,I	D,A	D,A,I	D,A,I	D,A,I
Object Localization & Counting	35.5	37.2	36.4	36.5	36.4	<u>37.5</u>	37.4	34.3	**39.5**
Scene Understanding	51.4	<u>52.4</u>	49.4	48.1	50.8	52.1	51.3	50.1	**53.9**
Object Classification	45.3	46.8	46.3	43.6	**49.3**	44.3	42.5	45.3	<u>47.5</u>
Event Detection	39.0	42.4	40.3	40.3	43.6	42.8	**46.9**	<u>46.6</u>	42.4
Average	39.9	<u>41.6</u>	40.3	39.8	41.3	41.5	<u>41.6</u>	39.8	**43.3**

with direct answer prediction is what helps the most. By contrast, among drop-one variants, removing *Abduction* performs the worst, implying that reasoning backward from outputs to plausible inputs provides a learning signal that neither deduction nor induction can fully substitute.

Both *BaseGen* and *SolvOnly* fall short of the full GeoX. The drop in *BaseGen* reveals that the same images yield little benefit unless the problems evolve with the solver. The drop in *SolvOnly* reveals that constructing problems is not enough unless the proposer is rewarded for making them challenging yet learnable. Overall, the two variants demonstrate that GeoX draws its strength not only from constructing its own problems but also from adapting them at the frontier of the solver's ability.

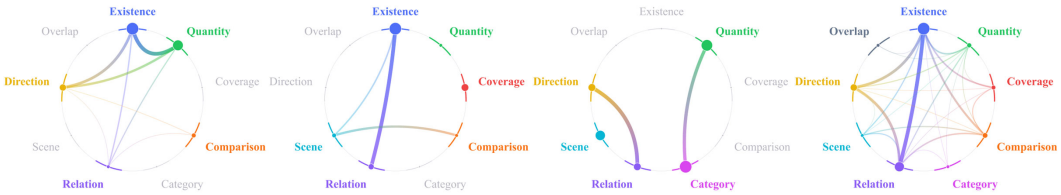


Figure 4: **Pairwise dimension compositions across datasets.** Each node represents one of nine question dimensions; node size denotes how often a dimension appears in a single problem and edge thickness denotes how often two dimensions co-occur within a single problem.

Query (q): " In which quadrant (TL, TR, BL, BR) does the centroid of the largest ship lie in the image? **"**



Argument (a): "ship"

Program (p):

```
def f(image, a):
    import numpy as np
    from scipy.ndimage import center_of_mass
    from tools import segment
    instances = segment(image, a)
    if not isinstance(instances, list) or not instances:
        return "none"
    largest = max(instances, key=lambda m:
int(np.sum(m)))
    cy, cx = center_of_mass(largest)
    h, w = largest.shape
    return ("T" if cy < h / 2 else "B") + ("L" if cx < w
/ 2 else "R")
```

Output (o): "TR" (program execution)

<think>
1. The input class is "ship".
2. In the image, there are multiple ships docked at the harbor.
3. The task is to find the largest ship and determine the quadrant where its center of mass lies.
4. From the image, the long vessel docked at the top of the image appears to be the largest ship.
5. The center of mass calculation will be applied to this ship to determine the quadrant it falls into.

Let's calculate the center of mass for the largest ship.
The coordinates of the ship in the image are approximately:
- Top-left corner: (260, 60) / Bottom-right corner: (540, 130)

The center of mass calculation for a rectangle is at the midpoint of the top-left and bottom-right corners.
The center of mass for the ship is:
- X-coordinate: (260 + 540) / 2 = 400 / Y-coordinate: (60 + 130) / 2 = 95

Since the center of mass (400, 95) is above the middle of the image (h/2 = 600/2 = 300) and to the right of the middle of the image (w/2 = 600/2 = 300), the quadrant is the top-right (TR). Therefore, the output is 'TR'.
</think>

<answer>TR**</answer>**

Answer ($\hat{o}$): "TR" (**Correct**)

Figure 5: **Qualitative analysis of deduction.** Given image I, argument $a = \texttt{"ship"}$, and program p, the solver's Chain-of-Thought follows the program's control flow and predicts expected output $\hat{o} = \texttt{"TR"}$, which exactly matches program-executed output $o = p(a; I)$.

5 Discussion

We investigate the behavior of GeoX along three complementary axes through the following questions, with extended results provided in Appendices E, F, and G.

Q. How broad is the compositional space GeoX explores? The compositional structure of the constructed problems can be represented as a graph over nine question dimensions (Existence, Quantity, Coverage, Comparison, Category, Relation, Scene, Direction, Overlap), where node size encodes single-dimension frequency and edge thickness encodes pairwise co-occurrence, as shown in Figure 4. Existing VQA benchmarks form a sparse graph that covers only a few dimensions and leaves most pairs disconnected. GeoX, in contrast, forms a markedly denser graph that populates nearly all nine dimensions and renders most pairs connected. This density highlights that proposer-solver self-play expands both the volume of training problems and the compositional richness of their distribution.

Q. What does self-play produce in practice? A qualitative deduction example illustrates the image-grounded reasoning that self-play induces in GeoX, as shown in Figure 5. The constructed problem asks in which quadrant the centroid of the largest ship lies. Its program p with argument $a = \texttt{"ship"}$ segments all instances of the target class, selects the largest by mask, estimates its center of mass, compares it against the image midlines to determine the quadrant $\{\texttt{TL}, \texttt{TR}, \texttt{BL}, \texttt{BR}\}$, and returns the output $o = p(a; I)$. Given the same image I, argument a, and program p, the solver follows the program's control flow step by step: it segments the ships docked in the harbor, selects the long

vessel that dominates the others, estimates its center of mass from the ship's extent, and compares it against the same midlines, predicting the expected output $\hat{o} =$ `"TR"`, which exactly matches the program-executed output o. Self-play thus encourages the policy to treat the executable program as a procedural scaffold grounded in image evidence, not as an opaque output generator.

Q. Which primitives are used by the proposer during self-play? Across roughly 6,500 programs constructed by the proposer over training, the usage frequency of each primitive in library $\mathcal{F}$ reveals which operations are called during self-play, as shown in Figure 6.

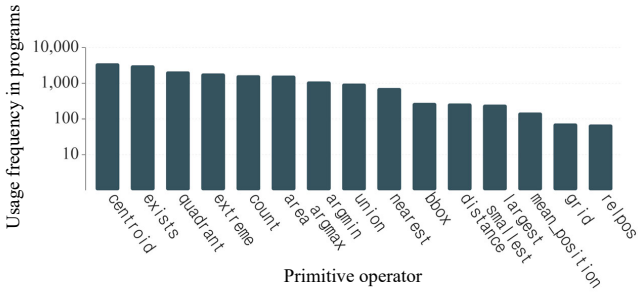


Figure 6: **Usage frequency of primitives in GeoX.** Each bar denotes the number of constructed programs that invoke a given operator in $\mathcal{F}$ (log scale), grouped by primitive type.

The most frequent primitives span all three groups: the geometric primitive `centroid` leads overall (over 3,500 programs), followed by the aggregation primitive `exists` (over 3,000) and the topological `quadrant` (over 1,800). Aggregation primitives are the most broadly used as a group, with `extreme`, `count`, `argmin/argmax`, and `union` all common. Geometric use centers on `centroid` and `area`, while topological primitives on `quadrant` for localizing objects and `nearest` for capturing inter-object proximity. This spread across all three groups, down to a long tail of rarely used primitives, reflects that GeoX discovers problem types far beyond its hand-written seed.

6 Concluding remarks

We presented a self-play framework for geospatial reasoning that learns without human-curated data. Through a verifiable proposer-solver loop, GeoX moves beyond simple pattern matching toward deep understanding of the "spatial physics" latent in overhead imagery. Rather than fitting curated answers end-to-end, it grounds reasoning in executable programs whose verifiable execution exposes the spatial structure of a scene. The results demonstrate that geospatial representations advance most effectively when learning signals align with the geometric and topological constraints of the physical world. This shift points toward a geospatial world model that treats the Earth as an environment for reasoning rather than a target for annotation. As satellite and aerial imagery scales, the demand for structured reasoning will expand, equipping us with the necessary tools to plan, monitor, and protect our evolving global landscape.

Limitations and future directions. This work has several limitations that suggest promising directions for future research. First, the current tool registry $\mathcal{T}$ holds only an open-vocabulary segmenter. While this choice helps isolate the effect of proposer-solver self-play, it restricts the present study to spatial reasoning tasks that can be verified from segmentation masks. As a result, vertical reasoning (e.g., depth and slope estimation), graph-based reasoning over road networks (e.g., connectivity and reachability), and metadata-grounded reasoning using external geospatial information (e.g., OSM tags or land-use attributes) remain outside the current scope. Expanding $\mathcal{T}$ with additional such tools, while preserving the self-play framework and its independence from human-curated data, is a natural next step. Second, the verifiable reward depends on the quality of the tools used for execution. Segmentation errors, especially on rare categories, propagate as noise through the proposer-solver loop and hinder model's improvement. Stronger verifiers, such as cross-tool agreement or self-consistency checks, is another step toward reliable rewards, reducing this dependence on any single perceptual module. Third, program execution provides verifiable answers but does not guarantee the quality of generated questions. Human validation is needed to assess their consistency with the underlying programs, answerability from the image alone, and practical relevance to remote sensing applications. Fourth, the induction mode verifies synthesized programs against held-out input–output pairs, but this does not guarantee that they capture the intended image-grounded reasoning. Explicit constraints on program structure could help address this limitation.

Acknowledgments and Disclosure of Funding

MCha was supported by the National Research Foundation of Korea (NRF) grant (RS-2022-00165347). We thank Jea Kwon, Dong-Kyum Kim, Sungwon Han, Donghyun Ahn, and Wenchao Dong for insightful discussions that helped shape this work, and the anonymous reviewers for valuable comments and suggestions.

References

[1] Donghyun Ahn, Minhyuk Song, Seungeon Lee, Yubin Choi, Jihee Kim, Sangyoon Park, Hyunjoo Yang, and Meeyoung Cha. Fine-grained socioeconomic prediction from satellite images with distributional adjustment. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3717–3721, 2023.

[2] Kyeongjin Ahn, Sungwon Han, Seungeon Lee, Donghyun Ahn, Hyoshin Kim, Jungwon Kim, Jihee Kim, Sangyoon Park, and Meeyoung Cha. Georeg: Weight-constrained few-shot regression for socio-economic estimation using llm. *arXiv preprint arXiv:2507.13323*, 2025.

[3] Kyeongjin Ahn, Sungwon Han, Sungwon Park, Jihee Kim, Sangyoon Park, and Meeyoung Cha. Generalizable disaster damage assessment via change detection with vision foundation model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27784–27792, 2025.

[4] Kyeongjin Ahn, YongHun Suh, Sungwon Han, Jeasurk Yang, Hannes TaubenbÃķck, and Meeyoung Cha. Mapping reduced accessibility to wash facilities in rohingya refugee camps with sub-meter imagery. *arXiv preprint arXiv:2511.07231*, 2025.

[5] Kanzhi Cheng, Li YanTao, Fangzhi Xu, Jianbing Zhang, Hao Zhou, and Yang Liu. Vision-language models can self-improve reasoning via reflection. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8876–8892, 2025.

[6] Muhammad Sohail Danish, Muhammad Akhtar Munir, Syed Roshaan Ali Shah, Kartik Kuckreja, Fahad Shahbaz Khan, Paolo Fraccaro, Alexandre Lacoste, and Salman Khan. GEOBench-VLM: Benchmarking vision-language models for geospatial tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7132–7142, 2025.

[7] Yihe Deng, Pan Lu, Fan Yin, Ziniu Hu, Sheng Shen, Quanquan Gu, James Zou, Kai-Wei Chang, and Wei Wang. Enhancing large vision language models with self-training on image comprehension. *Advances in Neural Information Processing Systems*, 37:131369–131397, 2024.

[8] Max J. Egenhofer and Robert D. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[9] Ritwik Gupta, Bryce Goodman, Nirav Patel, Ricky Hosfelt, Sandra Sajeev, Eric Heim, Jigar Doshi, Keane Lucas, Howie Choset, and Matthew Gaston. Creating xbd: A dataset for assessing building damage from satellite imagery. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 10–17, 2019.

[10] Yuan Hu, Jianlong Yuan, Congcong Wen, Xiaonan Lu, Yu Liu, and Xiang Li. Rsgpt: A remote sensing vision language model and benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 224:272–286, 2025.

[11] Kartik Kuckreja, Muhammad Sohail Danish, Muzammal Naseer, Abhijit Das, Salman Khan, and Fahad Shahbaz Khan. Geochat: Grounded large vision-language model for remote sensing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 27831–27840, 2024.

[12] Sumin Lee, Sungwon Park, Jeasurk Yang, Jihee Kim, and Meeyoung Cha. Generalizable slum detection from satellite imagery with mixture-of-experts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 38826–38834, 2026.

[13] Kaiyu Li, Shengqi Zhang, Yujie Wang, Yupeng Deng, Zhi Wang, Deyu Meng, and Xiangyong Cao. Segearth-ov3: Exploring sam 3 for open-vocabulary semantic segmentation in remote sensing images. *arXiv preprint arXiv:2512.08730*, 2025.

[14] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.

[15] Jiaqi Liu, Lang Sun, Ronghao Fu, and Bo Yang. Towards faithful reasoning in remote sensing: A perceptually-grounded geospatial chain-of-thought for vision-language models. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=lJ7zecny2e.

[16] 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. doi: 10.1109/TGRS.2020.2988782.

[17] Yang Long, Yiping Gong, Zhifeng Xiao, and Qing Liu. Accurate object localization in remote sensing images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(5):2486–2498, 2017.

[18] Dilxat Muhtar, Zhenshi Li, Feng Gu, Xueliang Zhang, and Pengfeng Xiao. Lhrs-bot: Empowering remote sensing with vgi-enhanced large multimodal language model. In *European Conference on Computer Vision*, pages 440–457. Springer, 2024.

[19] Chao Pang, Xingxing Weng, Jiang Wu, Jiayu Li, Yi Liu, Jiaxing Sun, Weijia Li, Shuai Wang, Litong Feng, Gui-Song Xia, et al. Vhm: Versatile and honest vision language model for remote sensing image analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 6381–6388, 2025.

[20] Qian Shi, Da He, Zhengyu Liu, Xiaoping Liu, and Jingqian Xue. Globe230k: A benchmark dense-pixel annotation dataset for global land cover mapping. *Journal of Remote Sensing*, 3: 0078, 2023.

[21] Minhyuk Song, Sungwon Han, Seungeon Lee, Donghyun Ahn, Jihee Kim, and Meeyoung Cha. Measuring fine-grained urban air temperature with satellite imagery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28397–28404, 2025.

[22] Sagar Soni, Akshay Dudhane, Hiyam Debary, Mustansar Fiaz, Muhammad Akhtar Munir, Muhammad Sohail Danish, Paolo Fraccaro, Campbell D Watson, Levente J Klein, Fahad Shahbaz Khan, et al. Earthdial: Turning multi-sensory earth observations to interactive dialogues. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14303–14313, 2025.

[23] Di Wang, Jing Zhang, Bo Du, Minqiang Xu, Lin Liu, Dacheng Tao, and Liangpei Zhang. Samrs: Scaling-up remote sensing segmentation dataset with segment anything model. *Advances in Neural Information Processing Systems*, 36:8815–8827, 2023.

[24] Junjue Wang, Zhuo Zheng, Ailong Ma, Xiaoyan Lu, and Yanfei Zhong. Loveda: A remote sensing land-cover dataset for domain adaptive semantic segmentation. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran Associates, Inc., 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/4e732ced3463d06de0ca9a15b6153677-Paper-round2.pdf.

[25] Junjue Wang, Zhuo Zheng, Zihang Chen, Ailong Ma, and Yanfei Zhong. EarthVQA: Towards queryable earth via relational reasoning-based remote sensing visual question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 5481–5489, 2024. doi: 10.1609/aaai.v38i6.28357.

[26] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

[27] Jeasurk Yang, Donghyun Ahn, Junbeom Bahk, Sungwon Park, Nurrokhmah Rizqihandari, and Meeyoung Cha. Assessing climate risks from satellite imagery with machine learning: A case study of flood risks in jakarta. *Climate Risk Management*, 46:100651, 2024.

[28] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. Star: Self-taught reasoner bootstrapping reasoning with reasoning. In *Proc. the 36th International Conference on Neural Information Processing Systems*, volume 1126, pages 0–55, 2024.

[29] Dengke Zhang, Fagui Liu, and Quan Tang. Corrclip: Reconstructing patch correlations in clip for open-vocabulary semantic segmentation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 24677–24687. IEEE, 2025.

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

[31] Yuanlin Zhang, Yuan Yuan, Yachuang Feng, and Xiaoqiang Lu. Hierarchical and robust convolutional neural network for very high-resolution remote sensing object detection. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5535–5548, 2019.

[32] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *Advances in Neural Information Processing Systems*, 2025.

A Datasets

This section describes the datasets used for training and evaluating GeoX. Table 3 summarizes imagery source, geographically covering region, spatial resolution, and size.

A.1 Training dataset

For training, GeoX draws images from two remote sensing datasets, SAMRS-FAST [23] and Globe230k [20]. Because our method constructs question-answer pairs through self-play, training uses only the image collections from these datasets, without relying on their annotations.

SAMRS-FAST is a high-resolution dataset for fine-grained object detection across 37 categories, derived from FAIR1M-2.0 aerial imagery (0.3–0.8 m GSD) collected globally.

Globe230k is a large-scale dataset for land cover semantic segmentation, derived from Google Earth imagery (1 m GSD) sampled worldwide.

Table 3: **Summary of training and evaluation datasets.** Data is characterized by source, geographic coverage, resolution (ground sample distance per pixel), and size. Evaluation is zero-shot for GeoX, which is trained without any question-answer pair from the evaluation benchmarks. Sizes for the evaluation benchmarks correspond to their test splits.

	Imagery Source	**Region**	**Resolution**	**# Images**
Training				
SAMRS-FAST	FAIR1M-2.0	Predominantly China	0.3–0.8 m	64,147
Globe230k	Google Earth	Worldwide	1 m	232,819
VQA Evaluation				
RSVQA-HR	USGS HRO	USA (3 cities)	0.15 m	9,505 (2,226 test)
EarthVQA	Google Earth	China (3 cities)	0.3 m	6,000 (1,809 test)
GEOBench-VLM	Mixed	Worldwide	Varies	2,503
Object Counting Evaluation				
HRRSD	Google Earth, Baidu Map	Worldwide	0.15–1.2 m	21,761 (10,943 test)
RSOD	Google Earth, Tianditu	Worldwide	0.3–3 m	976

A.2 Evaluation dataset for VQA

We evaluate the visual question answering capability of GeoX on three remote sensing VQA benchmarks: RSVQA-HR [16], EarthVQA [25], and GEOBench-VLM [6].

RSVQA-HR uses USGS High-Resolution Orthoimagery (0.15 m GSD, captured by low-altitude flight missions) over three USA cities. Its QA pairs are template-generated from OpenStreetMap annotations and span four types: presence, comparison, counting, and area.

EarthVQA uses Google Earth imagery (0.3 m GSD) covering 18 districts in China [24]. Its QA pairs span six question types, from auto-generated basic judging and counting to manually annotated relational analysis for urban planning.

GEOBench-VLM draws from a heterogeneous mixture of source datasets including DOTA, DIOR, xBD, FAIR1M, fMoW, RarePlanes, and FireRisk, across optical, bi-temporal, and SAR modalities. Its QA pairs are formatted as 5-way multiple-choice items, covering 8 broad categories and 31 fine-grained sub-tasks. We evaluate 17 sub-tasks most relevant to geospatial reasoning, categorized into object localization & counting, scene understanding, object classification, and event detection.

A.3 Evaluation dataset for Object Counting

We evaluate the object counting capability of GeoX on two remote sensing object detection benchmarks: HRRSD [31] and RSOD [17].

HRRSD uses Google Earth and Baidu Map imagery (0.15–1.2 m GSD) covering diverse regions worldwide. It contains 21,761 images with 55,740 annotated instances spanning 13 object categories, including vehicles, aircraft, ships, and sports facilities.

RSOD uses Google Earth and Tianditu imagery (0.3–3 m GSD) across heterogeneous terrain worldwide. It contains 976 images with 6,950 annotated instances across four object categories of aircraft, oil tank, overpass, and playground.

B Baselines

This section provides additional details on each baseline model used in our comparison, grouped into *base* general-purpose VLMs and *conventional* geospatial VLMs fine-tuned on curated remote sensing data.

B.1 Base VLMs

Qwen-2.5-VL-7B-Instruct [26] and LLaVA-1.5-7B [14] are used as base VLMs, serving both as the initialization for our self-play training and as zero-shot baselines for evaluation.

Qwen-2.5-VL-7B-Instruct is a general-purpose vision-language model from the Qwen-2.5-VL family that combines the Qwen-2.5 language backbone with a Vision Transformer encoder. We evaluate it zero-shot, without any remote-sensing-specific adaptation.

LLaVA-1.5-7B is a general-purpose vision-language model that connects a Vicuna-1.5 language backbone with a CLIP visual encoder, trained on visual instruction-tuning data drawn primarily from natural images. We evaluate it zero-shot, without any remote-sensing-specific adaptation.

B.2 Conventional Remote Sensing VLMs

We compare GeoX against four geospatial VLMs fine-tuned on curated remote sensing data: GeoChat [11], VHM [19], EarthDial [22], and RSThinker [15].

GeoChat adapts LLaVA-1.5-7B to remote sensing through LoRA fine-tuning on roughly 318K instruction-following pairs. It supports high-resolution imagery and region-level prompts.

VHM fine-tunes LLaVA-1.5-7B on 1.4M image-text pairs with rich-content captions and an honest instruction dataset that pairs factual questions with deceptive ones.

EarthDial extends InternVL2 to multi-sensor and multi-temporal imagery via instruction tuning on over 11.11M pairs.

RSThinker trains a base VLM through a two-stage alignment strategy on a 380K-sample dataset of geospatial Chain-of-Thought rationales, producing both a final answer and a verifiable analytical trace.

C Additional implementation details

This section provides additional implementation details on the training setup of GeoX, including model initialization, training data, execution environment, self-play configuration, optimization, and compute.

Setup. We initialize GeoX from a base vision-language model, using either Qwen-2.5-7B-Instruct or LLaVA-1.5-7B depending on the experiment. Training uses images drawn from two remote sensing datasets, SAMRS-FAST [23] and Globe230k [20]. Because GeoX generates supervision through self-play, we use only the image collections from these datasets and do not rely on their annotations. The execution interface consists of a primitives library $\mathcal{F}$ and a tool registry $\mathcal{T}$, with callable interface $\mathcal{E} = \mathcal{F} \cup \mathcal{T}$. The primitives library $\mathcal{F}$ implements deterministic geometric, topological, and aggregation operations using NumPy, SciPy, and scikit-image. The tool registry $\mathcal{T}$ contains a single open-vocabulary segmenter, SegEarth-OV3 [13], which returns image-grounded masks used by executable programs. Here, "zero curated data" refers to the absence of human-curated QA pairs during training, rather than the absence of hand-designed interface.

Self-play configuration. We maintain three mode-specific banks, $\mathcal{D}_{\text{abd}}$, $\mathcal{D}_{\text{ded}}$, and $\mathcal{D}_{\text{ind}}$, which provide in-context references for proposal and refill failed generations. Each bank begins with $N_{\text{seed}} = 100$ seed problems, as described in Section 3.3. At each proposal step, we use batch size $B = 32$ per mode. For abduction and deduction, the proposer constructs one image-grounded executable problem per sampled image, conditioned on $K = 6$ in-context references from the corresponding bank. For induction, N_{io}=6 input-output pairs are prepared for each program. These pairs are split evenly into a visible subset used by the solver to synthesize the program and a held-out subset used to score it. For each newly proposed problem, the solver makes R=8 attempts whose averaged reward defines the problem's learnability, which in turn defines the proposer reward r^{prop} (Eq. 10). The solver itself is trained on a separately sampled batch of B problems from the corresponding bank, where each rollout's prediction accuracy, scored by the executor, defines the solver reward r^{solve} (Eq. 7 and 8).

SFT warm-up. Before RL through self-play, we optionally run a brief SFT stage that teaches π_θ's output schema only: the proposer's outputs (question q, argument a, and program p) and the solver's `<think>...</think>` / `<answer>...</answer>` format. Correctness across the three reasoning modes, use of the call interface $\mathcal{E} = \mathcal{F} \cup \mathcal{T}$, and proposal learnability are all left to RL, making this stage a parser warm-up rather than a knowledge injection. The corpus is built by running the self-play loop with the policy frozen and supervision supplied by the executor. π_θ proposes problems for abduction and deduction, and we retain those for which $o = p(a; I)$ executes deterministically. For the solver, we keep π_θ's `<think>` traces but replace `<answer>` with the executor-verified o. Induction is omitted from the warm-up to keep its data construction simple. The resulting corpus ($\sim$1K proposer and $\sim$3K solver examples) LoRA-fine-tunes the base model (r=32, α=64) for 2 epochs at learning rate $5\text{e}{-5}$.

Optimization and compute. We optimize the policy via reinforcement learning using Task-Relative REINFORCE++ [32]. Each pair of role and reasoning mode is treated as a separate task, yielding six tasks in total (2 roles $\times$ 3 modes). During each update, the six tasks are sampled in equal proportion. We set the weight between proposer and solver objectives to $\lambda = 1$, giving equal weight to proposer and solver rewards. We train with the AdamW optimizer, learning rate $5\text{e}{-7}$, weight decay 0, gradient clipping threshold 1.0, and a constant learning rate schedule with no warm-up. The sampling temperature is set to 1.0 for proposal and 1.0 for solving. Unless otherwise stated, we follow the training settings of prior work [32]. A full self-play run consists of 150 steps and takes approximately 60 hours on 4 NVIDIA H200 GPUs.

D Additional experimental results

This section presents additional experimental results due to space constraints.

D.1 Training dynamics during self-play

We show the training dynamics of GeoX through VQA benchmarks: RSVQA-HR, EarthVQA, and GEOBench-VLM. For each benchmark, we select the subtask most closely tied to geospatial reasoning among its available evaluation categories: *Comparison* for RSVQA-HR, *Reasoning-based Counting* for EarthVQA, and *Spatial Relation Classification* for GEOBench-VLM. Figure 7 shows the accuracy (%) of GeoX as a function of training steps. Across all three subtasks, accuracy exhibits a consistent upward trend as self-play proceeds, indicating that the proposer-solver loop progressively strengthens the model's spatial reasoning capability. The vertical axis is scaled to the dynamic range of each task to make per-task trends visible.



(a) RSVQA-HR (Comparison) (b) EarthVQA (Reasoning-based Counting) (c) GEOBench-VLM (Spatial Relation Classification)

Figure 7: **Training dynamics during self-play.** Task accuracy of GeoX over training steps on representative geospatial reasoning subtasks drawn from three remote sensing VQA benchmarks: *Comparison* (RSVQA-HR), *Reasoning-based Counting* (EarthVQA), and *Spatial Relation Classification* (GEOBench-VLM).

D.2 Evaluation on Object Counting

Beyond the VQA results in Section 4.2, we further evaluate GeoX on object counting. Datasets, baselines details follow Section 4.1, Appendix A, and Appendix B. Table 4 reports the performance of GeoX against strong zero-shot and conventional remote sensing VLMs. Note that RSThinker's entries on HRRSD are marked with †, as HRRSD overlaps with its training distribution and is therefore excluded from the bold comparison. Both GeoX_L and GeoX_Q improve over their respective base models in average accuracy.

Table 4: **Results on Object Counting benchmarks.** Performance of GeoX, instantiated from two base VLMs (LLaVA and Qwen), is compared against six baselines on HRRSD and RSOD. Baseline models fine-tuned on fully human-curated data are grey shaded. The best result in each row is shown in **bold**, and our results are highlighted in blue when they surpass the corresponding base (LLaVA or Qwen). †HRRSD lies in RSThinker's training distribution, so its numbers (and any average that includes HRRSD) are not directly comparable to the zero-shot evaluations of the other rows and are excluded from **bold** consideration.

		Base		*Conventional*				*Self-play*	
Dataset	**Metric**	LLaVA	Qwen	GeoChat	VHM	EarthDial	RSThinker	**GeoX_L**	**GeoX_Q**
HRRSD	Acc ↑	55.6	60.5	52.4	62.5	59.6	85.5†	55.0	**63.1**
	MAE ↓	1.03	0.73	1.50	0.79	0.87	0.20†	1.00	**0.72**
RSOD	Acc ↑	33.3	27.8	16.0	27.1	10.4	37.3	**37.5**	30.6
	MAE ↓	4.49	**3.31**	8.17	8.47	8.83	6.19	4.32	3.69
Average	Acc ↑	44.5	44.2	34.2	44.8	35.0	61.4†	46.3	**46.8**
	MAE ↓	2.76	**2.02**	4.84	4.63	4.85	3.20†	2.66	2.21

D.3 Training and evaluation image overlap

We cross-check the 296,966 training images against all five evaluation benchmarks to identify potential image overlap. Among the 466 images in the Spatial Relation Classification subtask of GEOBench-VLM, 92 overlap with the training data, as both this subtask and SAMRS-FAST are derived from FAIR1M. No overlapping images are found in the other four benchmarks.

To assess the effect of this overlap, we evaluate the models after excluding the 92 overlapping images. Table 5 reports the performance on the original and overlap-free evaluation sets.

Table 5: **Effect of training–evaluation image overlap on GEOBench-VLM.** Performance on Spatial Relation Classification is compared before and after excluding 92 overlapping images from evaluation. The best result in each row is shown in **bold**.

	Base		Self-play	
Evaluation set	LLaVA	Qwen	GeoX_L	GeoX_Q
Original (466 images)	35.2	68.2	36.3	**70.0**
Overlap-free (374 images)	35.9	69.5	37.4	**71.2**

The performance gains over its respective base models are preserved after excluding the overlapping images (+1.1 to +1.5 points for LLaVA, +1.8 to +1.7 points for Qwen). These results indicate that the observed improvements are not primarily driven by training–evaluation image overlap.

D.4 Sensitivity to the segmentation tool

We examine the sensitivity of GeoX to the segmentation tool used during self-play. As described in Section 3.3, the segmenter is used only to compute rewards during training, with intermediate execution values hidden from the policy and no external tools are used at inference.

To evaluate this sensitivity, two variants of GeoX are considered alongside the default SegEarth-OV3 [13] setup: one trained with CorrCLIP [29], a weaker open-vocabulary segmenter on remote sensing imagery (40.7 vs. 53.4 average mIoU across eight benchmarks), and another with cross-tool agreement. Table 6 reports the results on RSVQA-HR.

Table 6: **Sensitivity to the segmentation tool on RSVQA-HR.** Performance of GeoX under different segmentation setups during self-play is compared against its base model. The best result in each row is shown in **bold**.

	Base	*Self-play*		
Task	Qwen	GeoX_Q	GeoX_Q	GeoX_Q
Segmentation Setup	–	CorrCLIP	Cross-tool	SegEarth-OV3
Presence	60.1	62.8	62.8	**63.2**
Count	35.2	36.0	36.1	**36.6**
Area	10.3	16.8	**17.0**	16.5
Comparison	69.7	71.8	71.7	**72.0**
Average	43.8	46.9	46.9	**47.1**

Despite its lower segmentation quality, GeoX trained with CorrCLIP improves over the base model by 3.1 points on average, approaching the performance achieved with SegEarth-OV3 (46.9 vs. 47.1). Cross-tool agreement also improves over the base model by 3.1 points, matching CorrCLIP (46.9). The comparable performance across these settings supports the single-tool design as a practical choice, avoiding the additional execution cost of cross-tool verification.

D.5 Sensitivity to the number of evaluation responses

We assess how the number of responses used for majority voting affects VQA accuracy. For each question, we aggregate the first $N_{\text{eval}} \in \{1, 2, 4, 8, 16, 32\}$ of the 32 responses generated during

evaluation. Table 7 reports the average accuracy of GeoX and its base model across three VQA benchmarks.

Table 7: **Sensitivity to the number of evaluation responses.** Performance of GeoX is compared against its base model across different numbers of responses used for majority voting. The best result in each row is shown in **bold**.

		N_{eval}					
Benchmark	**Model**	1	2	4	8	16	32
RSVQA-HR	Qwen	41.9	42.6	43.2	43.5	43.7	**43.8**
	GeoX_{Q}	**49.5**	48.7	47.9	47.2	47.1	47.1
EarthVQA	Qwen	48.1	48.9	49.4	49.9	49.7	**50.0**
	GeoX_{Q}	**53.6**	52.7	51.9	51.5	51.3	51.2
GEOBench-VLM	Qwen	40.3	41.5	42.0	**42.2**	41.9	42.1
	GeoX_{Q}	**45.5**	44.8	43.9	43.1	43.4	43.3

GeoX outperforms its base model at every N_{eval} across all three benchmarks. The gains are largest with a single response (+7.6, +5.5, and +5.2 points on RSVQA-HR, EarthVQA, and GEOBench-VLM, respectively) and remain positive at the default $N_{\text{eval}} = 32$ (+3.3, +1.2, and +1.2 points). These results show that the gains are robust across different values of N_{eval}, with the default setting of 32 responses providing a more conservative comparison than single-response evaluation.

D.6 Confidence intervals for VQA accuracy

We estimate 95% bootstrap confidence intervals for VQA accuracy by resampling evaluation items 10,000 times, stratified by subtask. Following the evaluation procedure in Section 4.1, each item's prediction is determined by majority voting over $N_{\text{eval}} = 32$ responses. Table 8 reports the confidence intervals for the average accuracy of GeoX and its base model, together with their performance differences.

Table 8: **Bootstrap confidence intervals on VQA benchmarks.** Performance of GeoX is compared against its base model. Average accuracy and confidence intervals are reported, with Δ denoting the performance gain. The best result in each row is shown in **bold**.

	Base	Self-play	Improvement
Benchmark	Qwen	GeoX_{Q}	Δ (95% CI)
RSVQA-HR	43.8 (41.7–45.9)	**47.1** (45.1–49.3)	+3.3 (+2.1–+4.4)
EarthVQA	50.0 (47.6–52.2)	**51.2** (48.8–53.4)	+1.2 (+0.3–+2.1)
GEOBench-VLM	42.1 (39.9–43.7)	**43.3** (42.1–44.9)	+1.2 (-0.1–+2.7)

The confidence intervals for the performance gains exclude zero on RSVQA-HR and EarthVQA, while the interval on GEOBench-VLM includes zero. However, for the Object Localization & Counting and Object Classification categories of GEOBench-VLM (12 subtasks), the improvement reaches 2.9 points with a 95% confidence interval of [+1.3, +4.4]. These intervals capture uncertainty across evaluation items rather than variability across independent training runs.

D.7 Understanding of performance gains

The gains of GeoX are investigated against three potential shortcuts: image-independent patterns in constructed problems, scene-level object priors, and answer-frequency biases. These possibilities are probed through image–problem correspondence, counterfactual occlusion, and answer-distribution comparison, respectively.

Image–problem correspondence. We train GeoX by providing the solver with images mismatched to the constructed problems during self-play. Table 9 compares this variant with the original matched-image configuration and its base model across three VQA benchmarks.

Table 9: **Effect of image–problem correspondence on VQA benchmarks.** Performance of GeoX trained with matched and mismatched images is compared against its base model. The best result in each row is shown in **bold**.

	Base	Self-play	
Benchmark	Qwen	GeoX_Q	GeoX_Q
Solver Image	–	Mismatched	Matched
RSVQA-HR	43.8	44.3	**47.1**
EarthVQA	50.0	50.6	**51.2**
GEOBench-VLM	42.1	42.6	**43.3**

Training with mismatched images reduces the gains over the base model across all three benchmarks (RSVQA-HR: +3.3 to +0.5 points; EarthVQA: +1.2 to +0.6 points; GEOBench-VLM: +1.2 to +0.5 points). The substantial reductions highlight the importance of image–problem correspondence during self-play, beyond the regularities of the constructed problems themselves.

Counterfactual occlusion. We apply counterfactual occlusion on EarthVQA to separate visual evidence from scene-level object priors. For 1,091 images with positive building-presence labels, all annotated buildings are covered with gray bounding boxes. This intervention changes the expected answer from "yes" to "no" while largely preserving the surrounding scene. Table 10 reports the results before and after occlusion.

Table 10: **Counterfactual occlusion on EarthVQA.** Performance of GeoX is compared against its base model before and after occluding buildings in 1,091 images. The best result in each row is shown in **bold**.

	Base	Self-play
Condition	Qwen	GeoX_Q
Before occlusion (Yes)	87.5	**91.6**
After occlusion (No)	83.2	**85.0**

Both models maintain high accuracy after occlusion, with GeoX outperforming Qwen both before (91.6 vs. 87.5) and after (85.0 vs. 83.2). This observation supports the view that the predictions are grounded in visual observations rather than scene-level object priors alone.

Answer-distribution comparison. We compare the answer distributions of problems constructed during self-play with those of RSVQA-HR and examine the performance across answer categories. For presence questions, "yes" is the majority answer during self-play (65.7%), whereas "no" is the majority in RSVQA-HR (60.8%). Despite this difference, GeoX improves over its base model on both "yes" (+6.5 points) and "no" (+0.4 points). For counting questions, counts of five or more are the majority answer during self-play (59.6%), whereas zero is the majority answer in RSVQA-HR (60.6%). Similarly, GeoX improves over its base model across several count ranges (0: +0.7 points; 1: +4.4 points; 5–10: +16.7 points). These contrasting distributions and answer-specific gains provide evidence against simple memorization of answer frequencies in the constructed problems.

E Compositional space analysis

This section supplements the details behind the compositionality analysis presented in Section 5 (Figure 4). We describe (i) the definition of nine question dimensions used to characterize each problem, (ii) the deterministic mapping rules that assign dimensions from two distinct problem representations (natural-language questions in existing VQA benchmarks and executable Python programs constructed by GeoX during self-play), and (iii) representative examples illustrating the rules.

E.1 Nine question dimensions

We adopt a nine dimensions of atomic geospatial concept to make problems from heterogeneous representations comparable. The dimensions are designed to satisfy two desiderata:

1. **Primitive coverage.** Every primitive in the call interface $\mathcal{F}$ (Section 3.3) is assigned to exactly one dimension, ensuring that any program constructed by the proposer is representable in the nine dimensions.
2. **Benchmark coverage.** Every question in our evaluation benchmarks (RSVQA-HR, EarthVQA, GEOBench-VLM) is assigned to at least one dimension, ensuring a fair side-by-side comparison.

Each problem is mapped to a *set* of dimensions rather than a single one, since real spatial questions routinely combine multiple atomic geospatial concepts. For instance, "*which class occupies the largest area?*" invokes both COVERAGE (pixel area) and COMPARISON (extremum selection).

Table 11 summarizes the nine dimensions together with their definitions and matching rule for the two problem representations.

Table 11: **Nine question dimensions for compositional analysis.** Each problem is mapped to a set of dimensions via deterministic rules. The *question rule* applies to natural-language questions in existing VQA benchmarks (RSVQA-HR, EarthVQA, GEOBench-VLM); the *code rule* applies to executable programs constructed by GeoX.

Dimension	Definition	Question rule	Code rule
EXISTENCE	Presence of an entity	*are there*, *is there*, *any*	`np.any`, `.any()`, `is not None`
QUANTITY	Cardinality of a class	*how many*, *number of*	`len()`, `np.count_nonzero`
COVERAGE	Pixel area of a region	*area of*, *ratio of*, *how large*	`np.sum`, `.sum()`, identifier `area`
COMPARISON	Extremum selection or ordering	*larger/smaller*, *most/least*, *X or Y*	`max`, `min`, `argmax`, `argmin`
CATEGORY	Reference to an object class	*what type of object/aircraft*	String literal of an object class (e.g., `"vehicle"`)
RELATION	Continuous spatial proximity	*near*, *next to*, *between*, *adjacent*	`center_of_mass`, `distance_transform`, `cdist`
SCENE	Reference to a scene-type class	*land use*, *rural/urban*, *what kind of scene*	String literal of a scene class (e.g., `"water"`)
DIRECTION	Cardinal or quadrant assignment	*north/south/east/west*, *top of*, *where is*	String literal in {`north`, `TL`, `NW`, ... }
OVERLAP	Set-theoretic mask interaction	*both*, *intersection*, *overlap*	`np.logical_and`, `m & m`, `m | m`

E.2 Deterministic mapping rules

For each problem p, we compute its dimension set $\mathcal{C}(p) \subseteq \{\text{EXISTENCE}, \ldots, \text{OVERLAP}\}$ via a representation-specific procedure. Here, problem includes both natural-language questions presented by benchmarks and the executable Python program constructed through self-play of GeoX.

Natural-language questions (VQA benchmarks). For a question q in RSVQA-HR, EarthVQA, or GEOBench-VLM, we apply regular expressions to the lower-cased text and assign every matched dimension to $\mathcal{C}(q)$. Two overriding rules resolve common ambiguities found in practice:

- **Comparison overrides Existence.** Questions such as "*Are there more vehicles than ships?*" contain the trigger `are there` but are genuinely comparative. We therefore drop EXISTENCE whenever COMPARISON is also matched.
- **Scene overrides Existence.** Questions such as "*Is this a rural or urban scene?*" perform scene-type classification rather than presence detection. We drop EXISTENCE whenever SCENE is matched and no explicit existence trigger (`are there`, `is there`, `any`) is present.

Executable programs (GeoX). For a program p produced by the proposer, we follow the procedure below:

1. For each *operational* dimension

$$d \in \{\text{EXISTENCE}, \text{QUANTITY}, \text{COVERAGE}, \text{COMPARISON}, \text{RELATION}, \text{OVERLAP}\},$$

   add d to $\mathcal{C}(p)$ if any associated regular expression matches a substring of s.
2. Add CATEGORY if p contains a string literal naming one of the 19 object classes (e.g. `vehicle`, `ship`, `building`, `tree`).
3. Add SCENE if p contains a string literal naming one of the 11 scene classes (`road`, `water`, `parking_lot`, `vegetation`, `forest`, `grass`, `cropland`, `land`, `bareland`, `intersection`, `roundabout`) or the identifier `scene_classes`.
4. Add DIRECTION if p contains a string literal drawn from the set of 18 direction tokens (cardinal directions and quadrant abbreviations).

Verification. We verified the procedure by manually annotating a held-out sample of 100 problems (25 from each of RSVQA-HR, EarthVQA, GEOBench-VLM, and GeoX) and observed that every problem agreed on dimension membership.

E.3 Illustrative examples

Figure 8 and Figure 9 illustrate the mapping rules across both problem representations.

```
Q: "Are there more buildings than roads in this image?"
# 'more ... than' -> Comparison
# 'are there' overridden by Comparison rule
```

(a) Example of COMPARISON + QUANTITY problem in RSVQA-HR dataset

```
Q: "What kind of scene is shown?"
# 'what kind of scene' -> Scene
```

(b) Example of SCENE problem in EarthVQA dataset

```
Q: "How many cargo ships are visible in the image?"
# 'how many' -> Quantity
# 'cargo ship' (object class) -> Category
```

(c) Example of QUANTITY + CATEGORY problem in GEOBench-VLM dataset

Figure 8: Examples of rule-based mapping of natural language problems.

```
def f(image, a):
    x1, x2 = a # x = ("water", "vegetation")
    return x1 if np.sum(segment(image, x1)) > np.sum(segment(image, x2)) else x2
# np.sum(.) -> Coverage
# > with selection -> Comparison
# "water", ... -> Scene
```

(a) Example of COVERAGE + COMPARISON + SCENE problem constructed in GeoX self-play

```
def f(image, a):
    x1, x2 = a # ("vehicle", "road")
    m2 = segment(image, x2)
    dt = distance_transform_edt(~m2)
    m1 = segment(image, x1)
    i = int(np.argmin([dt[ndi.center_of_mass(m)] for m in m1]))
    cy, cx = ndi.center_of_mass(m1[i])
    return ("TL" if cx < W/2 and cy < H/2 else
            "TR" if cy < H/2 else
            "BL" if cx < W/2 else "BR")
# "vehicle" -> Category
# distance_transform_edt, center_of_mass -> Relation
# "TL", "TR", ... -> Direction
```

(b) Example of CATEGORY + RELATION + DIRECTION problem constructed in GeoX self-play

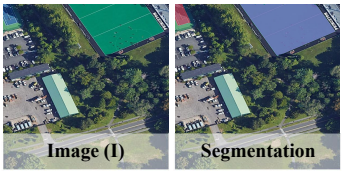
Figure 9: Examples of rule-based mapping of problems constructed in GeoX self-play.

F Qualitative analysis of the three reasoning modes

This section supplements the qualitative analysis presented in Section 5 (Figure 5). We extend the single deduction example to all three reasoning modes, with one representative example per mode drawn from a distinct image and program.

For each mode, we present the visible elements of the triplet (p, a, o), the solver's Chain-of-Thought, and its prediction verified against the program-executed output. In **abduction** (Figure 10), given (I, p, o), the solver simulates p forward and searches for an argument $\hat{a}$ whose execution reproduces o. In **deduction** (Figure 11), given (I, p, a), the solver follows the control flow of p and predicts $\hat{o}$; this example is reproduced from the main paper for direct comparison. In **induction** (Figure 12), given input-output pairs $\{(a_t, o_t)\}_{t \in \mathcal{V}}$, the solver synthesizes $\hat{p}$ consistent with the visible pairs. Across all modes, the solver's reasoning is anchored in concrete image regions rather than treating p as an opaque output generator, indicating that self-play encourages p to act as a procedural scaffold grounded in image evidence.

Query (q): " Which class has more instances, tennis-court or baseball-field in the image? Return 'tie' if equal. **"**



Argument (a): ("tennis-court", "football-field")

Program (p):

```
def f(image, a):
    from tools import segment
    c1, c2 = a
    l1 = segment(image, c1)
    l2 = segment(image, c2)
    l1 = l1 if isinstance(l1, list) else []
    l2 = l2 if isinstance(l2, list) else []
    if len(l1) > len(l2): return c1
    if len(l2) > len(l1): return c2
    return "tie"
```

Output (o): "tie" (program execution)

<think>
The image shows several distinct objects: a football field (the large green field with markings) is visible in the top-right area of the image, a tennis court is visible in the top-left corner, a parking lot with many vehicles is present on the left side, a long building with a light-colored roof is in the middle-left area, a road runs horizontally across the lower portion of the image, and trees are scattered throughout the image.

The function f compares the lengths of the lists returned by segment(image, c1) and segment(image, c2) and returns the class name with more instances, or "tie" if the lengths are equal. We need to count the instances of football-field and tennis-court.

Let's count the instances of each class: segment(image, "football-field") appears to return 1 instance (the football field in the top-right area), and segment(image, "tennis-court") appears to return 1 instance (the tennis court in the top-left corner).

The function f will compare the two lengths: len(segment(image, "football-field")) = 1 and len(segment(image, "tennis-court")) = 1. Since both classes have the same number of instances, the function will return "tie".
</think>

<answer>("tennis-court", "football-field")**</answer>**

Answer (â): ("tennis-court", "football-field") (**Correct**)

Figure 10: **Qualitative analysis of abduction.** Given (I, p, o), the solver infers $\hat{a}$ by simulating p forward and searching for arguments whose execution reproduces the observed output o.

Query (q): “ In which quadrant (TL, TR, BL, BR) does the centroid of the largest ship lie in the image? **”**



Argument (a): “ship”

Program (p):

```
def f(image, a):
    import numpy as np
    from scipy.ndimage import center_of_mass
    from tools import segment
    instances = segment(image, a)
    if not isinstance(instances, list) or not instances:
        return "none"
    largest = max(instances, key=lambda m:
int(np.sum(m)))
    cy, cx = center_of_mass(largest)
    h, w = largest.shape
    return ("T" if cy < h / 2 else "B") + ("L" if cx < w
/ 2 else "R")
```

Output (o): “TR” (program execution)

<think>
1. The input class is "ship".
2. In the image, there are multiple ships docked at the harbor.
3. The task is to find the largest ship and determine the quadrant where its center of mass lies.
4. From the image, the long vessel docked at the top of the image appears to be the largest ship.
5. The center of mass calculation will be applied to this ship to determine the quadrant it falls into.

Let's calculate the center of mass for the largest ship.
The coordinates of the ship in the image are approximately:
- Top-left corner: (260, 60) / Bottom-right corner: (540, 130)

The center of mass calculation for a rectangle is at the midpoint of the top-left and bottom-right corners.
The center of mass for the ship is:
- X-coordinate: (260 + 540) / 2 = 400 / Y-coordinate: (60 + 130) / 2 = 95

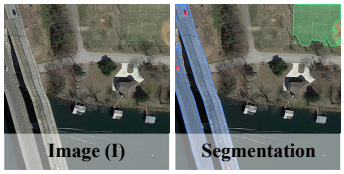
Since the center of mass (400, 95) is above the middle of the image (h/2 = 600/2 = 300) and to the right of the middle of the image (w/2 = 600/2 = 300), the quadrant is the top-right (TR). Therefore, the output is 'TR'.
</think>

<answer>TR**</answer>**

Answer ($\hat{o}$): “TR” (**Correct**)

Figure 11: **Qualitative analysis of deduction (reproduced from Figure 5).** Given (I, p, a), the solver follows the control flow of p and predicts $\hat{o}$, which exactly matches the program-executed label $o = p(a; I)$.

Query (q): “ How many instances of the given object class are in the image? **”**



Argument (a): “vehicle”, “bridge”, “football-field”

Program (p):

```
def f(image, a):
    from tools import segment
    instances = segment(image, a)
    if not isinstance(instances, list):
        return 0
    return len(instances)
```

Output (o): 3, 2, 1 (program execution)

<think>
The task is to deduce the function that computes the number of instances of a given class from the masks provided by the system. Let's analyze the given input/output pairs and the satellite image:

1. **Input 0**: `vehicle`
- **Output 0**: `3`
- From the image, we can see three vehicles on the road.

2. **Input 1**: `bridge`
- **Output 1**: `2`
- From the image, we can see two bridges spanning over the water.

3. **Input 2**: `football-field`
- **Output 2**: `1`
- From the image, we can see one football field in the top-right corner.

Based on these observations, the function needs to count the number of instances of the given class. The class names are `vehicle`, `bridge`, and `football-field`, and the outputs are the counts of instances for each class. **The function should return the length of the list of instance masks for the given class.**
</think>

<answer>
def f(image, a):
    from tools import segment
    return len(segment(image, a))
</answer>

Answer ($\hat{p}$): def f(image, a): return len(segment(image, a)) (**Correct**)

Figure 12: **Qualitative analysis of induction.** Given input-output pairs $\{(a_t, o_t)\}_{t \in \mathcal{V}}$, the solver synthesizes a program $\hat{p}$ consistent with the visible pairs.

G Primitives library and usage in self-play

This section supplements the analysis presented in Section 5 (Figure 6). We describe (i) the primitive library $\mathcal{F}$ exposed by the call interface, (ii) the procedure for detecting primitive invocations in programs, and (iii) the resulting per-primitive usage statistics.

G.1 Primitive library

The call interface $\mathcal{F}$ exposes 22 primitives organized into three groups. The first two groups follow the standard geometric-topological distinction in spatial information systems [8]; we extend it with a third group for operations over sets of masks. *Geometric primitives* extract metric attributes from a single mask, such as area, bounding box, centroid, or principal orientation. *Topological primitives* capture spatial relations between two masks (adjacency, containment, distance, overlap, proximity, relative direction), or between a mask and image partitions such as grids and quadrants. *Aggregation primitives* reduce or transform over a set of masks, producing outputs such as a count, an extremum, a filtered subset, a presence check, or a union.

Table 12 lists the primitives in each group.

Table 12: **Primitive library $\mathcal{F}$.** Within each group, primitives are listed alphabetically.

Group	Primitives
Geometric	`area`, `bbox`, `centroid`, `orientation`
Topological	`adjacent`, `contains`, `distance`, `grid`, `in_cell`, `nearest`, `overlaps`, `quadrant`, `relpos`
Aggregation	`argmin/argmax`, `components`, `count`, `exists`, `extreme`, `filter_by`, `largest/smallest`, `mean_position`, `union`

G.2 Deterministic detection rules

For each program p constructed by the proposer, we determine its set of primitives $\mathcal{A}(p) \subseteq \mathcal{F}$ via anchored regular-expression matching on the source code. A primitive is added to $\mathcal{A}(p)$ if any of its associated patterns matches a substring of p. Patterns are designed to identify the operations a program performs, regardless of surface syntax, covering both explicit calls to a primitive (e.g., `centroid(m)`) and the standard library idioms that realize the same operation inline (e.g., `ndi.center_of_mass(m)`).

Pattern construction. We derived these patterns in two stages. First, for each primitive, we enumerated the standard library functions and idioms that implement it (e.g., `np.any` for `exists`, `scipy.ndimage.center_of_mass` for `centroid`). Second, we inspected a sample of programs and added patterns for any recurring idioms not covered by the initial set. The full set of patterns and their match counts are reported alongside each primitive in Table 13.

Verification. We verified the matching procedure by manually annotating a held-out sample of 100 programs and comparing the predicted primitive set $\mathcal{A}(p)$ against expert annotations, observing every problem agreed on primitive membership.

G.3 Usage statistics

Across the 6,558 programs, Table 13 shows the program count, percentage, and top detection patterns for each primitive. Every group contains primitives appearing in over 1,000 programs, with `centroid` (55.4%), `exists` (48.7%), `quadrant` (32.9%), `extreme` (28.8%), `count` (25.7%), and `area` (25.0%) dominating the distribution.

Table 13: **Per-primitive usage across 6,558 programs.** A program is counted once per primitive. Pattern hits in parentheses give the number of programs matched by each pattern; a program may match multiple patterns of the same primitive. Within each group, primitives are ordered by frequency.

Group	Primitive	Count	%	Top detection patterns
Geometric	centroid	3,630	55.4	center_of_mass (3,558), cx, cy (2,331)
	area	1,642	25.0	np.sum (1,604), identifier area (41)
	bbox	282	4.3	np.where (255), xs/ys.min/max (136)
	orientation	3	0.0	regionprops (3)
Topological	quadrant	2,157	32.9	cy < H/2 (2,002), cx < W/2 (1,838)
	nearest	741	11.3	distance_transform_edt (713), min_dist, nearest (382)
	distance	272	4.1	Euclidean expansion (249), np.linalg.norm (2)
	grid	75	1.1	H/N, W/N for $N \geq 3$ (68)
	relpos	70	1.1	cardinal literals ("N", "NE", ...) (70)
	overlaps	52	0.8	np.any(&) (39), combined & (23)
	adjacent	3	0.0	adjacent (3)
	contains	1	0.0	contains (1)
	in_cell	0	0.0	—
Aggregation	exists	3,194	48.7	np.any (3,189), .any() (9)
	extreme	1,887	28.8	quadrant literals (1,050), direction literals (841)
	count	1,686	25.7	len(masks[...]) (1,168), len(instances) (518)
	argmin/argmax	1,129	17.2	argmax (1,079), argmin (70)
	union	985	15.0	|= (977), combined = (960)
	largest/smallest	253	3.9	max(..., key=) (220), sorted (29)
	mean_position	152	2.3	avg_y (115), avg_x (113)
	filter_by	11	0.2	list comprehension with if (11)
	components	1	0.0	from scipy.ndimage import label (1)

H Full results

This section reports the experimental results in Section 4. We provide (i) per-subtask breakdowns of the main VQA results in Table 1 (Section H.1), (ii) per-subtask breakdowns of the ablation study in Table 2 (Section H.2).

H.1 Full sub-task results on VQA benchmarks

Table 1 in the main paper presents the performance of GeoX and other baselines on RSVQA-HR and EarthVQA at the per-subtask level, and on GEOBench-VLM at the category level by averaging the subtasks within each category. Here, we provide the per-subtask breakdown for all benchmark datasets, including GEOBench-VLM across all 17 spatial-understanding tasks.

Table 14: **Full results on VQA benchmarks.** Performance of GeoX, instantiated from two base VLMs (LLaVA and Qwen), is compared against six baselines on RSVQA-HR, EarthVQA, and GEOBench-VLM. For each model, the number of human-curated QA pairs used for fine-tuning is listed; baseline models fine-tuned on fully curated data are grey shaded. The best result in each row is shown in **bold** and our results are highlighted in blue when they surpass the corresponding base (LLaVA or Qwen).

	Base		*Conventional*				*Self-play*	
Benchmark / Task	LLaVA	Qwen	GeoChat	VHM	EarthDial	RSThinker	**GeoX$_{L}$**	**GeoX$_{Q}$**
# Curated	0	0	318K	1.4M	11.1M	380K	0	0
RSVQA-HR								
Presence	57.9	60.1	57.9	**72.3**	60.0	55.4	61.9	63.2
Count	30.3	35.2	25.5	13.5	25.9	22.1	30.5	**36.6**
Area	2.7	10.3	**41.6**	13.5	38.8	22.1	40.3	16.5
Comparison	52.5	69.7	74.8	70.7	**78.9**	68.7	57.5	72.0
Average	35.9	43.8	50.0	42.5	**50.9**	42.1	47.6	47.1
EarthVQA								
Basic Judging	78.6	78.0	66.6	79.0	76.2	69.4	**82.2**	78.2
Reasoning-based Judging	55.8	**73.0**	37.0	71.8	41.4	42.6	53.8	68.6†
Reasoning-based Counting	24.8	23.4	16.6	12.8	5.6	15.4	**33.8**	26.6
Basic Counting	60.6	62.4	26.0	39.6	53.0	45.8	**70.2**	69.4
Object Situation Analysis	28.2	29.4	2.2	23.2	4.0	25.2	**30.4**	30.2
Comprehensive Analysis	19.6	33.6	7.8	24.6	11.0	19.0	19.5	**34.4**
Average	44.6	50.0	26.0	41.8	31.9	36.2	48.3	**51.2**
GEOBench-VLM								
Spatial Relation Classification	35.2	68.2	53.4	57.1	48.9	62.9	36.3	**70.0**
Building Counting	17.6	29.4	17.6	19.4	21.2	12.9	21.2	**37.1**
General Vehicle Counting	12.7	**38.7**	16.0	16.7	16.0	28.0	13.3	**38.7**
Specific Vehicle Counting	17.0	29.0	17.4	21.0	23.2	12.9	19.2	**33.9**
General Aircraft Counting	18.0	32.0	22.0	14.0	21.0	29.0	22.0	**33.0**
Specific Aircraft Counting	22.1	22.9	16.4	20.7	22.9	21.4	20.7	**27.9**
Marine Debris Counting	17.0	35.0	18.0	29.0	26.0	37.0	16.0	**38.0**
Trees Counting	15.3	**29.4**	21.2	24.7	21.2	5.9	20.0	28.2
Tree Health Assessment	17.8	**24.8**	19.1	19.1	16.6	8.9	15.9	19.1
Water Bodies Counting	32.9	63.5	21.2	48.2	30.6	28.2	37.7	**69.4**
Scene Classification	67.7	79.5	79.5	83.6	**94.0**	88.4	68.0	79.3
Land Use Classification	46.6	53.9	51.0	60.0	**62.0**	49.5	46.1	55.1
Crop Type Classification	25.5	**30.9**	18.2	21.8	25.5	10.9	27.3	27.3
Aircraft Type Classification	40.0	48.0	36.0	48.0	36.0	45.0	42.0	**53.0**
Ship Type Classification	30.4	39.6	32.9	24.6	35.7	27.5	30.9	**42.0**
Disaster Type Classification	11.8	**70.6**	58.8	64.7	17.6	23.5	11.8	64.7
Fire Risk Assessment	17.5	20.0	16.7	**29.2**	15.8	18.3	16.7	20.0
Average	26.2	42.1	30.3	35.4	31.4	30.0	27.4	**43.3**

H.2 Full sub-task results on the ablation study

Table 2 in the main paper summarizes the ablation across the four task categories of GEOBench-VLM. Here, we provide the per-subtask breakdown for each variant of GeoX. *Singleton* variants keep one solver mode active (A: Abduction, D: Deduction, I: Induction); *Drop-one* variants disable one of the three; *Training* variants (*BaseGen* and *SolvOnly*) retain all three modes but modify problem generation or proposer reward as described in Section 4.3; *Full* denotes the default configuration.

Table 15: **Full ablation study results on GEOBench-VLM.** Columns specify which solver modes are active: A (Abduction), D (Deduction), I (Induction). *Singleton Variants* keeps one mode; *Drop-one Variants* removes one; *Training Variants* use all three reasoning modes. *Full* denotes our default configuration. The best result in each row is shown in **bold** and the second-best is shown in underline.

	Singleton Variants			*Drop-one Variants*			*Training Variants*		*Full*
Config.	Abd	Ded	Ind	−Abd	−Ded	−Ind	BaseGen	SolvOnly	GeoX
Active Modes	A	D	I	D,I	A,I	D,A	D,A,I	D,A,I	D,A,I
Object Localization & Counting									
Spatial Relation Classification	67.8	67.8	67.8	67.8	67.8	68.5	66.5	68.2	**70.0**
Building Counting	33.5	35.3	32.9	32.9	35.3	35.9	36.5	32.9	**37.1**
General Vehicle Counting	34.7	36.7	36.0	34.7	33.3	36.7	38.0	28.7	**38.7**
Specific Vehicle Counting	27.7	28.6	27.7	26.3	30.8	30.4	31.7	27.7	**33.9**
General Aircraft Counting	29.0	**33.0**	28.0	28.0	30.0	31.0	30.0	29.0	**33.0**
Specific Aircraft Counting	22.9	25.0	**28.6**	22.1	25.7	24.3	21.4	22.1	27.9
Marine Debris Counting	33.0	35.0	34.0	**39.0**	33.0	35.0	32.0	36.0	38.0
Trees Counting	22.4	25.9	24.7	**29.4**	24.7	24.7	27.1	23.5	28.2
Tree Health Assessment	21.7	19.1	19.7	19.7	19.1	17.8	**24.8**	14.6	19.1
Water Bodies Counting	62.4	65.9	64.7	64.7	64.7	**70.6**	65.9	60.0	69.4
Average	35.5	37.2	36.4	36.5	36.4	37.5	37.4	34.3	**39.5**
Scene Understanding									
Scene Classification	78.1	78.6	76.6	75.4	76.9	78.1	78.6	75.4	**79.3**
Land Use Classification	54.4	54.9	53.4	52.5	53.7	54.7	**55.4**	53.2	55.1
Crop Type Classification	21.8	23.6	18.2	16.4	21.8	23.6	20.0	21.8	**27.3**
Average	51.4	52.4	49.4	48.1	50.8	52.1	51.3	50.1	**53.9**
Object Classification									
Ship Type Classification	40.6	40.6	41.5	38.2	43.5	38.6	**46.0**	39.6	42.0
Aircraft Type Classification	50.0	53.0	51.0	49.0	**55.0**	50.0	39.1	51.0	**53.0**
Average	45.3	46.8	46.3	43.6	**49.3**	44.3	42.6	45.3	47.5
Event Detection									
Disaster Type Classification	58.8	64.7	58.8	58.8	64.7	64.7	**70.6**	**70.6**	64.7
Fire Risk Assessment	19.2	20.0	21.7	21.7	22.5	20.8	**23.3**	22.5	20.0
Average	39.0	42.4	40.3	40.3	43.6	42.8	**47.0**	46.6	42.4
Total Average	39.9	41.6	40.3	39.8	41.3	41.5	41.6	39.8	**43.3**

I Prompt templates in self-play

This section provides the prompt templates used for the proposer and the three solver modes. All prompts share the same call interface introduced in Section 3.3: a library of spatial primitives implemented over `numpy`, `scipy.ndimage`, and `skimage.measure` and an open-vocabulary segmenter exposed as `from tools import segment`. The proposer prompt (Figure 13) constructs new triplets (p, a, o), while the three solver prompts (Figures 14–16) recover a masked element of the triplet under abduction, deduction, and induction, respectively.

Proposer Prompt | Generating a question q and an executable program p with argument a

TASK

Using the reference exemplars provided below, design a new and unique spatial reasoning problem for a remote sensing image. The submission must consist of three components, in the following order: a natural-language **question** q, an executable Python **program** p, and one **argument** a. The argument is plugged into the program together with the image to produce an output $o = p(a; I)$, completing the triplet (p, a, o) that defines the problem. The question q together with the image is then given to a test subject, who deduces o by reasoning about the visible content of the image (the program may optionally invoke the open-vocabulary segmenter to obtain class-aware regions).

CALL INTERFACE INSIDE `def f(image, a)`

`image` is a `PIL.Image.Image` (the remote sensing image, 600×600 RGB). The program may compose any callables from the call interface $\mathcal{E} = \mathcal{F} \cup \mathcal{T}$. All imports must be placed inside `def f`:

- *Tool registry* $\mathcal{T}$ — the open-vocabulary segmenter, imported as `from tools import segment`:
  - `mask = segment(image, a)` returns a binary segmentation mask (`np.ndarray` of shape (H, W), `bool`) for the target class on the given image.
  - The question `q` can be any natural-language phrase (e.g., `"building"`, `"vehicle"`, `"water surface"`, `"agricultural field"`).
  - Threshold and class confidence are handled internally; the returned mask is already boolean.
  - Call this only when the task genuinely requires class-aware regions.
- *Primitives library* $\mathcal{F}$ — operators implemented over `numpy`, `scipy.ndimage`, and `skimage.measure`.

QUESTION REQUIREMENTS (q)

Write a natural-language question q that describes what p computes. The question should be answerable by someone looking at the satellite image and may analyze scene or object classes individually or in comparison, at global or regional scale.

PROGRAM REQUIREMENTS (p)

- Entry program named `f` with signature `def f(image, a):`.
- Program must be *deterministic*, return a value, and complete within 10 seconds.
- Return type: `str`, `bool`, `int` (non-negative), `tuple[int, int, int, int]` (single bbox), or `list[tuple[int, int, int, int]]` (list of bboxes). For bboxes, use absolute pixel coordinates with `(x1, y1)` inclusive top-left and `(x2, y2)` exclusive bottom-right.
- Code must perform non-trivial spatial computation, ideally requiring multi-step reasoning over $\mathcal{E}$.

ARGUMENT REQUIREMENTS (a)

The varying input argument a may be a `str`, `int`, `tuple`, or bbox `(x1, y1, x2, y2)`. Submit one such argument as the test value, expressed as a valid Python expression (e.g., `'name'`, `1`, `('a', 'b')`, `(x1,y1,x2,y2)`).

OUTPUT FORMAT

You MUST provide exactly THREE separate blocks, in this order:

1. A ```` ```question ```` block for the natural-language question q.
2. A ```` ```python ```` block for the program p.
3. A ```` ```input ```` block for **one** argument a.

````
```question
<your natural-language question here>
```
```python
def f(image, a):
    <your code here>
    return <value>
```
```input
<your argument here>
```
````

REFERENCE EXEMPLES

$K = 6$ in-context examples are resampled per call from the corresponding bank. One is shown below; the remaining five are omitted for brevity.

question: `"What is the tight bounding box of the target class in the image?"`

```python
def f(image, a):
    from tools import segment
    import numpy as np
    mask = segment(image, a)
    if not mask.any():
        return (0, 0, 0, 0)
    ys, xs = np.where(mask)
    return (int(xs.min()), int(ys.min()),
            int(xs.max()) + 1, int(ys.max()) + 1)
```

argument `'building'` $\longmapsto$ **output** `(112, 87, 358, 294)`

Figure 13: **Proposer prompt**.

Solver Prompt | **Abduction** | Infer the argument $\hat{a}$ from the program p and its output o

TASK

You are given a remote sensing image, a Python **program** p, and the program's **output** o obtained by running p on the image with some unknown argument. Determine the **argument** $\hat{a}$ that produced the output. Examine the image and reason backwards through the program's logic to infer $\hat{a}$. When p invokes `segment`, mentally simulate the regions it would return for the candidate classes you consider.

CALL INTERFACE INSIDE `DEF F(IMAGE, A)`

Same as the proposer prompt; see Figure 13.

PROGRAM REQUIREMENTS (p)

Same as the proposer prompt; see Figure 13.

ARGUMENT REQUIREMENTS (a)

Same as the proposer prompt; see Figure 13.

OUTPUT FORMAT

Place your reasoning (2–3 sentences) in `<think>...</think>` tags, working backwards from o to infer $\hat{a}$. Then place $\hat{a}$ inside `<answer>...</answer>` tags as a valid Python expression matching the program's argument type, with no additional text afterward.

```
<think>
<your reasoning here>
</think>
<answer>
<your predicted argument here>
</answer>
```

EXAMPLE QUESTION

program p:

```python
def f(image, a):
    from tools import segment
    from skimage.measure import label
    return int(label(segment(image, a)).max())
```

output o: `3`

Infer the hidden argument $\hat{a}$ such that $p(\hat{a}; I) = o$.

Figure 14: **Solver prompt for abduction.** Given (I, p, o), the solver infers an argument $\hat{a}$ such that re-executing $p(\hat{a}; I)$ reproduces o (Eq. 4).

Solver Prompt | **Deduction** | Predict the output $\hat{o}$ from the program p and its argument a

TASK

You are given a remote sensing image, a Python **program** p, and an **argument** a to be supplied to p on the image. Determine the **output** $\hat{o}$ that $p(a; I)$ would produce. Examine the image and reason forward through the program's logic to predict $\hat{o}$. When p invokes `segment`, mentally simulate the regions it would return for the candidate classes you consider.

CALL INTERFACE INSIDE `DEF F(IMAGE, A)`

Same as the proposer prompt; see Figure 13.

PROGRAM REQUIREMENTS (p)

Same as the proposer prompt; see Figure 13.

ARGUMENT REQUIREMENTS (a)

Same as the proposer prompt; see Figure 13.

INPUTS PROVIDED TO THE SOLVER

- **Image** I: a 600×600 RGB remote sensing image.
- **Program** p: a Python program with signature `def f(image, a):`.
- **Argument** a: a Python value to be supplied to p.

OUTPUT FORMAT

Place your reasoning (2–3 sentences) in `<think>...</think>` tags, tracing the program forward from a to predict $\hat{o}$. Then place $\hat{o}$ inside `<answer>...</answer>` tags as a valid Python expression matching the program's return type, with no additional text afterward.

```
<think>
<your reasoning here>
</think>
<answer>
<your predicted output here>
</answer>
```

EXAMPLE QUESTION

program p:

```python
def f(image, a):
    from tools import segment
    from skimage.measure import label
    return int(label(segment(image, a)).max())
```

argument a: `'vehicle'`

Predict the output $\hat{o} = p(a; I)$.

Figure 15: **Solver prompt for deduction.** Given (I, p, a), the solver predicts an output $\hat{o}$ that matches $o = p(a; I)$ (Eq. 5).

Solver Prompt | **Induction** | Synthesize the program $\hat{p}$ from observed input-output pairs

TASK

You are given a remote sensing image, a set of **input-output** pairs $\{(a_t, o_t)\}_{t\in\mathcal{V}}$ produced by some hidden program p such that $o_t = p(a_t; I)$. Synthesize a Python **program** $\hat{p}$ that reproduces every visible pairs. Examine the image and reason inductively from the input–output pattern to recover $\hat{p}$. When p invokes `segment`, mentally simulate the regions it would return for the candidate classes you consider.

CALL INTERFACE INSIDE `DEF F(IMAGE, A)`

Same as the proposer prompt; see Figure 13.

PROGRAM REQUIREMENTS (p)

Same as the proposer prompt; see Figure 13.

ARGUMENT REQUIREMENTS (a)

Same as the proposer prompt; see Figure 13.

INPUTS PROVIDED TO THE SOLVER

- **Image** I: a 600×600 RGB remote sensing image.
- **Visible pairs** $\{(a_t, o_t)\}_{t\in\mathcal{V}}$: input-output examples produced by the hidden p on this image.

OUTPUT FORMAT

Place your reasoning (2–3 sentences) in `<think>...</think>` tags, generalizing from $\{(a_t, o_t)\}_{t\in\mathcal{V}}$ to recover $\hat{p}$. Then place $\hat{p}$ inside `<answer>...</answer>` tags as a single Python code block, with no additional text afterward.

````
<think>
<your reasoning here>
</think>
<answer>
```python
def f(image, a):
    <your code here>
    return <value>
```
</answer>
````

EXAMPLE QUESTION

visible pairs $\{(a_t, o_t)\}_{t\in\mathcal{V}}$:

```
a_1 = 'vehicle'        ->  o_1 = 3
a_2 = 'building'       ->  o_2 = 12
a_3 = 'water'          ->  o_3 = 1
```

Synthesize $\hat{p}$ such that $\hat{p}(a_t; I) = o_t$ for all $t \in \mathcal{V}$.

Figure 16: **Solver prompt for induction.** Given $(I, q, \{(a_t, o_t)\}_{t\in\mathcal{V}})$, the solver synthesizes a program $\hat{p}$ such that $\hat{p}(a_t; I) = o_t$ for all $t \in \mathcal{V}$, with consistency on a held-out subset $\mathcal{U}$ used for verification (Eq. 6).

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction (Section 1) clearly state the contributions and scope of the paper, and these are supported by experimental results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations of the proposed approach are discussed in Section 6

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The paper does not contain formal theorems or lemmas requiring proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The experimental setup, including baselines, datasets, model configurations, and evaluation protocol are described in Section 4.1 and Appendix C providing sufficient information to reproduce the main results.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The datasets used for training and evaluation are publicly available and documented in Section 4.1 and Appendix A. Detailed implementation settings and prompt templates are provided in the Appendix C and Appendix I, respectively.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: Training and evaluation details, including data splits, hyperparameters, and optimizer settings, are reported in Section 4.1 and Appendix C.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Our evaluation protocol uses majority voting over multiple sampled outputs to determine a single final answer per problem before grading, which yields a single deterministic score per configuration and therefore does not admit standard deviations or confidence intervals over the reported metric. The number of samples used for voting and its effect are discussed in Section 4.1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Compute resources, including hardware specifications and approximate runtime, are reported in Appendix C.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conducted in this paper conforms with the NeurIPS Code of Ethics in every respect.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Potential societal impacts are discussed in the Concluding Remarks in Section 6.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release any pre-trained models, image generators, or scraped datasets that pose a high risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All existing assets (datasets, models, and code packages) used in this paper are properly cited, and their licenses and terms of use are respected. Details are provided in Section 4.1, Appendix A and Appendix B.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We plan to release (i) the source code of our method, and (ii) a dataset of problems generated during self-play, derived from publicly available remote sensing imagery datasets. Both will be released with documentation describing usage, the generation procedure, and intended use, and the licenses of the underlying source datasets will be respected. An anonymized version of the code is included in the supplemental material at submission time, and details on the released assets are provided in Section 1, Section 4.1, Appendix C, Appendix E, and Appendix I.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLMs are not used as an important, original, or non-standard component of the core methodology of this research.

Guidelines:

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