TerraScope: Pixel-Grounded Visual Reasoning for Earth Observation

Supplementary Material

Appendix Overview

A. Limitations and Future Work

TerraScope focuses on pixel-level grounding for earth observation data, but it has several limitations. First, like other multimodal large language models, TerraScope may produce hallucinated outputs, *e.g.*, generating plausible but factually incorrect reasoning traces or inaccurate mask predictions that do not correspond to actual ground features [2, 40]. Mitigating such hallucinations through improved training strategies, verification mechanisms, or retrieval-augmented generation is an important direction for future work. Second, the interleaved generation of masks and reasoning traces increases context length during training and inference. We analyze its computational complexity in Sec. G. A potential solution is to compress masked visual tokens to reduce context length while retaining visual grounding capability. Third, although TerraScope supports multi-sensor reasoning, it currently handles only optical (RGB) and SAR data, with limited capability for multi-spectral and hyper-spectral imagery [27]. Future work will explore integrating these challenging data sources into the reasoning framework. Finally, the current temporal reasoning capability of TerraScope is limited to bi-temporal analysis (*i.e.*, comparing two time points). Many real-world EO applications, such as urban expansion monitoring, deforestation tracking, and climate trend analysis, require reasoning over long temporal sequences [41]. Extending TerraScope to support multi-temporal and continuous time-series reasoning is an important direction for future work.

B. Comparison to Concurrent Works

TerraScope belongs to the category of "thinking with images" models. In Sec. 1 and Sec. 2, we described the distinction between our approach and agent-based models. In this section, we provide detailed comparisons with both unified interleaved reasoning models and LLM-agent-based methods.

Comparison with Unified Interleaved Reasoning Models. Several concurrent works share similar interleaved reasoning mechanisms with TerraScope, including ICoT [9], GRIT [7], VGR [47], and Mint-CoT [3]. However, they differ from TerraScope in two key aspects. First, these models are designed for general vision tasks and have limited transferability to earth observation, as they lack multi-modal reasoning (optical/SAR) and multi-temporal reasoning capabilities essential for EO applications. Second, they employ different mechanisms for interleaved reasoning:

- **ICoT** [9] proposes a training-free module that leverages text-image cross-attention maps in LLMs to select relevant tokens. However, this approach is limited to scenarios with salient objects and fails when queries are complex or involve high-level semantic reasoning not directly tied to visible objects.
- **GRIT and VGR** [7, 47] use language to model object coordinates (bounding boxes), which is inadequate for representing pixel-level regions in EO data where spatial phenomena often lack clear boundaries.
- **Mint-CoT** [3] overcomes bounding-box limitations by selecting relevant image tokens through similarity-based implicit selection. However, this approach may include tokens irrelevant to the current reasoning step. To validate this, we trained Mint-CoT on our Terra-CoT dataset following their official training paradigm, converting our pixel-level masks into their token indices. Experiments (Tab. A) show Mint-CoT underperforms TerraScope on TerraScope-Bench, confirming the importance of explicit mask generation for pixel-grounded reasoning.

Comparison with LLM-Agent-Based Methods. We further compare TerraScope with concurrent agentic approaches, including ThinkGeo and EarthAgent. As shown in Tab. A, these methods significantly underperform TerraScope. We attribute this to two main limitations: (1) *Hallucination*: the LLM orchestrator may misinterpret tool outputs or introduce reasoning errors during multi-step planning [21, 24]; (2) *Weak perception*: ThinkGeo relies on box-level grounding, while EarthAgent adopts SAM-based grounding with independently trained modules, limiting cross-module synergy. In contrast, TerraScope's unified training paradigm enables bidirectional enhancement between reasoning and pixel-level grounding, which agentic pipelines with decoupled components cannot achieve.

Methods	TerraBench.	Landsat.
Interleaved Reasoning Models		
Mint-CoT (with SFT)	54.6	62.8
Mint-CoT (with SFT + RL)	55.7	63.2
LLM-Agent-Based Methods		
ThinkGeo	28.5	–
EarthAgent	37.6	–
TerraScope	**68.9**	**73.9**

Table A. Comparison of TerraScope with interleaved reasoning models and LLM-agent-based methods on TerraScope-Bench.

C. Details of TerraScope

Vision-Language Model. The VLM component of TerraScope is built upon InternVL-3 [59]. In InternVL-3, each image is divided into multiple patches at a pre-defined resolution (448×448). Each patch is processed by the vision encoder and encoded into 256 tokens. For instance, an image with 4 patches (plus one global thumbnail) yields $(4+1) \times 256 = 1{,}280$ visual tokens in total. For multi-temporal inputs, we do not split images into patches but directly feed independent images into the model. For example, for a multi-temporal sequence with T observations, the total number of visual tokens is $T \times 256$.

Pixel-Grounding Module. TerraScope's pixel-grounding module is initialized with the pre-trained SAM-2 model [36]. We connect SAM-2 and the LLM via the special token `[SEG]`. The hidden states of the `[SEG]` token from the last layer of LLM serve as a spatial prompt and are fed into SAM-2's decoder, which generates segmentation masks. This design allows the LLM to control mask generation through learned prompt embeddings.

During training, the SAM-2 decoder is fine-tuned to understand the spatial prompts, and gradients are backpropagated through the `[SEG]` token to the LLM, enabling it to generate better prompts. During inference, if the LLM does not generate a `[SEG]` token, we interpret this as indicating that no segmentation is needed for the current reasoning step.

Masked Token Selection. To balance effectiveness and efficiency, we set a maximum threshold $\lambda = 128$ for the number of visual tokens in $\mathbf{v}_i$. If the number of selected tokens exceeds this threshold, we apply spatial uniform sampling to retain λ tokens while preserving spatial coverage. Specifically, we divide the masked region into a $\lceil\sqrt{\lambda}\rceil \times \lceil\sqrt{\lambda}\rceil$ grid and select one token from each grid cell, choosing the token closest to the cell center. This ensures representative spatial sampling across the entire masked region rather than biased concentration in any local area.

Inference Process. TerraScope performs autoregressive generation with pixel-grounded reasoning (Algorithm 1).

Algorithm 1 TerraScope Inference

Input: Question embeddings $\mathbf{q}$, Visual features $\mathbf{v}$ (or $\mathbf{v}^{\text{opt}}$, $\mathbf{v}^{\text{SAR}}$), Mask decoder f_{mask}, Max tokens λ, Stopping criteria SC
Output: Generated answer $\mathbf{a}$ with reasoning traces
predicted_tokens $\leftarrow$ [] ▷ Initialize as empty list
reasoning_step $\leftarrow 0$ ▷ Track reasoning step index
inputs $\leftarrow$ Initialize($\mathbf{q}$, $\mathbf{v}$) ▷ Initialize inputs for prefilling
▷ Compute modality relevance scores if multi-modal
if both $\mathbf{v}^{\text{opt}}$ and $\mathbf{v}^{\text{SAR}}$ are available **then**
  $\beta_j^{\mu} \leftarrow \frac{1}{L}\sum_{\ell=1}^{L} \text{Softmax}\left(\frac{\mathbf{v}^{\mu}\mathbf{q}^{\top}}{\sqrt{D}}\right)_{j\ell}$ for $\mu \in \{\text{opt}, \text{SAR}\}$
end if
while SC not met **do**
  next_token, hidden_state $\leftarrow$ **LLM**(inputs)
  Append next_token to predicted_tokens
  ▷ Check if `[SEG]` token is generated
  if next_token = `[SEG]` **then**
    reasoning_step $\leftarrow$ reasoning_step $+ 1$
    $i \leftarrow$ reasoning_step
    ▷ Generate segmentation mask
    $\mathbf{m}_i \leftarrow f_{\text{mask}}(\text{hidden\_state})$
    ▷ Resize mask to token grid
    $\mathbf{m}_i^{\text{tok}} \leftarrow \text{ResizeToTokenGrid}(\mathbf{m}_i)$
    ▷ Select tokens with $> 50\%$ coverage
    $\mathcal{I} \leftarrow \{j \mid \text{Coverage}(\mathbf{m}_i^{\text{tok}}, j) > 0.5\}$
    ▷ Apply spatial sampling if exceeds threshold
    if $|\mathcal{I}| > \lambda$ **then**
      $\mathcal{I} \leftarrow \text{SpatialUniformSample}(\mathcal{I}, \mathbf{m}_i^{\text{tok}}, \lambda)$
    end if
    ▷ Extract masked visual features
    if both modalities available **then**
      for $j \in \mathcal{I}$ **do**
        if $\beta_j^{\text{opt}} > \beta_j^{\text{SAR}}$ **then**
          $\mathbf{v}_j \leftarrow \mathbf{v}_j^{\text{opt}}$
        else
          $\mathbf{v}_j \leftarrow \mathbf{v}_j^{\text{SAR}}$
        end if
      end for
    else
      Extract features from single modality
    end if
    $\mathbf{v}_i \leftarrow \{\mathbf{v}_j \mid j \in \mathcal{I}\}$
    Append $\mathbf{v}_i$ to predicted_tokens
  end if
  inputs $\leftarrow$ Update(inputs, predicted_tokens) ▷ Update KV cache for next generation
end while
$\mathbf{a} \leftarrow$ Tokenizer.decode(predicted_tokens)
return $\mathbf{a}$

The vision encoder processes input images to obtain visual features $\mathbf{v}$ (or $\mathbf{v}^{\text{opt}}$, $\mathbf{v}^{\text{SAR}}$ for multi-modal inputs), which are cached for efficiency. At each step, the LLM generates the next token. When a `[SEG]` token is generated, TerraScope: (1) generates a segmentation mask $\mathbf{m}_i$ via the mask

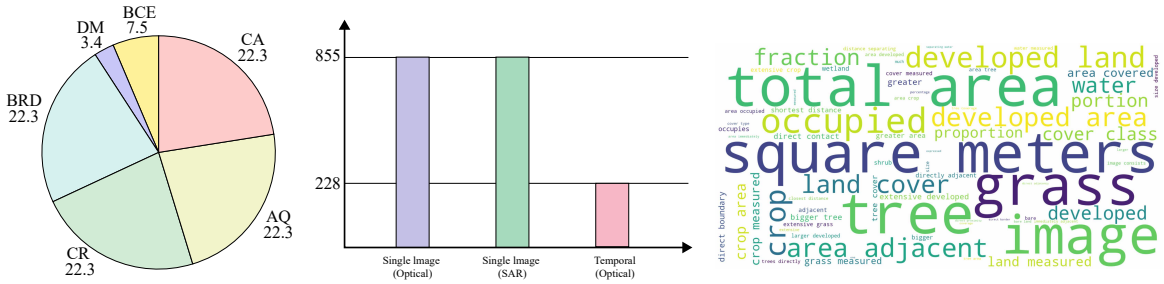


Figure A. Data distribution of TerraScope-Bench.

decoder conditioned on the [SEG] token's hidden states; (2) extracts masked visual features $\mathbf{v}_i$ by selecting tokens with $> 50\%$ coverage and applying spatial uniform sampling if the count exceeds $\lambda = 128$; (3) for multi-modal inputs, adaptively selects between optical and SAR based on text-relevance scores β_j^μ. The selected features $\mathbf{v}_i$ are then injected into the generation sequence, and the LLM continues reasoning conditioned on both textual and visual contexts through KV cache updates.

D. Details of TerraScope-Bench

D.1. Overview

We present a more detailed analysis of TerraScope-Bench in Fig. A. Subfigures (a–c) illustrate the distribution of task categories, image source (multi-sensor and multi-temporal) and the visualization of word clouds of question, showing that TerraScope-Bench covers a wide variety of object types and semantics, enabling comprehensive evaluation across pixel-level grounded visual reasoning tasks.

D.2. Data Annotations for TerraScope-Bench

TerraScope-Bench consists of six task types requiring pixel-grounded reasoning. We construct the benchmark through a three-stage pipeline: (1) heuristic-based answer generation from pixel-level annotations, (2) GPT-4o-based question rephrasing and distractor generation, and (3) expert validation and quality control.

Stage 1: Heuristic-Based Answer Generation. We leverage existing pixel-level segmentation annotations to generate ground-truth answers using deterministic rules. The benchmark includes three data sources: ChatEarthNet and BigEarthNet for land cover analysis and xBD for building damage assessment. For each image, we process the segmentation mask to extract spatial information required for different task types. The specific rules for each task are:

- **Absolute Area Calculation**: For a given land cover class c, we count all pixels with label c in the segmentation mask. The area is computed as $A_c = N_c \times r^2$, where N_c is the pixel count and r is the spatial resolution (10m for Sentinel-2). Questions specify a single target class (e.g., "What is the area of forest?"), and the ground-truth answer is the computed area in square meters or hectares. We only include classes with $A_c > 0$ to avoid trivial questions.
- **Coverage Percentage**: For a target land cover class c, we compute the percentage as $P_c = \frac{N_c}{N_{\text{total}}} \times 100\%$, where N_c is the pixel count of class c and N_{total} is the total number of valid pixels in the image (excluding background/void). Questions ask for the coverage of a specific class (e.g., "What percentage of the image is cropland?"). We require $P_c \geq 5\%$ to ensure the class is visually significant and avoid questions about negligible regions.
- **Comparative Area Ranking**: Given a set of land cover classes present in the image, we rank them by area in descending order: $c_1, c_2, \ldots, c_n$ where $A_{c_1} \geq A_{c_2} \geq \cdots \geq A_{c_n}$. Questions ask for the largest class (e.g., "Which land cover type has the largest area?") or relative ranking (e.g., "Is forest larger than grassland?"). We only include classes with $P_c \geq 5\%$ in the ranking to ensure clear visual distinction. For binary questions, we require $|A_{c_i} - A_{c_j}| > 0.1 \times \max(A_{c_i}, A_{c_j})$ to avoid ambiguous comparisons between similar-sized regions.
- **Distance Measurement**: To measure the minimum distance between two land cover classes c_i and c_j, we: (1) generate binary masks M_i and M_j for each class; (2) apply Euclidean distance transform (distance_transform_edt) to M_i to compute the distance from each pixel to the nearest c_i pixel; (3) extract the minimum value within M_j, which gives the minimum distance $d(c_i, c_j)$ in pixels; (4) convert to meters using spatial resolution ($d_{\text{meters}} = d \times r$). Questions specify two distinct classes (e.g., "What is the distance between forest and water?"). We require both classes to form spatially connected components (removing isolated pixels via morphological opening) and enforce $d(c_i, c_j) > 10$ pixels to avoid trivial adjacent cases. For classes with multiple disconnected regions, we report the minimum distance across all region pairs.
- **Boundary Relationship Detection**: To determine if two land cover classes c_i and c_j are adjacent, we: (1) generate binary masks M_i and M_j; (2) apply morphological dilation (binary_dilation) with a 3×3 structuring element to M_i, creating M_i^{dilated}; (3) check if $M_i^{\text{dilated}} \cap M_j \neq$

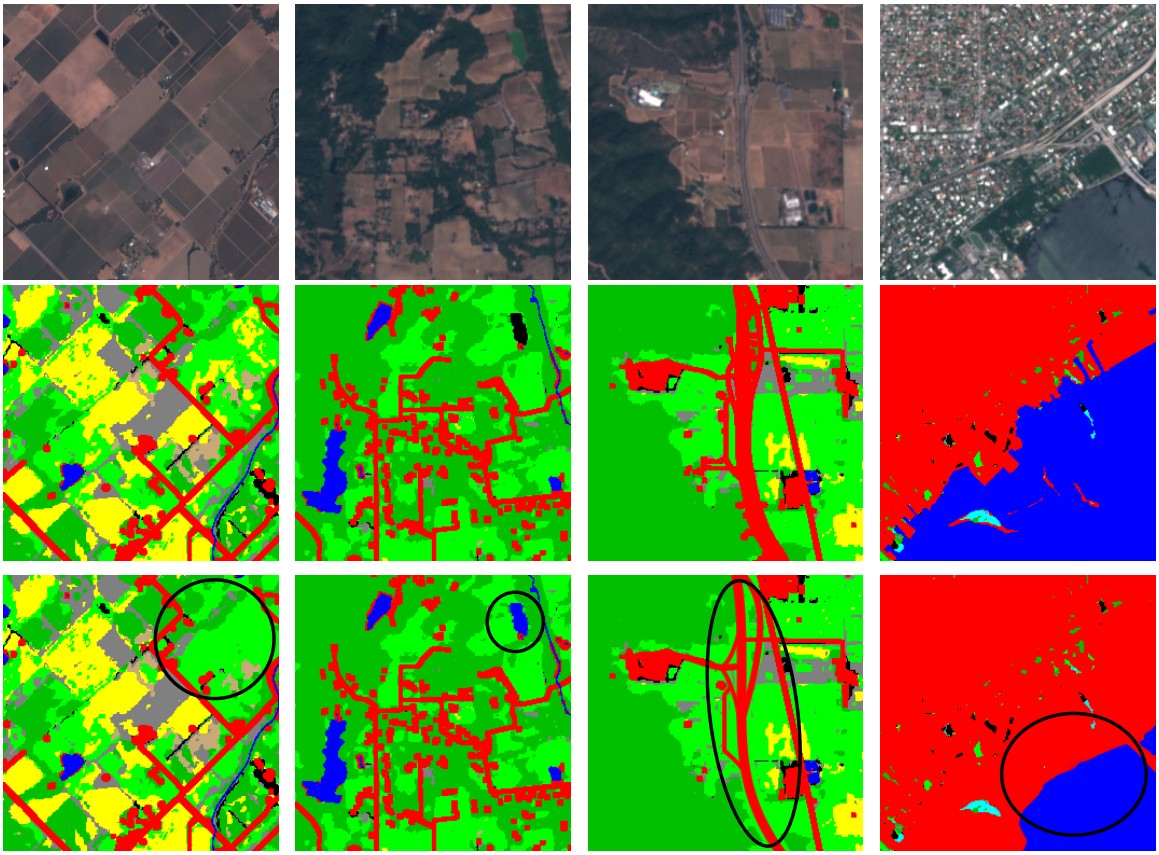


Figure B. Examples of human expert verification of mask accuracy. The first row shows original images, the second row shows original masks, and the third row shows masks modified by human annotators.

$\emptyset$. If the intersection is non-empty, the classes are considered adjacent (sharing a boundary). Questions ask binary yes/no queries (e.g., "Does forest border water?"). We filter out class pairs where either region is too small ($P_c < 3\%$) or fragmented (more than 5 disconnected components) to ensure clear, unambiguous boundaries. For multi-component classes, adjacency is determined if any component pair satisfies the criterion.

- **Building Change Estimation**: Using the xBD dataset, we compare pre-disaster and post-disaster satellite imagery to identify destroyed buildings. The annotation process: (1) parse building footprint polygons from JSON files in WKT format using Shapely (`wkt.loads`); (2) filter polygons based on damage classification labels (only retain buildings labeled as "destroyed"); (3) rasterize polygon geometries to binary masks using OpenCV (`cv2.fillPoly`) at the image resolution; (4) count destroyed buildings $N_{\text{destroyed}}$ and total buildings N_{total} to compute damage rate $R = \frac{N_{\text{destroyed}}}{N_{\text{total}}} \times 100\%$. Questions ask about building counts (e.g., "How many buildings were destroyed?") or damage percentages (e.g., "What percentage of buildings were destroyed?"). We only include samples with $N_{\text{total}} \geq 10$ buildings and $N_{\text{destroyed}} \geq 3$ to ensure statistically meaningful damage assessment. Polygon parsing handles potential coordinate precision issues and self-intersecting geometries using Shapely's built-in validation.

The implementation uses Python libraries including NumPy for array operations, SciPy for distance transforms (`distance_transform_edt`, `binary_dilation`), Shapely for geometry processing (`wkt.loads`, `Polygon`), and OpenCV for mask rendering.

Stage 2: GPT-4o-Based Question Refinement. To ensure linguistic diversity and difficulty, we use GPT-4o to: (1) rephrase template questions into natural language variations, and (2) generate plausible distractors for multiple-choice format. For comparative area ranking and boundary relationship detection, we generate 2 options (binary choice). For other tasks (absolute area, coverage percentage, distance measurement, building change estimation), we generate 4 options. The rephrasing prompt is designed to maintain semantic equivalence while varying question structure and wording.

GPT-4o Rephrasing Prompt:

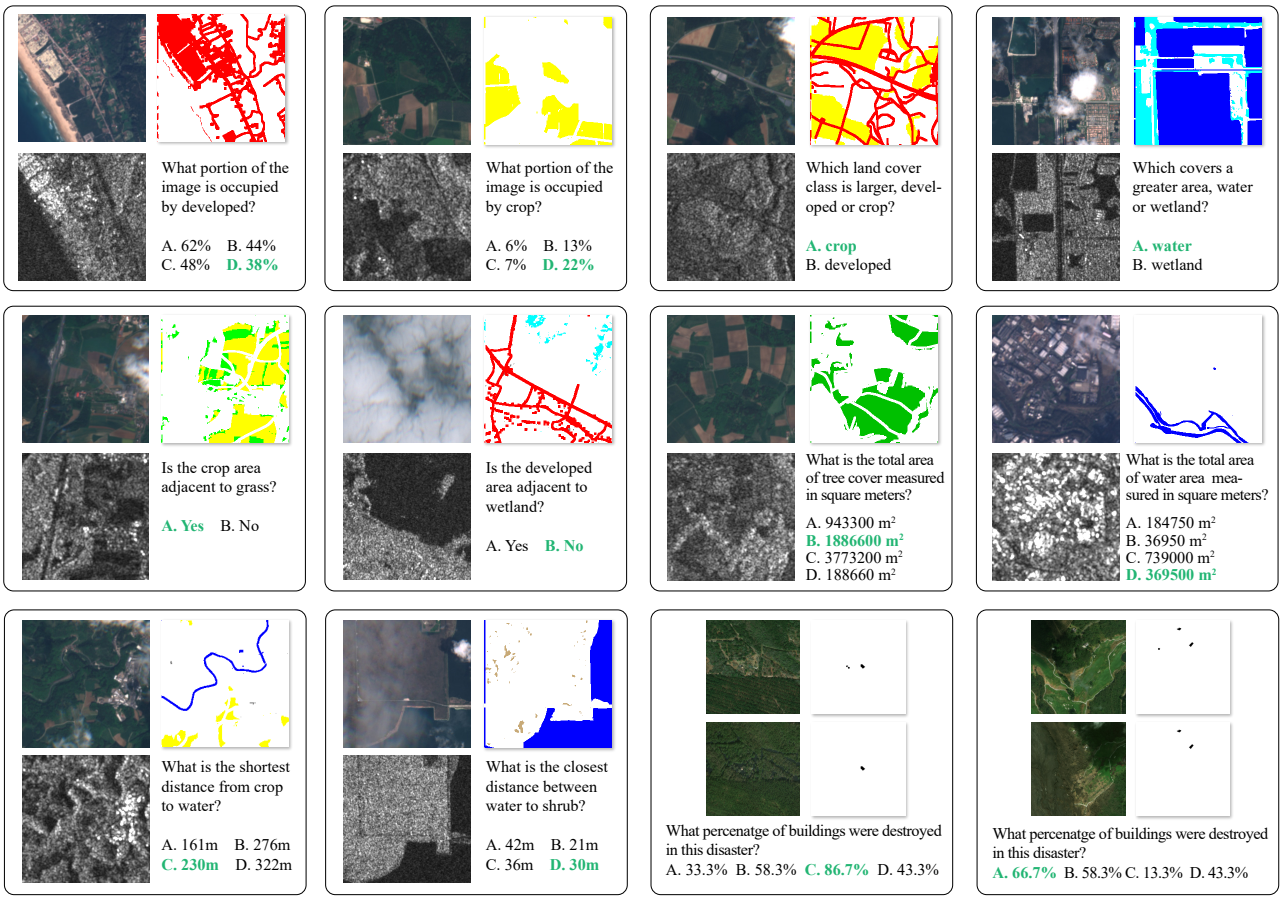


Figure C. More examples of TerraScope-Bench, including the questions, answers and the masks involved in the CoT.

Question Rephrasing Prompt

```
Given the following question template
and answer:
Question: {original_question}
Answer: {ground_truth_answer}

Task: Rephrase the question to make
it more natural and diverse while
preserving the original meaning.
Generate {num_options} plausible but
incorrect answer choices (distractors)
that are numerically/semantically
close to the ground truth but clearly
distinguishable. Ensure distractors
are realistic and challenging.

Output format:
{
"question": "rephrased question",
"options": ["option A", "option B",
"option C", "option D"],
"answer": "correct option letter"
}
```

Stage 3: Expert Validation. We recruit 4 domain experts in geoscience and disaster assessment to ensure annotation quality. Each expert is assigned to validate one or two specific task types. The validation process includes:

1. **Mask accuracy check**: Verify that segmentation masks correctly represent land cover boundaries or building footprints, as shown in Fig. B.
2. **Answer correctness**: Validate that ground-truth answers match the mask through manual calculation.
3. **Distractor quality**: Ensure distractors are plausible but clearly incorrect.
4. **Question clarity**: Check that questions are unambiguous and answerable from the image.

After initial annotation, experts cross-validate each other's work and score sample quality on a 3-point scale (low/medium/high). Only samples with consensus (all experts agree on high quality) are retained. Samples with erroneous masks, ambiguous questions, or invalid distractors are filtered out. The final benchmark contains 3,837 expert-verified samples across six task types. Sample visualizations are shown in Fig. C.

E. Details of Training Data

E.1. Pretraining Data

For Stage 1 grounded pretraining, we synthesize 2M referring expression segmentation (RES) samples from two sources: 1.5M from BigEarthNet and 0.5M from ChatEarthNet. Both datasets provide semantic segmentation annotations with pixel-level class labels. To convert them into RES format, we randomly select one land cover category from each image and construct the instruction as "Please segment the [class name]", where [class name] is replaced with the specific land cover type (e.g., "forest", "cropland", "water"). The corresponding ground-truth masks are extracted from the original semantic labels and encoded in Run-Length Encoding (RLE) format for efficient storage. This synthetic RES data enables the mask decoder to learn foundational pixel-level grounding capabilities before instruction tuning.

E.2. Terra-CoT Dataset Construction

Cap-CoT Curation. We construct the Cap-CoT (Caption with Chain-of-Thought) dataset from four sources: ChatEarthNet, BigEarthNet, xBD, and TEOChat (region-based change question answering). We employ an RoI-based summarization strategy where class information or original metadata, along with mask-overlaid images, are fed into Qwen3-VL-235B to generate captions with reasoning chains. The generation prompt instructs the model to produce chain-of-thought reasoning that explicitly refers to the provided segmentation semantic labels. This ensures that generated captions are grounded in precise spatial information rather than vague descriptions.

Caption Generation Prompt for Cap-CoT

System: You are an expert in remote sensing image analysis. Your task is to generate a detailed caption with step-by-step reasoning for the given satellite image.

Input:
- Satellite image with mask overlay
- Segmentation labels: {label_1, label_2, ..., label_n}
- Metadata: [resolution, sensor type, location]

Instructions:
1. Analyze the spatial distribution of each land cover type shown in the segmentation masks
2. Generate a chain-of-thought reasoning process that:
   - Explicitly mentions each segmented region
   - Describes spatial relationships between different land cover types
   - Estimates approximate coverage or area for major land cover classes
   - Notes any significant patterns or features
3. Provide a final comprehensive caption summarizing the image

Output Format:

```
<think>
First, I observe [description
of dominant land cover]. The
segmentation shows [specific
area/pattern]. [SEG for
region 1] covers approximately
[percentage/area]. Next, I
notice [another land cover type].
[SEG for region 2] appears in
[location/pattern]. The spatial
relationship between these regions
shows [description]. Additionally,
[other observations]...
</think>
<caption>
This satellite image shows
[comprehensive summary including
all major land cover types, their
spatial distribution, and key
characteristics].
</caption>
```

VQA-CoT Curation. Based on the 250K Cap-CoT dataset, we first train TerraScope-Cap, a caption-specialized variant of TerraScope. We then use TerraScope-Cap to annotate images from ChatEarthNet, BigEarthNet, RSVQA-LR, and xBD training sets, generating captions and predicted masks. For ground-truth mask refinement, we compute the intersection between predicted masks and available ground-truth annotations when available, ensuring higher quality.

Using these captions as context, we synthesize L1-level VQA samples covering six task types. We design predefined templates for each task type to ensure consistency and coverage:

L1-Level VQA Templates

Task 1: Object Existence
Template: "Is there any [class] in the image?"
Example: "Is there any forest in the image?"
Answer: "Yes" or "No"

Task 2: Object Counting
Template: "How many [object] are there in the image?"
Example: "How many buildings are there in the image?"
Answer: "[number] [object]" (e.g., "15 buildings")

Task 3: Localization
Template: "Where is the [class] located in the image?"
Example: "Where is the water body located in the image?"
Answer: "[cardinal direction/relative position]" (e.g., "in the northeastern part", "along the southern edge")

Task 4: Area Quantification
Template 1: "What is the area of [class]?"

Template 2: "What percentage of the image is covered by [class]?"
Example 1: "What is the area of cropland?"
Example 2: "What percentage of the image is covered by forest?"
Answer 1: "[number] square meters" or "[number] hectares"
Answer 2: "[percentage]%"

Task 5: Boundary Detection
Template: "Does [class1] border [class2]?"
Example: "Does forest border water?"
Answer: "Yes" or "No"

Task 6: Distance Measurement
Template: "What is the distance between [class1] and [class2]?"
Example: "What is the distance between cropland and water?"
Answer: "[number] meters"

Generation Strategy:
- For each image, randomly select 2-4 task types
- Ensure at least one task per image requires pixel-level reasoning
- Classes are sampled from available segmentation labels
- Answers are computed deterministically from ground-truth or refined masks

Building upon L1-level VQA, we use GPT-4o to synthesize more complex reasoning problems that require multi-step spatial analysis. The synthesis prompt encourages GPT-4o to create questions involving comparative reasoning, spatial relationships, and compositional understanding.

Fig. D visualizes the composition and distribution of the Terra-CoT dataset from three perspectives. First, we show the geographic distribution of source images, demonstrating global coverage across diverse geographical regions and climatic zones. Second, we present the data source breakdown for Cap-CoT and VQA-CoT subsets, illustrating how different source datasets contribute to caption generation and question-answering components. Third, we provide sample quantity statistics across the three dataset tiers: Cap-CoT (caption with chain-of-thought), L1-level VQA (simple spatial queries), and L2-level VQA (complex multi-step reasoning).

L2-Level VQA Synthesis Prompt

System: You are an expert in designing complex spatial reasoning questions for satellite imagery analysis.

Input:
- Image caption with spatial information
- L1-level QA pairs (simple questions and answers)
- Available land cover classes: {class_1, class_2, ..., class_n}

Task: Generate 2-3 complex reasoning questions from two categories:

Category 1: Spatial Reasoning Questions
These questions focus on geometric and spatial properties requiring pixel-level analysis, such as area comparison, distance measurement, boundary relationships, coverage quantification, and spatial distribution patterns.

Category 2: Semantic Reasoning Questions
These questions focus on understanding land cover semantics, ecological patterns, temporal changes, functional relationships, and overall landscape composition.

Requirements:
1. Generate at least one question from each category
2. Questions must require multi-step reasoning
3. Answers should be deterministic and verifiable
4. Spatial reasoning questions must involve precise geometric analysis
5. Semantic reasoning questions must demonstrate understanding of land cover semantics

Output Format:
For each question, provide:
- Question text
- Category: [Spatial Reasoning] or [Semantic Reasoning]
- Ground-truth answer
- Reasoning steps required (brief description)
- Classes involved

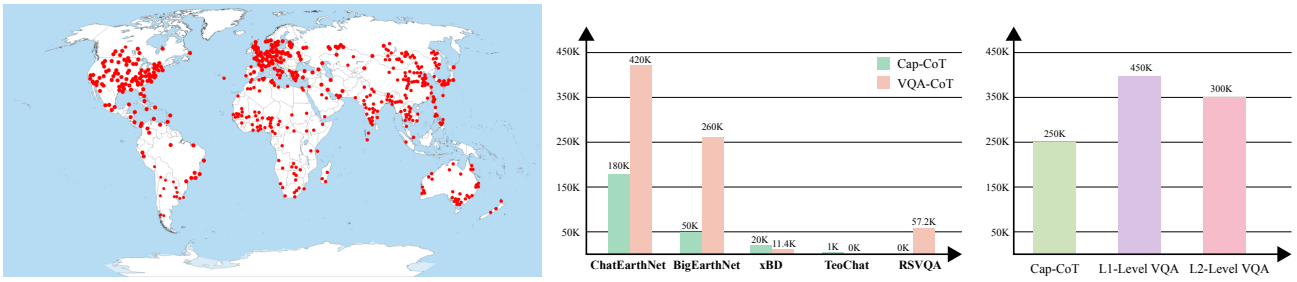
Hyperparameter	Value
Overall batch size	32
Learning rate	4e-5
LR Scheduler	Cosine decay
DeepSpeed ZeRO Stage	ZeRO-2
Optimizer	Adam
Warmup ratio	0.3
Epoch	1
Weight decay	0
Precision	bf16

Table B. Hyperparameters of TerraScope.

F. Experimental Settings

F.1. Training Details of TerraScope

We provide complete training hyperparameters for TerraScope in Tab. B. The model is fine-tuned for one epoch on Terra-CoT with a total batch size of 32, using the Adam optimizer with cosine learning rate decay. We employ DeepSpeed ZeRO-2 for memory-efficient training and use bf16 mixed precision to accelerate computation.



(a) (b) (c)

Figure D. Data distributions of Terra-CoT.

System Prompt for Reasoning Mode. To facilitate pixel-grounded reasoning, we modify the system prompt of InternVL-3 to encourage structured thinking with segmentation. The modified prompt instructs the model to: (1) generate reasoning steps within `<think>` tags, where it can insert `[SEG]` tokens to trigger mask generation; (2) provide the final answer within `<answer>` tags. The complete system prompt is:

System Prompt for TerraScope

```
A conversation between User and
Assistant. The user asks a question,
and the Assistant solves it. The
Assistant first thinks about the
reasoning process in their mind,
generating segmentation masks when
needed using [SEG] tokens, and
then provides the user a concise
final answer in a short word or
phrase. The reasoning process and
answer are enclosed within <think>
</think> and <answer> </answer>
tags, respectively, i.e., <think>
reasoning process with [SEG] for
segmentation </think><answer> answer
here </answer>.
```

Example Prompt with Pixel-Grounded Reasoning:

Example Output

User: What percentage of the image is covered by forest?

Assistant:
`<think>`
To answer this question, I need to identify and measure the forest area in the image. Let me first segment the forest regions. [SEG]

[Mask generated for forest regions]

Now I can see the forest areas clearly marked. Let me calculate the coverage:
- The segmented forest region covers approximately 3,450 pixels
- The total image area is about 10,000 pixels
- Coverage percentage = (3,450 / 10,000) × 100% = 34.5%

Therefore, forest covers approximately 35% of the image.
`</think>`

`<answer>`35%`</answer>`

The full prompt template including additional few-shot examples is provided in our code repository.

F.2. Evaluation Details on TerraScope-Bench

Baseline Models. We evaluate TerraScope-Bench on three categories of vision-language models:

- **General-purpose VLMs**: We include proprietary models (GPT-4o) and open-source models (LLaVA-OV), Qwen2.5-VL, InternVL-3. For GPT-4o, we utilize its multi-image API to handle multi-temporal inputs for the Building Change Estimation task.
- **Reasoning-enhanced VLMs**: We evaluate two models with explicit reasoning capabilities: GLM-4.1V-Think and Qwen3-VL-Think, which generate intermediate reasoning steps before final answers.
- **EO-specific VLMs**: We include domain-adapted models fine-tuned on earth observation data: GeoChat, TEOChat, LHRS-Bot, EarthDial, and EarthMind.

Evaluation Protocol. All tasks in TerraScope-Bench are formatted as multiple-choice questions with 2 or 4 options (A, B, C, D). We use a unified prompt template across all evaluated models, requesting them to select the correct option. To ensure reliable option extraction, we incorporate **option prediction guidance** in the prompt: *"Please respond with only the option letter (A, B, C, or D) corresponding to your answer."* Since some models have limited instruction-following ability and may generate verbose explanations instead of direct option letters, we implement post-processing using regex patterns (e.g., `r'\b[A-D]\b'`) to extract the predicted option from model outputs. If multiple option letters appear, we select

the first occurrence; if no valid option is found, the prediction is marked as incorrect.

Multi-temporal Handling. For the Building Change Estimation task, which requires comparing pre-disaster and post-disaster imagery:

- Proprietary models (GPT-4o): Use multi-image input API
- Open-source models: Concatenate images horizontally or process as separate frames
- Models without multi-image support: Provide both images sequentially in the conversation

Evaluation Metrics. We compute accuracy by exact matching between predicted option letters and ground-truth answers. For each task type, we report:

- Per-task accuracy: Percentage of correct predictions for each task
- Overall accuracy: Macro-average across all six tasks

Implementation Details.

- For open-source models, we use their official repositories and recommended inference settings
- For proprietary APIs (GPT-4o), we set temperature=0 for deterministic outputs
- All evaluations use greedy decoding (top-p=1.0, temperature=0)

To ensure fair comparison, we fine-tune baseline models on our Terra-CoT dataset with appropriate adaptations:

- **InternVL-3**: We remove all special tokens (`<think>`, `</think>`, `[SEG]`) from the training data and perform standard supervised fine-tuning using the official training scripts. The model is trained to directly predict answers without explicit reasoning traces or segmentation masks.
- **GLM-4.1V-Think**: We preserve the thinking mode structure (`<think>`, `</think>`) but remove the `[SEG]` token, as this model does not support pixel-level grounding. We use the official training pipeline combining SFT (Supervised Fine-Tuning) and RLVR (Reinforcement Learning with Verifiable Rewards) as described in [11].

This design allows us to assess whether baseline models can benefit from our training data while maintaining their original architectures. The complete evaluation code, prompts, and output parsing scripts are available in our repository.

G. Efficiency Analysis

We analyze TerraScope's computational efficiency from multiple perspectives, including inference time, memory consumption, parameter count, and the impact of pixel-grounded reasoning on computational cost.

G.1. Model Complexity

Tab. C compares TerraScope with mainstream baseline models in terms of model size.

TerraScope integrates the SAM-2 image encoder (224.4M parameters) and mask decoder (3.9M parameters) to enable pixel-level grounding. These two modules together introduce only about 0.228B additional parameters, increasing the overall model size from 8.1B (base InternVL-3) to 8.3B. This corresponds to a parameter overhead of merely $\sim$2.8%.

Model	Total Params	Additional Modules
GPT-4o	-	-
Qwen2.5-VL-7B	7.6B	-
InternVL-3-8B	8.1B	-
GLM-4.1V-9B	9.4B	-
LLaVA-OV-7B	7.2B	-
TerraScope-8B	**8.3B**	**SAM-2 (0.228B)**
– Base InternVL-3	8.1B	-
– SAM-2 image encoder	-	0.224B
– SAM-2 mask decoder	-	0.004B

Table C. Model complexity comparison. TerraScope adds a lightweight pixel-level grounding module on top of InternVL-3.

Crucially, the added segmentation components are extremely lightweight compared to the backbone large multimodal model: the extra 0.228B parameters account for only a small fraction of the total parameter budget, while the vast majority of parameters still reside in the LLM. In other words, TerraScope incurs only a minimal parameter increase yet gains the substantial benefit of being able to produce verifiable, pixel-level segmentation masks at each reasoning step.

G.2. Inference Time Analysis

We measure inference time on a single NVIDIA A100 80GB GPU with batch size 1. Tab. D reports the average time per sample on TerraScope-Bench.

Model	Avg. Time (s)
InternVL-3-8B	0.85
Qwen2.5-VL-7B	0.92
TerraScope-8B	**2.48**
GLM-4.1V-9B	2.60

Table D. Average inference time per sample (seconds).

TerraScope achieves faster inference than GLM-4.1V-9B (2.4s vs 2.6s) despite generating additional segmentation masks. We identify two key efficiency advantages: First, TerraScope performs deterministic reasoning with structured output (`<think>` and `<answer>` tags), while GLM-4.1V tends to generate overly verbose reasoning traces with significantly more tokens. Second, our interleaved mask injection is highly efficient—masked visual features are directly inserted into the KV cache without re-encoding through the vision encoder, avoiding redundant visual pro-

cessing. InternVL-3 remains the fastest (0.85s) as it generates answers directly without reasoning, but lacks both reasoning transparency and pixel-level grounding capabilities that TerraScope provides.

G.3. Memory Consumption

We profile GPU memory usage during inference on a single NVIDIA A100 80GB GPU. Tab. E shows peak memory consumption with different numbers of generated masks.

Model	1 Mask	2 Masks	3+ Masks
InternVL-3-8B	18.2	18.3	18.2
Qwen2.5-VL-7B	16.8	17.0	17.0
TerraScope-8B	**22.4**	**23.1**	**24.2**

Table E. GPU memory consumption (GB) on NVIDIA A100.

TerraScope requires approximately 22% more memory than InternVL-3 (22.4GB vs 18.2GB for single-mask cases), primarily due to the SAM-2 decoder weights (3.9GB). Memory consumption increases with the number of generated masks, as each mask adds approximately 0.7GB for storing mask features and intermediate activations. In contrast, baseline models (InternVL-3, Qwen2.5-VL) maintain constant memory usage regardless of output complexity, as they do not generate pixel-level grounding. The memory overhead is acceptable given TerraScope's additional capability of producing verifiable segmentation masks.

H. Additional Experimental Results

Beyond the geospatial reasoning tasks reported in Sec. 5, we evaluate TerraScope on additional benchmarks to demonstrate its generalization ability across diverse earth observation tasks.

H.1. Comprehensive Results on Landsat30-AU

Tab. F presents complete results on all eight task types in Landsat30-AU. The benchmark includes Agro-Phenology Reasoning (APR) for agricultural growth stages, Cloud-Occlusion Assessment (COA) for detecting cloud coverage, Dominant Land-Cover (DLC) for identifying main land types, Fine-Object Detectability (FOD) for detecting small objects, Macro-Object Presence (MOP) for large-scale objects, Object Counting (NUM), Spatial Relationship (SRI) for spatial layout reasoning, and Urban Scale Recognition (USR) for classifying settlement scale. TerraScope achieves competitive performance across all task types, with particularly strong results on fine-grained visual tasks requiring precise spatial understanding, such as Cloud-Occlusion Assessment (COA) and Fine-Object Detectability (FOD). This demonstrates that pixel-grounded reasoning capabilities transfer effectively to general earth observation understanding tasks.

H.2. Results on RSVQA and Scene Classification

Tab. G reports performance on RSVQA-LR [29] and BigEarthNet scene classification [5]. On RSVQA-LR, TerraScope performs slightly below EarthDial. We attribute this to the difference in training data scale—LHRS-Bot and EarthDial were trained on significantly larger VQA datasets, which benefits general question-answering tasks. On BigEarthNet scene classification, TerraScope achieves competitive accuracy comparable to EarthDial, demonstrating effective transfer learning despite being primarily designed for pixel-grounded reasoning.

H.3. Complete Results on DisasterM3

We report comprehensive results on DisasterM3, which includes both optical-optical and optical-SAR multi-modal evaluation. In the main paper (Sec. 5), we reported only optical-optical results as most baseline models do not support SAR imagery. Tab. H presents results on both modality configurations. TerraScope is the only model capable of handling optical-SAR multi-modal inputs through adaptive modality selection. On optical-optical pairs, TerraScope achieves competitive performance with EO-specific baselines. On optical-SAR pairs, TerraScope demonstrates its unique capability to leverage complementary information from heterogeneous modalities for damage assessment.

I. More Ablation Studies

Effectiveness of Two-Stage Training. TerraScope employs a two-stage training strategy: Stage 1 performs grounded pretraining on 2M referring expression segmentation pairs to train the mask decoder, and Stage 2 applies instruction tuning on Terra-CoT to jointly optimize the projector, LLM, and mask decoder. Tab. J compares models with and without Stage 1 pretraining on three benchmarks. The results demonstrate that grounded pretraining establishes foundational pixel-level grounding capability, which substantially improves performance on pixel-grounded reasoning tasks and also benefits general EO understanding and disaster assessment tasks.

Effectiveness of Terra-CoT data composition. Our Terra-CoT dataset is synthesized using a hierarchical data synthesis strategy combining three data types: L1-level VQA, L2-Level VQA, and captioning. To validate the effectiveness of this composition, we train TerraScope with different data mixtures in Tab. I. First, training with Terra-Cap (captioning only) provides limited instruction-following capability, as the model struggles with both perception and reasoning tasks. Second, adding L1-level VQA establishes

Model	Size	APR	COA	DLC	FOD	MOP	NUM	SRI	USR	Overall
EarthDial	4B	23.49	10.34	75.27	**99.00**	61.16	43.62	51.24	15.52	48.29
RS-LLaVA	7B	68.57	80.88	71.24	87.00	63.09	49.85	26.17	10.34	57.24
MiMo	7B	40.00	45.77	92.47	93.33	**84.30**	**61.42**	**94.21**	**88.97**	75.55
GLM-4.1V	9B	45.71	36.36	72.85	62.67	67.49	58.63	69.97	88.28	62.87
Qwen2.5-V	7B	29.84	89.66	**94.09**	71.67	76.03	53.12	92.84	82.07	74.28
LLaVA-OV	8B	39.37	79.00	83.06	59.00	72.45	46.59	85.12	10.34	60.96
TerraScope	8B	**69.84**	**98.12**	83.06	87.67	61.98	60.82	91.12	85.52	**79.36**

Table F. Performance on the VQA task on Landsat30-AU. Bold indicates the best score.

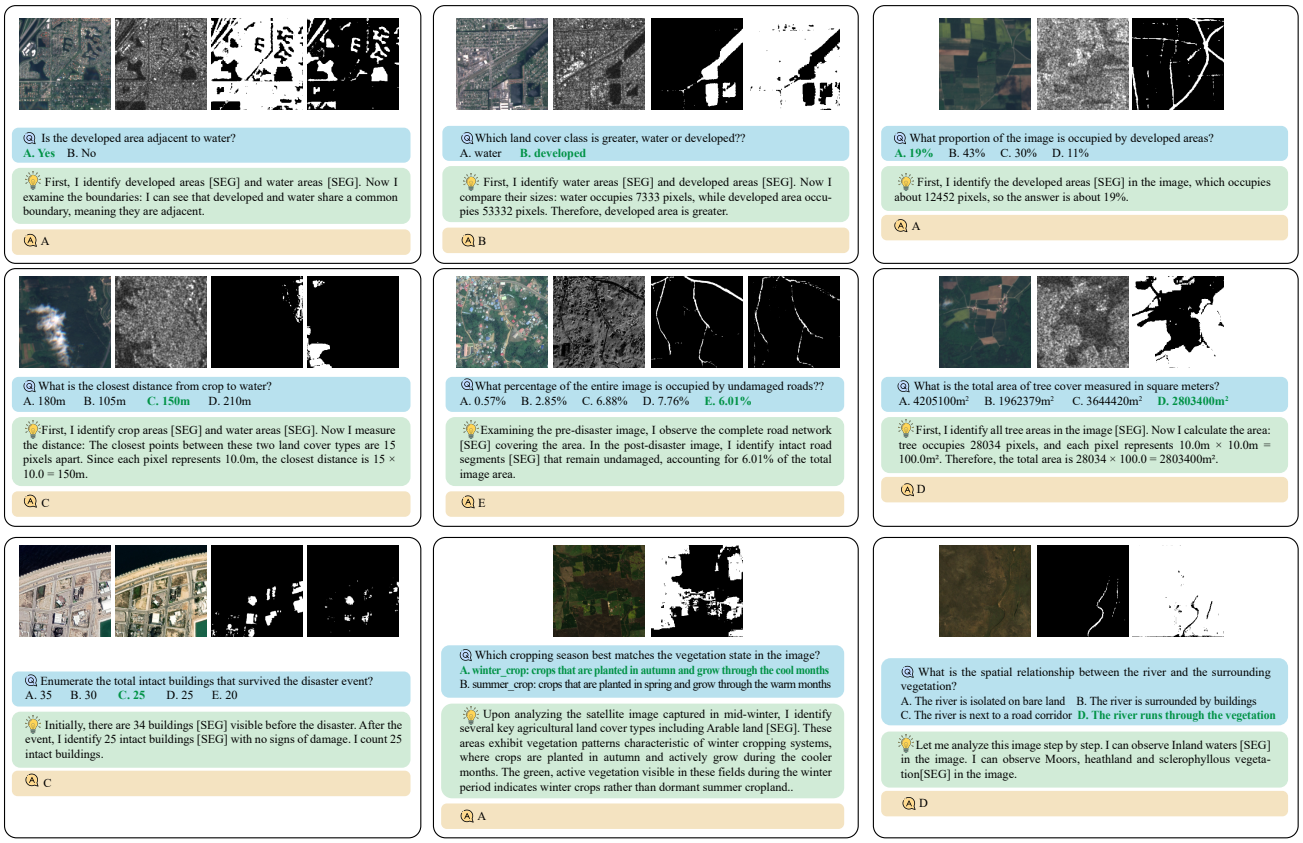


Figure E. More qualitative results of TerraScope.

Model	RSVQA-LR	BigEarthNet
GeoChat	90.7	20.4
LHRS-Bot	89.2	-
EarthDial	**92.7**	68.8
TerraScope	91.4	**69.2**

Table G. Results on RSVQA and scene classification.

Model	Optical-SAR		
	BDC	DRE	Avg
LLaVA-OV	22.2	19.4	20.8
TEOChat	18.4	9.4	13.9
InternVL3-8B	20.7	18.4	19.6
EarthDial	19.5	10.2	14.9
TerraScope	**50.4**	**32.6**	**41.5**

Table H. Optical-SAR results on DisasterM3 benchmark.

foundational pixel-grounded visual understanding, significantly improving performance on tasks requiring accurate segmentation. However, this perception-focused training still lacks complex reasoning capabilities, resulting in poor performance on challenging tasks like those in LandSat30-AU that require multi-step spatial reasoning. Third, incor-

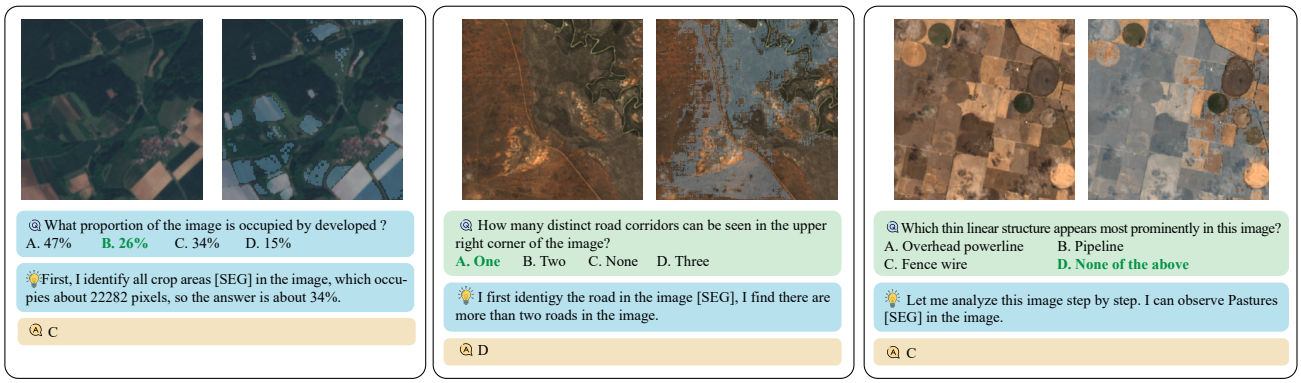


Figure F. Failure cases of TerraScope.

Data	TerraBench.	Landsat.	Disaster.
Cap-CoT	42.8	50.1	26.9
Cap-CoT + L1-VQA	66.7	61.0	46.2
Cap-CoT + L1-VQA + L2-VQA	**68.9**	**73.9**	**46.5**

Table I. Ablations of Terra-CoT.

Training Strategy	TerraScope-Bench	Landsat30-AU	DisasterM3
w/o Grounded Pretrain	65.4	71.8	43.0
w/ Grounded Pretrain	**68.9**	**73.9**	**46.5**

Table J. Ablation study on grounded pretraining.

Multi-modal Encoding	Masked Feature Interleaving	TerraScope-Bench Mean IoU (%)	TerraScope-Bench Accuracy (%)	Efficiency
Optical only	Optical only	53.4	65.0	High
Optical only	Concat Opt+SAR	53.5	67.6	Low
Optical only	Adaptive selection	53.1	67.4	High
Concat Opt+SAR	Optical only	56.8	69.2	High
Concat Opt+SAR	Concat Opt+SAR	**57.2**	**73.0**	Low
Concat Opt+SAR	Adaptive selection	**57.2**	72.6	High

Table K. Ablation study on multi-modal reasoning. "Efficiency" indicates inference efficiency: "High" for methods with shorter context length (single modality or adaptive selection), "Low" for concatenation methods that double the visual token count.

porating L2-Level data enables strong generalization across diverse task types. The full Terra-CoT mixture achieves the best overall performance, with improvements scaling consistently as we increase the proportion of reasoning data.

Ablations about multi-modal reasoning. We investigate how multi-modal data (optical and SAR) contributes to TerraScope's performance. We design ablation experiments by controlling two aspects: (1) **Multi-modal encoding**: whether to concatenate optical and SAR features as input to the LLM during initial image encoding; (2) **Masked feature interleaving**: how to inject masked visual features during reasoning steps—using optical only, concatenating both modalities, or adaptively selecting based on relevance scores (Eq. 4-5).

Tab. K presents results on TerraScope-Bench, evaluated on both segmentation quality (mean IoU) and final answer accuracy.

Our ablation study reveals two key findings. First, multi-modal encoding is essential for both accurate segmentation and reasoning. Concatenating optical and SAR features as initial input substantially improves performance compared to optical-only encoding, demonstrating that the LLM benefits from complementary multi-modal representations from the beginning of reasoning. Second, the masked feature injection strategy during reasoning steps also matters. Both concatenation and adaptive selection of masked features significantly outperform optical-only injection. While concatenation achieves slightly higher answer accuracy, adaptive selection demonstrates a favorable trade-off: it maintains comparable segmentation quality and nearly equivalent reasoning performance while significantly reducing context length by dynamically selecting only the most informative modality at each spatial location. This reduction in context length translates to substantial savings in memory consumption and inference time, making adaptive selection the more practical choice for deployment.

J. Additional Visualizations and Failure Analysis

J.1. Qualitative Results

Fig. E presents additional qualitative results demonstrating TerraScope's capabilities across diverse scenarios. The visualizations show that TerraScope can perform pixel-grounded reasoning on: (1) single-modality optical imagery, generating accurate segmentation masks and spatial analysis; (2) multi-modal optical-SAR fusion, adaptively selecting the most informative modality for each spatial region; and (3) temporal change detection, providing chain-of-thought reasoning traces that explain land cover changes with supporting visual evidence. These results validate TerraScope's versatility in handling different data modalities and temporal information while maintaining pixel-level grounding throughout the reasoning process.

J.2. Failure Cases and Analysis

Fig. F presents typical failure cases to understand TerraScope's limitations. We identify two primary failure modes:
(1) Limited spectral information. TerraScope currently processes only RGB bands as input, discarding additional spectral channels available in multispectral sensors like Sentinel-2 (which provides 13 bands including near-infrared, red-edge, and shortwave infrared). This limitation makes it challenging to distinguish spectrally similar land cover types that appear visually identical in RGB but exhibit distinct spectral signatures in other bands. For example, certain crop types or vegetation health conditions that are easily separable using NDVI or red-edge indices become ambiguous in RGB-only input, leading to incorrect segmentation and subsequent reasoning errors.
(2) Error propagation from segmentation. For scenes containing small or low-contrast objects (e.g., narrow roads, sparse buildings, thin water channels), the mask decoder may produce inaccurate segmentation due to insufficient visual salience. These segmentation errors directly propagate to the reasoning stage: when spatial claims are grounded in incorrect masks, the derived answers become unreliable even if the reasoning logic is sound. This highlights the critical dependency of pixel-grounded reasoning on high-quality segmentation, particularly for fine-grained objects in complex landscapes.

Future improvements could address these limitations by: (1) extending the vision encoder to process full multispectral inputs rather than RGB only, enabling better spectral discrimination; (2) incorporating uncertainty estimation in the segmentation module to flag low-confidence masks and trigger refinement; and (3) developing iterative refinement mechanisms that allow the model to correct initial segmentation errors through multi-step reasoning.