POLARTOOLS: BENCHMARKING TOOL-AUGMENTED MULTIMODAL AGENTS FOR POLAR REMOTE SENSING ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in multimodal large language models (MLLMs) and autonomous agents have opened up new opportunities for automated scientific analysis. However, their capabilities in specialized domains such as polar remote sensing remain largely unexplored. Existing polar benchmarks primarily evaluate perception models, while it remains unclear whether domain-specific tools provide additional value beyond the growing code-generation capabilities of MLLMs. To address this gap, we introduce PolarTools, a benchmark for evaluating tool-augmented multimodal agents in polar remote sensing. PolarTools contains 710 question-answer instances covering iceberg, sea-ice, and polar oil-spill analysis, spanning perception and evidence interpretation; object and spatial structure analysis; temporal change and motion; geometric measurement and counting; quantitative product diagnostics; and preprocessing and cartography. We further develop a unified environment integrating 113 domain-specific tools for preprocessing, target extraction, geospatial analysis, temporal modeling, and visualization. Using PolarTools, we design two evaluation modes: General Solving (GS), in which agents are provided with a Python environment in a Docker container and rely on their own capabilities to solve tasks, and Tool-Augmented Solving (TA), in which agents additionally have access to polar-specific tools. We conduct experiments on 13 multimodal model configurations to analyze how domain-specific tools affect task completion, execution behavior, and efficiency. Unlike trajectory-matching evaluations, PolarTools evaluates agents based on task requirements and verifiable outputs, allowing diverse valid solutions while enabling analysis of whether domain-specific tools improve task completion and efficiency. PolarTools provides a testbed for studying multimodal agents in polar scientific workflows and the value of domain-specific tool augmentation.

1 INTRODUCTION

Remote-sensing visual question answering enables users to express image-grounded tasks in natural language (Lobry et al., 2020; Wang et al., 2023). Recent agent systems further connect scientific questions with executable remote sensing operations through external tools (Kuckreja et al., 2024; Feng et al., 2026; Shabbir et al., 2026b), raising a practical question: can multimodal agents reliably turn polar observations and scientific requests into verifiable results? At the same time, as modern MLLMs increasingly gain strong code-generation and program-execution capabilities, a second question arises: whether domain-specific tools still provide additional value beyond what agents can achieve through general-purpose programming.

Existing polar benchmarks assess important components of this problem. AutoICE evaluates sea ice concentration, development stage, and floe size (Stokholm et al., 2024), while IceBench and IceFMBench focus on sea ice classification and segmentation (Alkaee Taleghan et al., 2025a;b). These datasets support research on polar perception, but their predefined prediction tasks do not examine how an agent interprets a request or organizes the operations needed to answer it. Broader remote-sensing vision-language benchmarks, including VRSBench and GEOBench-VLM, cover diverse

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

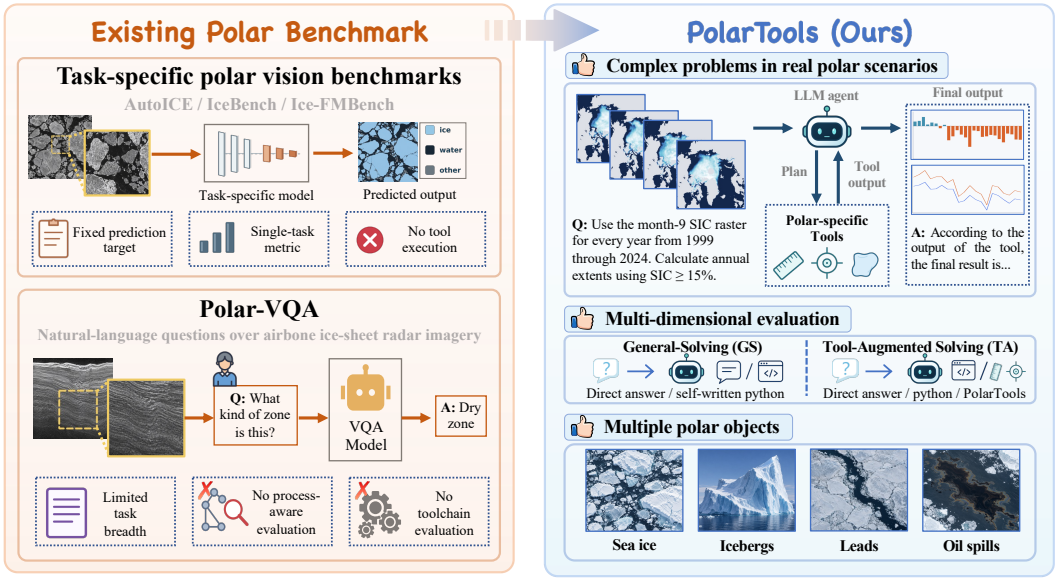


Figure 1: Scope of representative polar benchmarks and PolarTools. Existing task-specific benchmarks assess outputs such as sea-ice predictions, while Polar-VQA evaluates natural-language answers to polar radar imagery. PolarTools instead presents questions over multiple polar observations and exposes callable analysis tools, intermediate results, and complementary evaluation settings.

imagery and geospatial tasks (Li et al., 2024a; Danish et al., 2025), but do not specifically evaluate executable polar analysis. Polar-VQA introduces natural language questions about polar radar imagery, including questions about sensors, locations, and wet or dry zones (Sarkar & Rahnemoonfar, 2023). It evaluates language-based interaction with polar observations, yet its answers do not require an executable analytical workflow. Current benchmarks therefore provide limited evidence about whether multimodal agents can reliably complete question-driven polar analysis and produce verifiable results.

The difference between answering a question and carrying out the analysis behind it has motivated benchmarks with executable tools (Wang et al., 2024; Qin et al., 2024). Specialized perception models can locate or segment a target (Lai et al., 2024), geospatial operations can convert these outputs into physical measurements (Gorelick et al., 2017), and numerical programs can derive statistics from remote sensing products (Gao et al., 2023; Chen et al., 2022). Earth-Agent and ThinkGeo evaluate agents that select and execute such operations in general remote sensing tasks (Feng et al., 2026; Shabbir et al., 2026a), while AgroTools studies tool use in agricultural vision tasks (Ye et al., 2026). With these capabilities, an agent can segment an object before measuring its area, match observations before estimating movement, or process raster values before calculating a quantitative indicator. Although these tasks involve different inputs and operations, they share a common requirement: the agent must translate a scientific request into executable steps and integrate their outputs into a verifiable answer. This makes polar remote sensing a relevant setting for studying multimodal agents, as many analytical questions require precise perception, spatial reasoning, and scientific computation. However, such workflows do not necessarily require pre-built domain-specific tools. Increasingly capable MLLMs can also generate Python code and use general-purpose libraries to implement parts of the same analysis. This motivates a controlled comparison between general-purpose coding and domain-specific tool support to examine whether specialized tools provide additional value for executable polar analysis.

However, existing benchmarks do not jointly capture the requirements needed to evaluate these questions in polar remote sensing. As illustrated in Figure 1, we summarize these differences into three challenges. *(i) Heterogeneous data and tasks:* Polar questions may involve optical images, SAR observations, georeferenced products, spatial metadata, or observations acquired at different times, whereas most polar benchmarks isolate one prediction task or a limited set of visual questions. *(ii) Scientifically constrained tool execution:* Many polar questions ask for physical quantities,

such as target area, drift, and concentration statistics, that cannot be derived reliably from direct visual reasoning alone. Their computation depends on sensor-specific choices. Therefore, evaluation must assess whether an agent performs appropriate operations, uses product-aware parameters, and correctly interprets the resulting outputs, regardless of whether these operations are implemented through self-generated code or domain-specific tools. *(iii) Diverse and variable targets:* Sea ice, icebergs, leads, and oil spills differ in scale, appearance, spatial structure, and temporal behavior. Their observed characteristics also vary with sensor type, acquisition conditions, season, and location, so an operation suited to one target or product may fail on another. As a result, there remains no unified benchmark for evaluating tool-augmented multimodal agents across diverse, executable, and image-grounded polar tasks.

To address this gap, we introduce PolarTools, a benchmark for tool-augmented multimodal agents in polar remote sensing.PolarTools is designed to address both questions: whether multimodal agents can reliably complete executable polar analysis, and whether access to domain-specific tools provides additional benefits beyond their general-purpose coding capabilities. PolarTools organizes questions into six task families: perception and evidence interpretation; object and spatial structure analysis; temporal change and motion; geometric measurement and counting; quantitative product diagnostics; and preprocessing and cartography. Each question is grounded in polar observations and the metadata needed to interpret them, with a specified output and a reference for evaluation. The accompanying environment provides tools for product processing, target perception, geospatial measurement, temporal analysis, numerical computation, and visualization. We evaluate agents on the same eligible questions under two autonomous settings. In General Solving (GS), an agent can answer directly or execute self-written Python code. In Tool-Augmented Solving (TA), it can also call the polar analysis tools provided by PolarTools. We evaluate 13 model configurations under the GS and TA settings to examine how access to polar-specific tools affects task performance and to identify bottlenecks in tool selection, argument specification, result interpretation, and answer synthesis. Overall, the main contributions of this work are summarized as follows:

- We introduce PolarTools, a benchmark for tool-augmented multimodal agents in polar remote sensing. It contains 710 question-answer instances with 1,230 image or raster-product uses across six analytical categories, drawing on public observations, independently processed data, and our own UAV imagery. We additionally curate 9,865 segmentation samples to support model-backed perception tools.
- We develop an unified toolkit of 113 callable tools, covering sensor-aware preprocessing, target perception, geospatial measurement, temporal analysis, and visualization. Its explicit inputs and inspectable intermediate outputs support multi-step execution and process-level diagnosis without enforcing a single reference trajectory.
- We benchmark 13 open-source and closed-source MLLMs on the same questions under General Solving (GS) and Tool-Augmented Solving (TA), and analyze how domain-specific tools affect task performance and execution efficiency. Results show that current models remain unreliable in multi-step polar analysis.

2 Related Work

2.1 Remote Sensing Multimodal Benchmarks

Remote sensing multimodal benchmarks have expanded along several dimensions, including task coverage, spatial detail, and sensor diversity. RSVQA establishes visual question answering over remote sensing imagery, covering object presence, counting, and spatial relationships (Lobry et al., 2020). EarthVQA emphasizes object-centric relational reasoning (Wang et al., 2023), while VRS-Bench integrates captioning, visual grounding, and question answering within a unified dataset (Li et al., 2024b).

Broader evaluations include GEOBench-VLM, which covers localization, segmentation, and temporal analysis (Danish et al., 2025), and XLRS-Bench, which examines perception and reasoning in extremely large, high-resolution imagery (Wang et al., 2025). GeoMMBench further incorporates diverse geoscience disciplines and sensor modalities, alongside a tool-equipped agent framework (Xiao et al., 2026). These efforts provide increasingly comprehensive assessments of multimodal

Dataset	Polar-Specific	Natural-Language Eval.	Cross-Modal EO Data	Executable Tools	Agent-Written Code
ThinkGeo	✗	✓	✓	✓	✗
Earth-Bench	✗	✓	✓	✓	✗
AgroTools	✗	✓	✓	✓	✗
OpenEarth-Bench	✗	✓	✓	✓	✗
AutoICE	✓	✗	✓	✗	✓
Ice-FMBench	✓	✗	✗	✗	✓
Polar-VQA	✓	✓	✗	✗	✓
PolarTools (Ours)	✓	✓	✓	✓	✓



Figure 2: Benchmark Comparison and Task Overview. The left panel compares PolarTools with representative EO and polar benchmarks. The right panel shows the distribution of 710 questions across six task families and twelve subcategories.

geospatial understanding. PolarTools builds on these capabilities by evaluating their use in executable polar analysis workflows, where task completion depends on combining perception with measurement and producing the requested numerical, spatial, and explanatory outputs.

2.2 Tool-Augmented Benchmarks

General tool-use benchmarks evaluate how language models select and combine external capabilities to fulfill user requests. ToolBench provides instructions and solution trajectories involving real-world APIs, including tasks that require multiple tools (Qin et al., 2024). GTA extends evaluation to multimodal inputs and deployed tools, using queries that specify user objectives while leaving the required operations implicit (Wang et al., 2024). Together, these benchmarks support evaluating tool use through interactions and task execution.

Domain-specific benchmarks introduce specialized operations and application requirements. ThinkGeo evaluates tool-augmented agents on remote sensing tasks requiring multi-step planning (Shabbir et al., 2026a). Earth-Bench, introduced with Earth-Agent, incorporates RGB imagery, spectral observations, and derived products, and evaluates intermediate decisions alongside final outcomes (Feng et al., 2026). OpenEarthAgent provides executable geospatial trajectories for agent training and evaluation (Shabbir et al., 2026b), while Earth-Bench-Pro extends evaluation to open-world workflows that include observation acquisition (Lv et al., 2026). In agriculture, AgroTools couples image-based tasks with specialized tools and reference trajectories, assessing both execution processes and final answers (Ye et al., 2026). PolarTools adopts this joint assessment of execution and outcomes for tasks involving icebergs, sea ice, and leads. Its focus is on composing specialized perception with spatial and temporal analysis across heterogeneous polar observations. By bringing these tasks into a common tool environment and evaluation protocol, PolarTools supports studying how tool-use decisions affect the completion of polar analytical workflows.

OpenEarth-Agent explicitly studies the transition from tool calling to tool creation (OpenEarth-Bench). Accordingly, we do not claim that allowing generated code is new by itself. PolarTools uses that capability as a capable baseline for evaluating the additional value of polar-specific processing and scientific output validation.

3 Task and Tool Design

3.1 Polar Task Taxonomy and Task Format

PolarTools assigns each question a primary scientific objective and retains secondary capability tags. The six families are **perception and evidence interpretation** (93 questions), **object and spatial structure analysis** (144), **temporal change and motion** (176), **geometric measurement and counting** (65), **quantitative product diagnostics** (109), and **preprocessing and cartography** (123). Perception includes inference from raw imagery and source-grounded knowledge questions, recorded as distinct subtypes. Object/spatial analysis operates on supplied masks or geometries and checks inventories, topology, and spatial support. Temporal tasks concern correspondence,

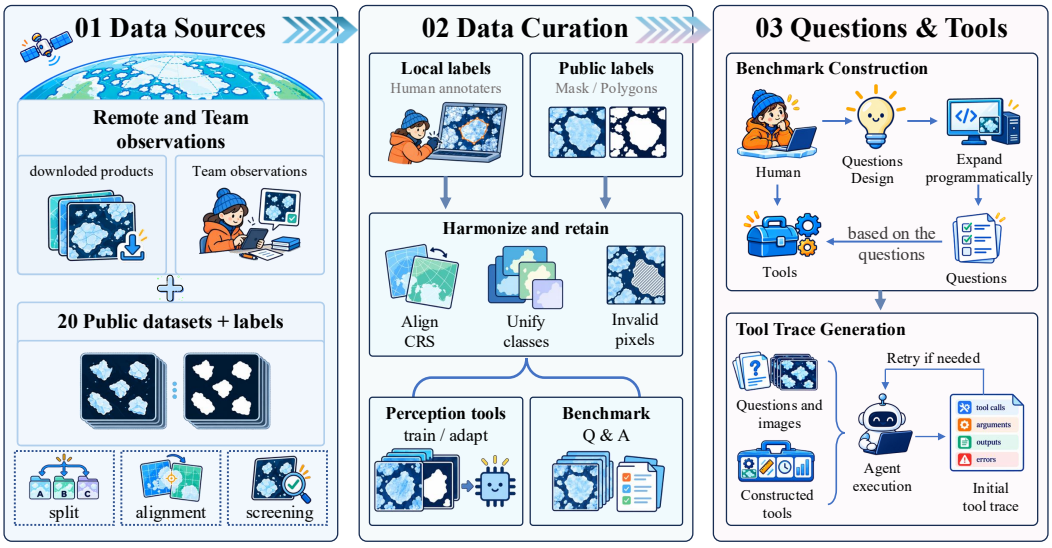


Figure 3: Construction workflow: source curation, question and tool design, and recorded agent execution. Reference checks and version freezing distinguish construction candidates from scored evaluation instances.

displacement, and change; geometric tasks concern explicit measurements. Product diagnostics concern physical decoding, quality selection, and support-aware summaries. Preprocessing and cartography cover preparation, coordinate transformations, and map products. Category assignment is based on the requested scientific result, so one solution may use tools from multiple functional groups. This revised classification is an analytical grouping of the existing inventory, not a claim of an exactly balanced task distribution.

Each instance supplies a natural-language question q and observations X, which may include optical or SAR images, georeferenced rasters, derived products, masks, acquisition times, and other metadata. The requested answer y may be a category, location, region, numerical value, time series, or explanatory conclusion. Questions specify the scientific objective and necessary conditions without prescribing a unique sequence of tool calls. For example, Figure 1 asks for annual September sea-ice extents from concentration rasters using a 15% threshold. Answering it requires interpreting the product values and valid pixels before calculating and aggregating area; identifying ice in a preview image alone does not produce the requested series. Section 4 describes how the instances are constructed and assessed.

3.2 Polar Tool Environment and Interaction

The tool environment exposes operations needed to turn heterogeneous polar observations into checkable answers. Its interfaces are organized into five functional groups: ***data access and preprocessing***; ***target perception and region extraction***; ***measurement and statistical analysis***; ***temporal change and motion analysis***; and ***visualization and export***. The distinction matters because a perception output is often only an intermediate result. An iceberg mask, for example, must be related to the image grid and coordinate reference system before its area can be measured or the object can be compared with a later observation. Accordingly, the environment combines model-backed perception interfaces with product decoding, masking, geometric computation, temporal analysis, and result export. Detailed interfaces and their availability requirements are specified in the appendix.

The sea-ice question in Figure 1 illustrates this composition. One valid route uses `ProductRead` to decode each concentration raster with its declared scale and invalid-value codes, `ValidMaskBuild` to establish the pixels eligible for analysis, and `IceAreaExtent` to calculate thresholded extent using pixel-area weights. The annual values can then be assembled and displayed with `ChartRender`. For an iceberg-change question, a different route may combine `IcebergSegment`, georeferenced area measurement, and `ObjectMatch` or

`IceMotionEstimate`. These are examples of possible solutions, not mandatory reference sequences: the agent must choose operations appropriate to the supplied observations and requested answer.

In autonomous execution, the agent selects a tool and its arguments, observes the returned value or artifact, and decides its next action using the updated history, following a ReAct-style interaction loop (Yao et al., 2023). The common runtime records calls, arguments, outputs, and errors, while keeping private reference answers and evaluator functions outside the agent's tool environment. Tool arguments expose consequential scientific choices, including SAR linear power versus decibels, concentration scaling, invalid pixels, coordinate reference systems, acquisition times, and decision thresholds. Incompatible inputs produce an explicit error or documented limitation; model-backed interfaces report unavailability when required weights are absent. Such behavior lets an evaluation distinguish a planning or argument error from an incorrect final calculation, without treating candidate masks or matches as confirmed scientific observations.

4 PolarTools Benchmark

4.1 Benchmark Construction

We construct PolarTools from two complementary data streams: observations and annotations collected by our team, and observations or labels obtained from publicly released datasets. For each source, we record the sensor, acquisition, geographic region, original annotation, license, and subsequent processing. The construction inventory contains 710 question records. Source observations and questions are counted separately: one observation may support several distinct analytical questions. We inspect source labels, harmonize their coordinate systems and class definitions, and retain invalid-data masks and acquisition metadata when these affect the answer.

The curated material supports both perception-tool development and benchmark construction. These uses require a source-level provenance audit: parent scenes, overlapping crops, and temporal near-duplicates must be tracked before claims of held-out generalization are made. The present results do not establish a completed benchmark-wide training/evaluation separation. For each benchmark instance, annotators formulate a natural-language request, select the files that an agent may inspect, specify the expected output fields, and construct a reference answer or artifact with an explicit scoring rule. Questions are reviewed for scientific interpretation, answerability, and consistency between the stated inputs and reference. A supplied annotation mask, when present, is treated as an input to a measurement question rather than as hidden ground truth for a segmentation question. Multiple questions derived from the same parent observation are tracked as related instances.

Some tasks call for pixel-level perception that cannot be reduced to reading an existing product. PolarTools therefore includes model-backed interfaces alongside deterministic tools for decoding, measurement, and temporal analysis (Section 3.2). The current iceberg segmentation implementation, BergSeg, has separate SAR and optical branches; its training sources, comparators, ablations, and limitations are reported in Appendix C.2. Claims of superiority over other polar models or general-purpose segmenters are reserved for matched evaluations on the same held-out images, with the same input information and clearly specified prompt or training protocol. The more detailed BergSeg study reflects the work of assembling and reconciling multiple SAR and optical iceberg-label sources and evaluating two separate branches. Sea-ice and oil-slick segmentation instead adapt established architectures and are reported more compactly in Appendix C.3. For ice-lead segmentation, `LeadSegment` calls the published, pretrained LeadNet model (Yang et al., 2026); we do not train an ice-lead model.

4.2 Evaluation Protocol

We evaluate each agent under two autonomous modes. In *General Solving* (GS), the agent receives the task files and may answer directly or write and execute its own Python code in a general-purpose environment; PolarTools interfaces are unavailable. In *Tool-Augmented Solving* (TA; recorded as AE in the implementation), the agent retains both options and can additionally invoke the PolarTools catalog. Both modes require the agent to plan its own solution; TA does not prescribe a reference tool sequence or require a tool call. The paired runs use the same model, task wording, general-purpose environment, resource limits, and interaction budget. Access to PolarTools is the controlled

Table 1: PolarTools benchmark composition and task-family performance.

Task family	Questions	Share(%)
Geometric measurement	65	9.2
Object/spatial analysis	144	20.3
Perception/evidence	93	13.1
Preprocessing/cartography	123	17.3
Product diagnostics	109	15.4
Temporal change	176	24.8
Total	710	100

Task family	Pairs	GS	TA	Gain
Geometric measurement	480	65.8	66.7	0.8
Object/spatial analysis	711	35.7	37.0	1.3
Perception/evidence	132	19.7	14.4	-5.3
Preprocessing/cartography	578	40.3	56.1	15.7
Product diagnostics	591	67.5	73.8	6.3
Temporal change/motion	1027	28.2	47.9	19.7

difference. The runtime records code and tool calls, arguments, returned artifacts, and errors. Private reference answers and scoring functions are inaccessible during execution.

Code runs inside a task-isolated Docker environment with Python and the scientific libraries packaged in the frozen image. The task inputs are read-only; generated artifacts remain within the task workspace. GS cannot import the specialist package or its private model weights. Both modes can construct functions and scripts for the current task, but the protocol does not evaluate cross-task persistence or a reusable generated-tool library. A successful `DataOutput` submission terminates execution immediately; no additional model reply is required. Task manifests record call and turn limits, and these limits are matched within each GS/TA comparison rather than assumed identical across historical protocol versions.

Each question specifies a typed output contract and a task-specific reference check. Numeric fields are evaluated with the stated units and tolerances; region outputs are compared with spatial overlap or boundary criteria where applicable; categorical and structured outputs use their corresponding checks. We report answer quality and whole-task correctness over the same eligible question set for GS and TA, together with output-format compliance. Tasks awaiting reference annotation or human judgment are reported separately and are excluded from the scored denominator until resolved. Tool traces support diagnostic analysis of execution and error propagation; a plausible tool-call sequence does not replace a correct final answer.

4.3 Benchmark Statistics

The construction inventory contains 710 questions, distributed as shown in Table 1. The taxonomy identifies each question's primary objective; it does not count individual operations as separate tasks. There are 2,772 annotated scientific checkpoints. Annotation availability is distinct from independent oracle readiness and recorded evidence coverage. Source-scene reuse and task revisions are retained in provenance records, and only compatible versions contribute to paired results.

5 Experiments

5.1 Experimental Setup

Models and snapshot. The current snapshot contains 13 model configurations from six families and 8,405 recorded runs. Results are provisional: models have different completion coverage, and only compatible GS/TA pairs are used for accuracy and tool gain. Compatibility requires identical task, model, and protocol version identifiers. The snapshot yields 3,519 such pairs; it is not a common question subset shared by all models. Consequently, comparisons within a model are paired, whereas cross-model rankings remain sensitive to the questions completed.

Protocol and metrics. Both modes use autonomous planning, general-purpose Python, and the same per-question limits; TA additionally exposes the specialist tools. Temperature is configured as zero. We report five measures: whole-task Accuracy, Critical-Step attainment, paired Time-Saving, per-task execution Failure-Rate, and paired Tool-Gain. Infrastructure failures and unresolved ref-

Table 2: Performance of different LLM backbones under the *GS* and *TA* regimes. Accuracy, Critical-Step, and Failure-Rate are reported as percentages, while Tool-Gain is measured in percentage points. Time-Saving denotes the median relative saving on jointly correct pairs, with positive values indicating that TA is faster. Best results are shown in **bold**, and runner-up results are underlined.

Model	Accuracy↑		Critical-Step↑		Failure-Rate↓		Tool-Gain↑	Time-Saving↑
	GS	TA	GS	TA	GS	TA	Δ	Paired
GPT-4o	12.2	53.8	21.9	52.8	24.0	21.9	**41.6**	23.2
GPT-6-Luna	70.7	67.5	80.2	77.7	3.3	3.2	-3.2	11.5
GPT-6-Sol	**73.6**	**88.2**	**85.2**	**90.1**	3.1	**2.8**	14.6	5.6
Claude Sonnet 5	60.9	68.1	67.6	74.6	15.4	9.8	7.1	11.2
DeepSeek Flash	63.9	79.8	74.9	82.9	**1.7**	3.8	15.9	-26.4
Gemma4-E2B	2.0	21.7	1.9	20.6	43.9	42.3	19.7	50.1
Gemma4-E4B	6.2	29.3	4.8	28.5	46.1	36.8	23.1	**83.9**
Gemma4-12B	23.0	16.0	15.9	31.4	26.9	45.9	-7.0	3.6
MiMo-V2.5	46.9	40.3	54.9	47.6	7.5	8.6	-6.6	-3.3
MiMo-V2.6-Flash	35.5	41.5	49.9	59.0	8.4	6.1	6.1	1.0
MiMo-V2.6-Pro	43.6	45.4	57.6	59.4	7.9	5.1	1.8	5.2
InternVL3.5-4B	1.1	21.4	1.1	20.9	65.3	40.0	20.3	19.2
InternVL3.5-8B	0.8	17.4	0.8	17.3	66.0	50.8	16.7	13.4

erence checks are excluded and tracked separately. Budget exhaustion counts as an unsuccessful solve. Appendix D.1 defines denominators, missing-evidence handling, and numeric tolerances.

5.2 Results

Overall comparison. Table 2 summarizes the comparison between General Solving (GS) and Tool-Augmented Solving (TA) across 13 multimodal model configurations. Access to PolarTools leads to heterogeneous but generally positive effects. Several models obtain substantial accuracy improvements with tool augmentation, including GPT-6-Sol (+14.6 percentage points), DeepSeek Flash (+15.9 points), and Gemma4-E4B (+23.1 points), indicating that domain-specific tools can provide useful capabilities beyond general-purpose code generation. However, the improvement is not universal: some models show limited gains or slight decreases, suggesting that effective tool use depends on the model's ability to select appropriate operations, specify valid arguments, and interpret intermediate outputs.

The results also reveal that accuracy improvement and execution efficiency are not always aligned. Although TA improves task completion for several models, the paired time-saving metric varies across configurations. This indicates that tool augmentation introduces additional interaction steps and execution overhead, while the benefit comes primarily from improving the reliability of scientific workflows rather than consistently reducing runtime.

Task-family analysis. Table 1 analyzes tool effects across different scientific objectives. The largest improvements are observed in temporal change and motion analysis (+19.7 percentage points) and preprocessing and cartography (+15.7 percentage points). These tasks typically require sensor-specific processing, coordinate handling, temporal alignment, or product interpretation, where specialized tools directly provide reusable scientific operations.

Quantitative product diagnostics also benefit from tool augmentation (+6.3 percentage points), demonstrating the value of explicit processing pipelines for extracting physical quantities from remote sensing products. In contrast, perception and evidence interpretation shows a negative pooled difference (-5.3 percentage points). This suggests that tool availability alone does not resolve visual recognition difficulties and that model-backed perception remains an important bottleneck.

Execution behavior and intermediate steps. Beyond final accuracy, PolarTools records execution traces and scientific checkpoints to analyze intermediate behavior. Critical-Step evaluation shows that successful completion requires more than producing a plausible final answer: models

must correctly perform operations such as product decoding, valid-pixel handling, coordinate transformation, and quantitative analysis.

The execution traces demonstrate that tool augmentation changes the problem from direct generation toward scientific workflow construction. Successful TA cases often involve selecting appropriate tools and combining multiple operations, whereas failures frequently arise from incorrect parameter choices, incomplete reasoning chains, or inability to interpret tool outputs.

5.3 Discussion

When do domain-specific tools help? Our results show that domain-specific tools provide the largest benefits for tasks requiring explicit scientific operations, including temporal analysis, preprocessing, and product diagnostics. These tasks depend on sensor-specific processing, coordinate handling, and quantitative computation, where reusable tools reduce implementation errors. However, tool augmentation does not universally improve performance. Perception-heavy tasks remain challenging, indicating that tool availability alone cannot replace robust visual understanding and appropriate tool selection.

Accuracy and efficiency trade-offs. Tool augmentation improves task completion for several models, but accuracy gains do not always translate into lower execution time. Additional tool interactions may introduce overhead, as observed in DeepSeek Flash, where TA improves success rates but increases median solving time on jointly correct tasks. Therefore, tools should be considered primarily as mechanisms for improving scientific reliability rather than guaranteed acceleration.

Moreover, tool-augmented performance is not determined solely by model scale. As shown in Appendix 8, models with larger active or effective parameter counts do not consistently achieve higher TA accuracy, indicating that effective tool use also depends on reasoning, tool selection, and workflow execution capabilities.

Towards scientific agent evaluation. PolarTools demonstrates that evaluating scientific agents requires analyzing both final outputs and intermediate execution behavior. Errors may originate from incorrect preprocessing, parameter selection, or result interpretation, even when the final response appears plausible. Future work should further expand process-level evaluation, investigate reusable tool discovery, and study long-term adaptation of scientific agents.

6 Conclusion

We introduce PolarTools, a benchmark for evaluating tool-augmented multimodal agents in polar remote sensing analysis. Unlike existing benchmarks that focus primarily on individual perception tasks, PolarTools evaluates whether agents can transform scientific questions into executable workflows involving perception, preprocessing, measurement, temporal analysis, and visualization.

Using 710 question-answer instances and a unified environment containing 113 domain-specific tools, we compare multimodal agents under General Solving and Tool-Augmented Solving settings. Our results show that domain-specific tools can substantially improve task completion for scientifically constrained workflows, particularly temporal analysis, preprocessing, and quantitative product analysis. However, the benefits are task- and model-dependent: tool access alone does not eliminate perception challenges or guarantee more efficient execution.

These findings demonstrate that the future of scientific multimodal agents depends not only on stronger foundation models, but also on effective integration between models and domain-specific computational tools. PolarTools provides a platform for studying this interaction and for developing more reliable autonomous systems for complex Earth observation workflows.

AI Use Statement

We used generative AI tools to assist with manuscript drafting and language polishing, literature summarization, and the presentation and organization of figures and tables. Generative AI was also used to create a small number of decorative graphical icons in the manuscript figures; these icons are

purely illustrative and do not encode experimental data or scientific results. All AI-assisted content was independently reviewed, verified, and revised by the authors. The authors take full responsibility for the final text, claims, figures, and other contents of this work.

ETHICS STATEMENT

PolarTools is developed for research on multimodal agents and polar remote-sensing analysis. The benchmark uses publicly available remote-sensing datasets and products together with project-owned UAV imagery. We respect the licenses, attribution requirements, and usage conditions of the original data sources. The benchmark is intended for research and evaluation rather than direct operational or safety-critical decision making. Model-generated outputs may contain errors and should not be treated as authoritative scientific conclusions without appropriate expert verification.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the data sources, acquisition and preprocessing procedures, benchmark construction, task taxonomy, tool interfaces, evaluation protocols, and experimental settings in the main paper and appendix. The anonymous code repository contains the implementation of PolarTools, evaluation scripts, and execution-environment specifications, while the anonymous data repository provides the benchmark questions, annotations, and associated data required for evaluation. These resources are provided to support reproduction of the benchmark construction and reported experiments.

REFERENCES

Samira Alkaee Taleghan, Andrew P Barrett, Walter N Meier, and Farnoush Banaei-Kashani. Icebench: a benchmark for deep-learning-based sea-ice type classification. *Remote Sensing*, 17 (9):1646, 2025a.

Samira Alkaee Taleghan, Morteza Karimzadeh, Andrew P Barrett, Walter N Meier, and Farnoush Banaei-Kashani. Ice-fmbench a foundation model benchmark for sea ice type segmentation. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Polar Data Science*, pp. 1–10, 2025b.

Frank Amdal and Nick Hughes. Multi-sensor ice analysis data: Analysis for belgica bank, north east greenland 2019-20, September 2022. URL https://doi.org/10.5281/zenodo.7053975.

Anne Braakmann-Folgmann, A. Shepherd, D. Hogg, and E. Redmond. Mapping the extent of giant antarctic icebergs with deep learning. *The Cryosphere*, 17:4675–4690, 2023. doi: 10.5194/tc-17-4675-2023.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018. URL https://arxiv.org/abs/1802.02611.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Muhammad 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)*, pp. 7132–7142, October 2025.

Peilin Feng, Zhutao Lv, Junyan Ye, Xiaolei Wang, Xinjie Huo, Jinhua Yu, Wanghan Xu, Wenlong Zhang, Lei Bai, Conghui He, and Weijia Li. Earth-agent: Unlocking the full landscape of earth observation with agents, 2026. URL https://arxiv.org/abs/2509.23141.

Florence Fetterer, Ken Knowles, W. N. Meier, Matt Savoie, Ann Windnagel, and Trey Stafford. Sea ice index, version 4, 2025. URL https://nsidc.org/data/G02135/versions/4.

Henrik Fisser. IcebergSegmentation: released training and analysis code, 2026. URL https://github.com/hfisser/IcebergSegmentation. Accessed 23 September 2026.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

Noel Gorelick, Matt Hancher, Mike Dixon, Simon Ilyushchenko, David Thau, and Rebecca Moore. Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote sensing of Environment*, 202:18–27, 2017.

Nick Hughes and Frank Amdal. Extremeearth polar use case training data, April 2021. URL https://doi.org/10.5281/zenodo.4683174.

K. Jiao, A. D. Fraser, J. Lohse, P. Wongpan, C. Adams, and A. C. Bradley. Grounded icebergs around antarctica: a high-resolution dataset derived from deep learning and sentinel-1 synthetic aperture radar. *Earth System Science Data*, 18(8):6017–6041, 2026. doi: 10.5194/essd-18-6017-2026. URL https://essd.copernicus.org/articles/18/6017/2026/.

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 (CVPR)*, pp. 27831–27840, June 2024.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9579–9589. IEEE, 2024.

Bo Li, Xinru Liu, and Xi Zhao. Data and code for: A dual-branch network for joint monitoring of melt-period sea ice evolution in arctic marginal ice zone, May 2026. URL https://doi.org/10.5281/zenodo.20233826.

Xiang Li, Jian Ding, and Mohamed Elhoseiny. Vrsbench: A versatile vision-language benchmark dataset for remote sensing image understanding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 3229–3242. Curran Associates, Inc., 2024a. doi: 10.52202/079017-0106. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/05b7f821234f66b78f99e7803fffa78a-Paper-Datasets_and_Benchmarks_Track.pdf.

Xiang Li, Jian Ding, and Mohamed Elhoseiny. Vrsbench: A versatile vision-language benchmark dataset for remote sensing image understanding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 3229–3242. Curran Associates, Inc., 2024b. doi: 10.52202/079017-0106. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/05b7f821234f66b78f99e7803fffa78a-Paper-Datasets_and_Benchmarks_Track.pdf.

Danxu Liu et al. SARMAE: Masked autoencoder for SAR representation learning. *arXiv preprint arXiv:2512.16635*, 2025. URL https://arxiv.org/abs/2512.16635.

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.

Zhutao Lv, Chenhao Dang, Yi Feng, Yanpei Gong, Xiaolei Wang, Junyan Ye, Conghui He, and Weijia Li. Earth-agent-pro: Towards real-world full-chain earth observation with agents, 2026. URL https://arxiv.org/abs/2609.12533.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Argho Sarkar and Maryam Rahnemoonfar. Polar-vqa: Visual question answering on remote sensed ice sheet imagery from polar region. *arXiv preprint arXiv:2303.07403*, 2023.

Akashah Shabbir, Muhammad Akhtar Munir, Akshay Dudhane, Muhammad Umer Sheikh, Muhammad Haris Khan, Paolo Fraccaro, Juan Bernabe Moreno, Fahad Shahbaz Khan, and Salman Khan. Thinkgeo: Evaluating tool-augmented agents for remote sensing tasks, 2026a. URL https://arxiv.org/abs/2505.23752.

Akashah Shabbir, Muhammad Umer Sheikh, Muhammad Akhtar Munir, Hiyam Debary, Mustansar Fiaz, Muhammad Zaigham Zaheer, Paolo Fraccaro, Fahad Shahbaz Khan, Muhammad Haris Khan, Xiao Xiang Zhu, and Salman Khan. Openearthagent: A unified framework for tool-augmented geospatial agents, 2026b. URL https://arxiv.org/abs/2602.17665.

A. Stokholm, J. Buus-Hinkler, T. Wulf, A. Korosov, R. Saldo, L. T. Pedersen, D. Arthurs, I. Dragan, I. Modica, J. Pedro, A. Debien, X. Chen, M. Patel, F. J. P. Cantu, J. N. Turnes, J. Park, L. Xu, K. A. Scott, D. A. Clausi, Y. Fang, M. Jiang, S. Taleghanidoozdoozan, N. C. Brubacher, A. Soleymani, Z. Gousseau, M. Smaczny, P. Kowalski, J. Komorowski, D. Rijlaarsdam, J. N. van Rijn, J. Jakobsen, M. S. J. Rogers, N. Hughes, T. Zagon, R. Solberg, N. Longépé, and M. B. Kreiner. The autoice challengeicebench. *The Cryosphere*, 18(8):3471–3494, 2024. doi: 10.5194/tc-18-3471-2024. URL https://tc.copernicus.org/articles/18/3471/2024/.

Fengxiang Wang, Hongzhen Wang, Zonghao Guo, Di Wang, Yulin Wang, Mingshuo Chen, Qiang Ma, Long Lan, Wenjing Yang, Jing Zhang, Zhiyuan Liu, and Maosong Sun. Xlrs-bench: Could your multimodal llms understand extremely large ultra-high-resolution remote sensing imagery? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14325–14336, June 2025.

Jize Wang, Zerun Ma, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. Gta: A benchmark for general tool agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 75749–75790. Curran Associates, Inc., 2024. doi: 10.52202/079017-2412. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/8a75ee6d4b2eb0b777f549a32a5a5c28-Paper-Datasets_and_Benchmarks_Track.pdf.

Junjue Wang, Zhuo Zheng, Zihang Chen, Ailong Ma, and Yanfei Zhong. Earthvqa: Towards queryable earth via relational reasoning-based remote sensing visual question answering, 2023. URL https://arxiv.org/abs/2312.12222.

Aoran Xiao, Shihao Cheng, Yonghao Xu, Yexian Ren, Hongruixuan Chen, and Naoto Yokoya. Geommbench and geommagent: Toward expert-level multimodal intelligence in geoscience and remote sensing, 2026. URL https://arxiv.org/abs/2604.08896.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34, 2021. URL https://arxiv.org/abs/2105.15203.

Xinran Yang, Yibin Ren, Youhao Zhao, and Xiaofeng Li. Pan-arctic winter sea-ice lead mapping at 80-m resolution from Sentinel-1 SAR using LeadNet: product generation and morphological insights. *International Journal of Digital Earth*, 19(2):2728309, 2026. doi: 10.1080/17538947.2026.2728309.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Zi Ye, Yibin Wen, Xiaoya Fan, Xinyu Zhang, Jing Wu, Kun Zeng, Zurong Mai, Jiarui Zhang, Bohan Shi, Juepeng Zheng, Jianxi Huang, Yutong Lu, and Haohuan Fu. Agrotools: A benchmark for tool-augmented multimodal agents in agriculture, 2026. URL https://arxiv.org/abs/2605.22366.

Appendix

Contents

A OVERVIEW OF THE APPENDIX **15**

B Data Acquisition and Preprocessing **15**

B.1 Sources and current-use accounting 15

B.2 Preprocessing and segmentation sample splits 16

C Tools Construction and Implementation **18**

C.1 Overview of Tools 18

C.2 BergSeg: Iceberg Segmentation Tool 18

C.2.1 Data Sources and Annotation 18

C.2.2 Model Design 20

C.2.3 Experimental Setup 20

C.2.4 Results and Analysis 21

C.2.5 Inference and Tool Outputs 23

C.3 Sea-ice and Oil-spill Segmentation Tools 26

C.4 Tools List 27

C.4.1 Data Access and Preprocessing Tools 28

C.4.2 Object Perception and Region Extraction Tools 30

C.4.3 Measurement, Statistics, and Spatial Analysis Tools 31

C.4.4 Temporal Change and Motion Analysis Tools 32

C.4.5 Visualization and Result Export Tools 34

D Experimental Details **34**

D.1 Details of Evaluation Metric 34

D.2 Evaluation Models and Scale Details 35

E Case Study **36**

F Additional Analysis of Model Scale **42**

Table 3: Current QA questions and image/raster uses by task type. An image used by two questions contributes two uses; saved crops are counted individually. Label-only raster uses are separated from source imagery.

Primary task type	Questions	Image uses	Label-only uses
Perception and evidence interpretation	93	81	0
Object and spatial structure analysis	144	42	110
Temporal change and motion	176	548	0
Geometric measurement and counting	65	61	20
Quantitative product diagnostics	109	351	0
Preprocessing and cartography	123	147	0
Total	**710**	**1,230**	**130**

A OVERVIEW OF THE APPENDIX

The appendix documents the data, tool implementations, scoring protocol, and recorded examples behind the main results. Its counts distinguish source observations, persisted model samples, question–image uses, questions, and model–question runs; these units must not be interchanged.

Section B describes the provenance and processing of public products and locally collected observations. It reports the current 710-question QA inventory, image uses, label-only inputs, and the separate segmentation sample splits. Section C describes the tool catalog, its implementation status, and the iceberg, sea-ice, oil-slick, and ice-lead perception backends. Its catalog tables list tool inputs, outputs, and backend requirements.

Section D specifies the five metrics, their denominators, tolerances, missing-evidence rules, evaluation models, and the coverage behind the provisional results. In particular, key-step labels are defined for all 710 questions, while independently verified stage scores currently cover only a small pilot; missing stage evidence is not treated as success or failure. Finally, Section E presents ten version-matched task records with input descriptions, observed outcomes, and abridged execution traces from multiple models and modes.

B DATA ACQUISITION AND PREPROCESSING

This section follows a source–integration–preprocessing order: we first identify the public datasets, separately acquired satellite products, and owned UAV observations; then state how these inputs enter the current QA collection or the segmentation experiments. This distinction matters because an acquisition, an image file, a cropped sample, and a question are different units. A *model sample* is one persisted spatial image window or gridded product in the segmentation manifest. A large image split into k stored windows contributes k samples. Paired polarizations, optical bands, validity layers, and a mask for the same window remain components of one sample. For QA we count each question–image use: if two questions reference the same persisted image, it contributes two uses. We do not collapse cross-question copies by hash. Training-time random transformations without a persisted manifest are not enumerated. The inventory is the current local QA revision and the current segmentation-data versions, not the union of historical experiments.

B.1 SOURCES AND CURRENT-USE ACCOUNTING

After grouping paired bands and removing preview, validity, and area-support layers within each question, the current QA set contains 1,230 image or gridded-product *uses*, classified by the six current task types in Table 3. A further 130 label-only raster uses support mask-focused questions; they are reported separately because they are annotations rather than source imagery. The current segmentation manifests contain 9,865 image samples: 8,101 training, 977 validation, and 787 reserved test samples. Thus the two branches account for 11,095 *branch-level image uses or samples* (1,230+9,865). This is an additive accounting, not a claim that 11,095 independent source scenes were acquired: the same public collection may support both branches. The QA count is also not a question count; one question can require several dates or scenes.

Temporal change and motion accounts for 548 image uses across 176 questions because many tasks reference multiple dates or reuse the same grids across different questions. These are question–image links, not independent acquisitions. The 710-question taxonomy also includes tasks with tabular, vector, text, or other non-image inputs; image-use counts therefore do not sum to 710.

Public datasets and products. The iceberg branch uses released SARIB Sentinel-1 image–mask pairs (Jiao et al., 2026) and Belgica Bank and ExtremeEarth Sentinel-2 polygon labels (Amdal & Hughes, 2022; Hughes & Amdal, 2021). The sea-ice branch uses the Kara Sea DSI-Net dual-label release (Li et al., 2026): 1,206 512×512 HH/HV quicklook pairs from 71 Sentinel-1 source products, with author ice–water semantic labels assisted by SAM and manually corrected. The QA collection additionally uses the NSIDC Sea Ice Index G02135 v4 (Fetterer et al., 2025), OSI SAF sea-ice concentration, edge, and drift grids, and publicly released iceberg tracking records. For leads, it uses MOSAiC binary lead maps (PANGAEA 962904) and accumulated lead fractions (PANGAEA 963671), together with selected corresponding Sentinel-1 HH observations. The oil-slick source is the public Sentinel-1 image–mask release (Zenodo 8346860). Published masks and polygons retain their public provenance even where we downloaded the underlying satellite bands and rasterized or reviewed labels locally. The BYU consolidated iceberg-track records support 90 QA questions but consist primarily of trajectory and ancillary files; they therefore do not add 90 image uses to Table 3. The mixed-source questions combine otherwise listed NSIDC/OSI sea-ice grids or MOSAiC SAR windows, rather than constituting a new public dataset. This explains why the number of source families, QA questions, and image units differ.

Separately downloaded satellite imagery. The acquisition-level inventory in Table 6 lists all 52 products in the project scene-download collections: 21 Sentinel-1, 20 Sentinel-2, six ALOS-2/PALSAR-2, four Landsat, and one Capella product, including the dates, channels, pixel grids, and experimental status of each collection. Within the *current* iceberg experiments, one Greenland Sentinel-1 EW product yields 46 locally labelled SAR inputs, three Antarctic IW products yield 72, 13 Belgica Bank Sentinel-2 products yield 3,472 optical inputs, and three ExtremeEarth products yield 65. The other downloaded scene products are retained for provenance or inspection but are excluded from current training counts. QA-specific Fram/Weddell, iceberg-tracking, preprocessing, and Kara Sea Sentinel-2 windows are accounted for as QA inputs in Table 3; their repeated dates and preview files are not inferred to be independent acquisitions. For the oil-slick dataset, 1,178 source TIFF–mask pairs trace to 270 Sentinel-1 product identifiers; only the 1,030 training and 77 validation source images generated the fixed crops in the current model manifest. The remaining 66 source images are reserved for testing and five for a latitude buffer. Neither source-image nor product counts substitute for the number of persisted model samples.

Owned UAV imagery. Ten project-owned orthomosaics were acquired over seven flight dates in August 2025, at approximately 0.0975–0.1000 m pixel spacing. Their image identifiers are listed in Table 4. The first QA group uses the ten originals; later QA construction records four planned tiles per original, with one tile in each group further exported as four fragments. The resulting current QA image files comprise 30 retained larger tiles and 40 fragments, or 70 final crop images. Along with the ten orthomosaics, this gives 80 UAV image-use records. Six QA records also use UAV-derived labels, while no owned UAV image is in a current segmentation training split. This separates data ownership from annotation origin and from model-training use.

B.2 Preprocessing and segmentation sample splits

Table 5 is the manifest-level accounting for the four current segmentation datasets. SARIB supplies 576 iceberg SAR samples, including all 100 retained SAR test inputs; locally reviewed Greenland and Antarctic Sentinel-1 imagery supplies another 118 training samples. Belgica Bank supplies 3,472 optical samples and ExtremeEarth 65. The DSI-Net release supplies 1,206 sea-ice samples. The oil-slick release supplies 4,428 fixed crops. Of the 8,101 training samples, 7,983 have released public labels, 118 combine publicly obtainable imagery with project-local labels, and zero come from the owned UAV imagery. Validation and held-out test samples are shown separately and were not optimized as training inputs.

For iceberg SAR, each retained sample stores calibrated backscatter, an iceberg/background/unknown mask, and a valid-data mask; the working input is predominantly

Table 4: The ten owned UAV orthomosaics. Each contributed four planned tile positions; the current QA inputs comprise 70 final crop/fragment files plus these ten originals. Filenames are retained to make each acquisition traceable.

Date (2025)	Orthomosaic file	Approx. pixel size
Aug 13	01_20250813-M4E.tif	0.1000 m
Aug 17	01_20250817_CTD_ICE01_M4E.tif	0.0977 m
Aug 17	01_20250817-ICE01-Test01-M4E.tif	0.0977 m
Aug 18	01_20250818-ICE02-ew-M4E.tif	0.0976 m
Aug 18	01_20250818-ICE02-ns-M4E.tif	0.0976 m
Aug 21	01_20250821-ICE03-M4E.tif	0.0975 m
Aug 22	01_20250822-ICE04-ew-M4E.tif	0.0975 m
Aug 22	01_20250822-ICE04-ns-M4E.tif	0.0975 m
Aug 24	01_20250824-ICE05-M4E.tif	0.0975 m
Aug 26	01_20250826-ICE06-M4E.tif	0.0977 m

Table 5: Current segmentation image samples, one row per saved spatial crop/record rather than per parent scene. Oil test/buffer source TIFFs have no fixed crops in this manifest. The lead model has no eligible supervised training set.

Target	Source	Train	Val.	Test	Total
Iceberg SAR	SARIB public masks	395	81	100	576
Iceberg SAR	Greenland local labels	46	0	0	46
Iceberg SAR	Antarctic IW local labels	72	0	0	72
Iceberg optical	Belgica Bank public polygons	2,512	444	516	3,472
Iceberg optical	ExtremeEarth public labels	65	0	0	65
Sea ice	DSI-Net public dual labels	891	144	171	1,206
Oil slick	Zenodo source masks, fixed crops	4,120	308	0	4,428
Ice leads	Eligible supervised pairs	0	0	0	0
Total		**8,101**	**977**	**787**	**9,865**

256×256 pixels, with 15 retained SAR test records at 288×256. The 46 Greenland entries are independently processed single-polarization inputs from 23 source windows, consistent with the single-channel model interface. For Sentinel-2 iceberg segmentation, B08, B04, and B02 are aligned with rasterized public polygons on 256×256 samples. The 13 Belgica source products are Level-2A image acquisitions matched to Level-1C identifiers in the source annotations. Both identifier forms are preserved. Unknown or invalid pixels are excluded from supervised loss. Parent-window and scene grouping determine splits; the optical split separates geographic bands but is not fully independent by Sentinel-2 acquisition. The archived Belgica label export still awaits reconciliation with later review decisions.

For sea ice, a 512×512 DSI-Net HH/HV pair and its ice–water mask form one stored sample. The inputs are uint8 quicklooks without calibrated backscatter or sufficient georeferencing to infer physical area directly. The current 891/144/171 partition is a development split whose formal spatial assignment is not yet frozen. Random 256×256 crops and augmentations during training were not saved as separate images and are therefore excluded from the inventory. For oil slicks, fixed 256×256 crops were generated from the 1,030 training and 77 validation TIFFs, four per source image. The patch manifest retains the source product, row/column offset, mask, container file, and array index for all 4,428 crops. The data are nonpolar development imagery; no polar or event-independent oil-slick result is implied by this split.

Ice-lead material and admission status. The local MOSAiC source manifest indexes 1,596 downloaded binary lead-map TIFFs from PANGAEA 962904. QA uses selected SAR windows, maps, and two accumulated-fraction products; their uses are included in the task-type accounting above. Separately, seven monthly archives of the public LeadNet release (Zenodo 17718323) index 14,093 automatic prediction TIFFs. Seven source-SAR audit cutouts were saved with diagnostic previews and overlays, but the automatic products lack verified dense manual masks paired to the relevant original SAR scenes. Their number is a source-product inventory, not a training sample

count. We do not train an ice-lead model from these archives. Instead, `LeadSegment` calls the authors' released LeadNet checkpoint under its five-band input contract (Yang et al., 2026); see Section **??**.

The per-image QA ledger records representative path, content digest, domain, and linked question IDs. Model manifests retain sample ID, source scene, split, input container, label provenance, and—for fixed oil crops—pixel offsets and array index. These records support the sample-level counts above while keeping acquisition counts and multiple uses of the same source image distinct.

C Tools Construction and Implementation

C.1 Overview of Tools

Tables 13–17 in Section C.4 present the six categories of PolarTools, their interface counts, and the functions, inputs, outputs, and implementation status of individual tools. The tool system covers the entire process from raw observation to result delivery in polar remote sensing analysis: ***Data acquisition and preprocessing*** is responsible for reading, checking, and standardizing the input from different sensors; ***Target perception and region extraction*** generates land cover masks, boundaries, and object candidates; ***Measurement statistics and spatial analysis*** calculates area, distance, quantity, and spatial relationships; ***Time series change and motion analysis*** processes cross-temporal matching, trajectories, and changes; ***Visualization and result export*** converts intermediate evidence and analysis results into checkable maps and structured files. The five types of tools can be combined and called according to task requirements, forming a traceable processing chain for image processing, target extraction, quantitative analysis, and result presentation.

C.2 BergSeg: Iceberg Segmentation Tool

BergSeg provides sensor-specific iceberg segmentation from satellite imagery. Its SAR branch, **BergSeg-SAR**, processes a single calibrated backscatter channel; its optical branch, **BergSeg-Opt**, processes Sentinel-2 near-infrared, red, and blue bands. Each branch predicts iceberg probabilities and a binary segmentation mask while retaining invalid-pixel information. The branches operate independently. BergSeg names the application-specific model suite and inference workflow built on established segmentation architectures. Below, we describe its data, implementation, archived experiments, and the distinction between experimental inputs and the newly assembled PALSAR-2 annotation supplement.

C.2.1 Data Sources and Annotation

We assembled imagery from Sentinel-1, Sentinel-2, and ALOS-2/PALSAR-2, with three annotation sources (Table 7). *Public masks* are released image–mask pairs from the grounded-iceberg collection of Jiao et al. (2026), called SARIB in our archive, retained after local screening. *Public polygons* are the Belgica Bank (Amdal & Hughes, 2022) and ExtremeEarth (Hughes & Amdal, 2021) annotations, matched to Sentinel-2 imagery, rasterized, and screened locally. *Local labels* are the Greenland and Antarctic Sentinel-1 collections, for which image-derived candidates and review decisions were recorded. The supplementary PALSAR-2 windows contain retained annotation material whose boundaries remain under review; they are not finalized training labels.

Downloaded imagery and provenance. Table 6 separately inventories 52 unique products in the scene-download collections: 21 Sentinel-1, 20 Sentinel-2, six ALOS-2/PALSAR-2, four Landsat, and one Capella product. These are product entries, not independent overpasses or model-input counts; selected bands and spatial windows may be stored instead of complete products. The scope excludes separately released SARIB pairs, SAR Grid and wave-mode collections, withheld Mendeley data, and auxiliary thematic datasets. Only one Greenland EW product and three Antarctic IW products contribute the current locally labelled SAR inputs. The optical inputs derive from 13 Belgica Bank and three ExtremeEarth products. The four supplementary Sentinel-2 products, Landsat and Capella imagery, and the retained PALSAR-2 windows are not included in the reported experiments. In particular, Belgica Bank annotation identifiers refer to Level-1C products, whereas the matched images actually downloaded are Level-2A products from the corresponding acquisitions. Both identifiers are retained in the provenance record. Reported spacings describe the pixel grid,

Table 6: Acquisition-level inventory of downloaded imagery. Counts are unique products within the listed collections, not training tiles or independent satellite passes. Selected bands/windows may be stored instead of complete products. Public SARIB pairs and other separately released datasets are excluded from this download inventory. The reported experimental inputs remain those in Table 7.

Collection / product	No.	Acquisition dates	Channels, pixel spacing, and experimental use
S1 Antarctic EW GRDM	6	2020-10-13 to 2020-10-17	HH/HV; 40 m; archived candidates only.
S1 additional EW GRDM	1	2020-10-24	HH/HV; 40 m; archived candidates only.
S1 ExtremeEarth EW	2	2018-03-12 to 2018-03-13	HH/HV; 40 m; not used in current SAR experiments.
S1 Greenland EW	3	2017-07-30 to 2021-08-23	HH/HV; 40 m; one product contributes 46 training inputs.
S1 polar IW GRDH	8	2017-07-25 to 2021-08-20	HH/HV or HH; native 10 m spacing; three Antarctic products yield 72 training inputs on a 20 m grid.
S1 A23a/A23b EW	1	2025-01-30	HH; 40 m; supplementary inspection only.
S2 Belgica Bank L2A	13	2019-05-02 to 2020-05-02	B02/B03/B04/B08 at 10 m; SCL at 20 m. Matched to L1C-indexed public polygons; 3,472 experimental inputs.
S2 ExtremeEarth L2A	3	2018-03-13	Multiscale band windows at 10 m; 65 training inputs use B08/B04/B02.
S2 supplementary L2A	4	2024-11-20 to 2026-08-04	Six spectral bands plus SCL; 10/20 m; four 60-km windows, unlabelled.
PALSAR-2 ScanSAR L2.2	6	2022-02-19 to 2025-02-15	HH/HV; 25 m; 13 retained review windows from three products; no current experimental inputs.
Landsat 8/9 OLI L2	4	2025-01-17 to 2026-08-02	Six spectral bands plus QA; 30 m; four 60-km windows, unlabelled.
Capella GEC	1	2021-05-14	HH; 0.6 m pixel spacing; exploration only, iceberg labels unavailable.
Total products	**52**		S1: 21; S2: 20; PALSAR-2: 6; Landsat: 4; Capella: 1.

not necessarily the sensor's resolving power. The accompanying acquisition inventory lists every product identifier, UTC acquisition time, region, channels, local files, and experimental use.

A model-input sample is one image and its corresponding mask, rather than an independent satellite acquisition. Optical inputs and SAR training/validation inputs are 256×256 pixels. The SAR test set retains 85 inputs at 256×256 and 15 at 288×256 (height $\times$ width). Masks encode iceberg, known background, and unknown pixels as 1, 0, and 255. Unknown or invalid pixels are excluded from both supervised loss and evaluation; unreviewed surroundings are not automatically assigned to background. Counts exclude training-time augmentation. The 46 Greenland inputs represent separate HH and HV channels from 23 windows.

SAR splits use disjoint scene-group identifiers; all current local Sentinel-1 samples are training inputs, and validation/test inputs come from SARIB. Optical splits use separated latitude bands within Belgica Bank, with every child tile of a parent window kept in one split; ExtremeEarth contributes only to training. This is a spatial partition, not an acquisition-independent partition: different portions of the same Sentinel-2 acquisition can occur in different splits. The optical counts and experiments refer to an archived label snapshot. Later Belgica Bank review decisions have not all been reconciled with its label export, so this snapshot is not a finalized annotation release.

The PALSAR-2 supplement comprises five Weddell Sea windows acquired on 19 February 2022 (A22_full, B01, L02, L07, L08), six A23a/A23b-area windows acquired on 5 February 2025 (A25_11, A25_12, A25_13, A25_21, A25_33, A25_42), and two Ross Sea windows acquired on 15 February 2025 (W38, W40). It includes one 4096×4096 overview and twelve 1024×1024 windows at 25 m pixel spacing, with overlapping footprints and paired HH/HV inspection. User-selected target points and intensity-based candidate extraction support subsequent boundary review. None of these 13 windows has been admitted to the current training, validation, or quantitative test sets, and

Table 7: Data sources and experimental splits. Upper-block counts are image–mask inputs. The 13 PALSAR-2 windows are supplementary review material and are excluded from the SAR and optical subtotals; no experimental split has been assigned to them.

Annotation	Collection	Sensor	Train	Val.	Test	Total
Public masks	SARIB	Sentinel-1	395	81	100	576
Local labels	Greenland	Sentinel-1	46	0	0	46
Local labels	Antarctic IW	Sentinel-1	72	0	0	72
	SAR subtotal		**513**	**81**	**100**	**694**
Public polygons	Belgica Bank	Sentinel-2	2,512	444	516	3,472
Public polygons	ExtremeEarth	Sentinel-2	65	0	0	65
	Optical subtotal		**2,577**	**444**	**516**	**3,537**
Local candidates	Review supplement	PALSAR-2	–	–	–	13

no retraining was performed after this addition. They must not be confused with the 88 historical PALSAR-2 inputs in the earlier 718-input SAR archive (388/212/118 training/validation/test), which included 48 PALSAR-2 training and 40 validation inputs. The current 20-epoch SAR experiment uses the later 513/81/100 split; removal of eight additional diagnostic test entries from its original archive did not alter training, validation, or the retained main-test samples.

C.2.2 Model Design

BergSeg-SAR uses U-Net with a ResNet-34 encoder and one input channel. Each polarization is processed independently. ImageNet encoder weights initialize the model; the three first-layer channel kernels are summed for the single-channel input. Calibrated backscatter is clipped to $[-35, 5]$ dB and mapped linearly to $[0, 1]$. A single-channel output head produces an iceberg logit at each pixel. The reported SAR experiments use this preprocessing on calibrated inputs; PALSAR-2 display intensities are not interchangeable with calibrated backscatter.

BergSeg-Opt extends SegFormer with a MiT-B2 encoder (Xie et al., 2021). Inputs use B08/B04/B02 in that order, not natural-color RGB. Reflectance in $[0, 1]$ is normalized by channel means $(0.485, 0.456, 0.406)$ and standard deviations $(0.229, 0.224, 0.225)$, as in the archived implementation. The ImageNet-initialized encoder and SegFormer decoder provide multiscale semantic features. A *detail branch* applies two 3×3 convolutions with group normalization and GELU, the first with stride two, to obtain 32-channel features at half input resolution. A 1×1 projection reduces the decoder's 256 channels to 32; the resized semantic features are concatenated with the detail features. Convolutional fusion and a sigmoid gate yield a 32-channel representation. A zero-initialized 1×1 projection predicts a residual logit, resized and added to the baseline logits, so the residual initially leaves the baseline unchanged. An auxiliary *boundary head* predicts one boundary logit from this representation, or from decoder features when the detail branch is absent. The boundary head supplies training supervision; final iceberg masks use the segmentation output. The design analysis evaluates the complete model and variants omitting each of these mechanisms, alongside the SegFormer baseline.

C.2.3 Experimental Setup

Training Configuration. Both branches use masked binary cross-entropy and soft Dice loss. For known pixels V_i, probabilities p_{iu}, and binary labels y_{iu}, the segmentation objective is

$$\mathcal{L}_{\text{seg}} = \text{BCE}_V(p, y) + \frac{1}{|\mathcal{B}_V|} \sum_{i \in \mathcal{B}_V} \left(1 - \frac{2 \sum_{u \in V_i} p_{iu} y_{iu} + 1}{\sum_{u \in V_i} p_{iu} + \sum_{u \in V_i} y_{iu} + 1} \right), \tag{1}$$

where BCE is averaged over known batch pixels and $\mathcal{B}_V$ contains images with at least one known pixel. Boundary variants add $0.1\mathcal{L}_{\text{boundary}}$, an unweighted binary cross-entropy on the local maximum minus minimum of the mask in a 3×3 neighborhood. Neighborhoods intersecting unknown pixels or image borders are excluded.

Table 8: Training configuration for the SAR source control and optical component ablation. Historical SAR baselines instead use their archived 100-epoch protocols. Optical U-Net and CBAM-ResUNet use batch size 8.

Setting	SAR source control	Optical design analysis
Epochs; batch size	20; 12	100; 32
Optimizer; schedule	AdamW; cosine	AdamW; cosine
Learning rate	3×10^{-4}	Encoder: 3×10^{-5}; other: 3×10^{-4}
Weight decay	10^{-4}	0.05
Steps per epoch	43	81
Training / evaluation	FP16 / FP32	BF16 / FP32
Augmentation	D4 and intensity perturbations	D4
Model selection	Validation scene-group IoU	Validation pooled-pixel IoU

Table 8 summarizes the focal protocols. The SAR source control holds the number of optimization steps and sampled inputs fixed. Mixed-source batches contain six SARIB and six local Sentinel-1 inputs, sampled with replacement; the public-only control draws all twelve from SARIB. SAR augmentation combines right-angle rotations/reflections with gain, offset, and gamma perturbations. Optical augmentation samples uniformly from the eight right-angle rotation/reflection transforms (D4). The four optical ablations share data order and sampler seed 4242. Initialization seed is 42, and gradient norms are clipped to one. Checkpoints maximize validation foreground IoU: scene-group macro IoU for the SAR source control and pooled known-pixel IoU for the optical runs. The SAR control and BergSeg-SAR select epochs 8 and 14, respectively; the optical baseline, detail-only, boundary-only, and combined variants select epochs 53, 40, 57, and 55. The four historical SAR comparators use 100-epoch runs on the earlier 388-input training split and their respective historical validation sets; BergSeg-SAR uses 20 epochs and 513 training inputs. These differences are retained when interpreting the common-test comparison.

Evaluation Protocol and Metrics. Within each sensor, all configurations are evaluated on identical test inputs and the same archived reference masks. Inference uses FP32 and a fixed probability threshold of 0.5, without test-time augmentation or morphological postprocessing. Confusion counts are pooled over all known test pixels before computing

$$\mathrm{IoU} = \frac{TP}{TP + FP + FN}, \qquad \mathrm{F1}\ (= \mathrm{Dice}) = \frac{2TP}{2TP + FP + FN}, \tag{2}$$

$$\mathrm{Precision} = \frac{TP}{TP + FP}, \qquad \mathrm{Recall} = \frac{TP}{TP + FN}. \tag{3}$$

Here, TP denotes correctly predicted iceberg pixels, FP background incorrectly predicted as iceberg, and FN missed iceberg pixels. IoU measures foreground overlap; precision measures resistance to false alarms; recall measures iceberg coverage. F1 and binary Dice are identical under this aggregation and are reported once. We report foreground IoU rather than foreground/background mean IoU (mIoU). Overall pixel accuracy is not a primary metric because background dominates these images.

For optical experiments, $\mathrm{BF1}_1$ measures boundary alignment with a one-native-pixel Manhattan-distance tolerance. Inner boundaries use four-connected binary erosion. Predicted and reference boundary matches are accumulated across tiles to obtain boundary precision and recall; their harmonic mean is BF1. Boundaries within two pixels of unknown regions or tile edges are excluded. BF1 differs from region F1: a similar region score can conceal misplaced edges. All table metrics are percentages and higher is better. These are single-seed archived experiments; prior test-set use during development and the unresolved optical label reconciliation preclude claims of fully blind, finalized benchmark performance.

C.2.4 Results and Analysis

Quantitative Comparison. Tables 9 and 10 separate model comparisons (upper block) from the effect of local training data for SAR and design analysis for BergSeg-Opt (lower block). We include

four SAR comparator configurations and five optical comparator configurations. U-Net, U-Net++, and DeepLabV3+ (Chen et al., 2018) use ResNet-34. SARMAE (Liu et al., 2025) uses a ViT-B encoder with SAR-oriented masked-autoencoder pretraining and a segmentation decoder. Optical CBAM-ResUNet adapts the author-released ResNet-152 SAR topology of Jiao et al. (2026) to three input bands.

Table 9: Sentinel-1 performance on the same 100 test inputs. Upper: comparison of the specified archived models, whose training conditions differ (Section C.2.3). Lower: effect of adding local Sentinel-1 training labels under the matched 20-epoch protocol. U-Net (SARIB only) uses public training data; BergSeg-SAR also uses local labels. Scores are percentages; bold and underline mark the highest and second-highest values within each block, respectively, and gray shading identifies our model. The ranking marks do not imply statistical significance.

Model / configuration	IoU	F1	Prec.	Rec.
Model comparison				
U-Net	<u>73.69</u>	<u>84.85</u>	82.48	<u>87.37</u>
U-Net++	67.64	80.69	73.29	**89.77**
DeepLabV3+	73.21	84.53	<u>83.51</u>	85.57
SARMAE	71.53	83.41	82.10	84.75
BergSeg-SAR	**73.87**	**84.97**	**83.97**	86.00
Effect of local training data				
U-Net (SARIB only)	<u>73.52</u>	<u>84.74</u>	<u>81.75</u>	**87.96**
BergSeg-SAR	**73.87**	**84.97**	**83.97**	<u>86.00</u>

SAR results. BergSeg-SAR achieves 73.87% IoU, compared with 73.69% for the archived U-Net comparator. All 100 common test inputs and their acquisitions are absent from the historical training/validation sets. Because training data, epoch budgets, and validation sets differ, this comparison does not isolate an architectural improvement. In the matched lower block, adding local Sentinel-1 data changes IoU from 73.52% to 73.87% (+0.35 percentage points): precision increases from 81.75% to 83.97%, while recall falls from 87.96% to 86.00%. This is a modest precision–recall tradeoff, not evidence of uniform improvement or an effect of the new PALSAR-2 data.

Table 10: Sentinel-2 results on the same 516 test inputs. Upper: model comparison; all runs use 100 epochs, but U-Net and CBAM-ResUNet use batch size 8 and other rows use 32. Lower: BergSeg-Opt design analysis, with all configurations trained at batch size 32 under the matched MiT-B2 protocol. Every row uses its own validation-selected checkpoint. The full model is compared with versions omitting boundary supervision or detail refinement and with the SegFormer baseline. F1 equals Dice; $\mathrm{BF1}_1$ is boundary F1 at one-pixel tolerance. Bold and underline mark the highest and second-highest values within each block, respectively; gray shading identifies our model. Baseline and combined rows are repeated for readability.

Model / configuration	IoU	F1	Prec.	Rec.	$\mathrm{BF1}_1$
Model comparison					
U-Net	33.44	50.12	36.63	79.33	58.19
U-Net++	24.31	39.11	26.38	75.55	57.61
DeepLabV3+	22.75	37.07	24.67	74.53	43.06
SegFormer	<u>56.65</u>	<u>72.33</u>	<u>63.80</u>	<u>83.49</u>	<u>60.34</u>
CBAM-ResUNet	12.69	22.53	13.31	73.30	48.66
BergSeg-Opt	**68.40**	**81.24**	**77.09**	**85.85**	**64.56**
BergSeg-Opt ablation analysis					
BergSeg-Opt (full)	**68.40**	**81.24**	**77.09**	<u>85.85</u>	**64.56**
Without boundary supervision	<u>66.50</u>	<u>79.88</u>	<u>73.56</u>	**87.39**	<u>63.48</u>
Without detail refinement	54.45	70.51	60.22	85.05	60.41
SegFormer baseline	56.65	72.33	63.80	83.49	60.34

Optical results. BergSeg-Opt achieves 68.40% IoU and 81.24% F1, compared with 56.65% and 72.33% for SegFormer. The +11.75-point IoU change is computed from the full test set, not an average of selected visual examples. False-positive pixels decrease from 42,262 to 22,758, and false negatives from 14,727 to 12,620. Boundary F1 increases from 60.34% to 64.56%. Detail-only gives 66.50% IoU, whereas boundary-only gives 54.45%: its 1,393 additional true positives are outweighed by 7,860 additional false positives. Thus, boundary supervision alone is not beneficial on this snapshot; its gain when added to detail is 1.90 points and should not be treated as proof that the component is necessary. An exploratory paired bootstrap over 131 test parent windows (10,000 resamples) gives a 95% interval of $[-0.09, 25.38]$ points for combined-minus-baseline IoU and $[-3.28, 7.90]$ for combined-minus-detail. Spatial dependence and the single training seed further limit interpretation of these intervals.

The optical baselines share preprocessing, augmentation, objective, learning rates, and validation-based checkpoint selection, but the batch-size difference means this is not a fully controlled architecture ranking. In particular, U-Net makes 323 optimizer updates per epoch versus 81 for batch-32 runs. Its validation IoU of 80.94% falls to 33.44% on test; replaying the frozen checkpoint reproduces the archived predictions. Precision falls from 91.64% to 36.63%, and three parent windows account for 55.5% of its test false positives. The known-pixel iceberg fraction also falls from 0.741% in validation to 0.266% in test. These findings identify concentrated false alarms and distribution differences; they do not isolate batch size as the cause or establish that U-Net is generally unsuitable for iceberg segmentation.

Relation to published iceberg methods. U-Net has been used for giant Sentinel-1 icebergs, with a reported F1 of 0.84 across 191 images and a median absolute area deviation of 4.1% (Braakmann-Folgmann et al., 2023). The Antarctic grounded-iceberg framework of Jiao et al. (2026) reports detection F1 above 0.91; that system also includes multi-temporal identification and physical constraints. These original-paper results concern different targets, datasets, aggregation rules, and pipelines and are not ranked against our pixel-level test scores. For optical imagery, the released iceberg code of Fisser (2026) uses a Sentinel-2 U-Net++/ResNet-34 model. Our U-Net++ row is a locally trained binary configuration, not a full reproduction of that recipe. A separate archived replay of the author's three-class recipe on the author's 36 test inputs gives 57.98% foreground IoU and 73.40% Dice at its prescribed readout point (161 completed epochs). That number is our replay result, not a score quoted from the paper, and cannot be compared directly with our 516-input test table. Differences in dependency versions also prevent calling it an exact reproduction.

Qualitative Results. Figures 4 and 5 show selected improvements against four SAR and five optical comparator configurations. Within each case, all models share the input, reference, display stretch, and threshold. Teal denotes correctly identified iceberg pixels (TP), vermilion false alarms (FP), blue missed iceberg pixels (FN), and gray unknown reference pixels excluded from scoring. IoU/Dice printed in the figures refer to the full displayed input; Dice is the F1 reported in the tables.

The examples are deliberately selected after evaluation to illustrate gains. The four Sentinel-1 cases are retained improvement examples with at least 500 reference iceberg pixels, all from one acquisition. For Sentinel-2, S2-1 and S2-3 are retained from a prior ranking by improvement over SegFormer. S2-2 is selected for clearer visual interpretation and higher BergSeg-Opt IoU than every displayed comparator. All three are complete 512×512 test parents with at least 500 reference iceberg pixels. Optical parents are assembled from four independently inferred 256×256 tiles. Shared 128×128 detail crops maximize net error-pixel reduction relative to the selection comparator, with ties resolved by proximity to the corrected-pixel centroid. These selections illustrate favorable cases rather than average performance. The complete-test-set design analysis in Table 10 quantifies component effects, including the boundary-only regression.

C.2.5 Inference and Tool Outputs

The SAR interface accepts calibrated single-channel GeoTIFFs or prepared arrays and returns probabilities, a binary mask (invalid pixels: 255), and metadata recording the checkpoint checksum, units, threshold, dimensions, and pixel counts. Large rasters use overlapping 256×256 tiles (default overlap: 64 pixels), with probabilities averaged in overlaps. GeoTIFF outputs preserve input georeferencing; array inputs do not acquire artificial coordinates.

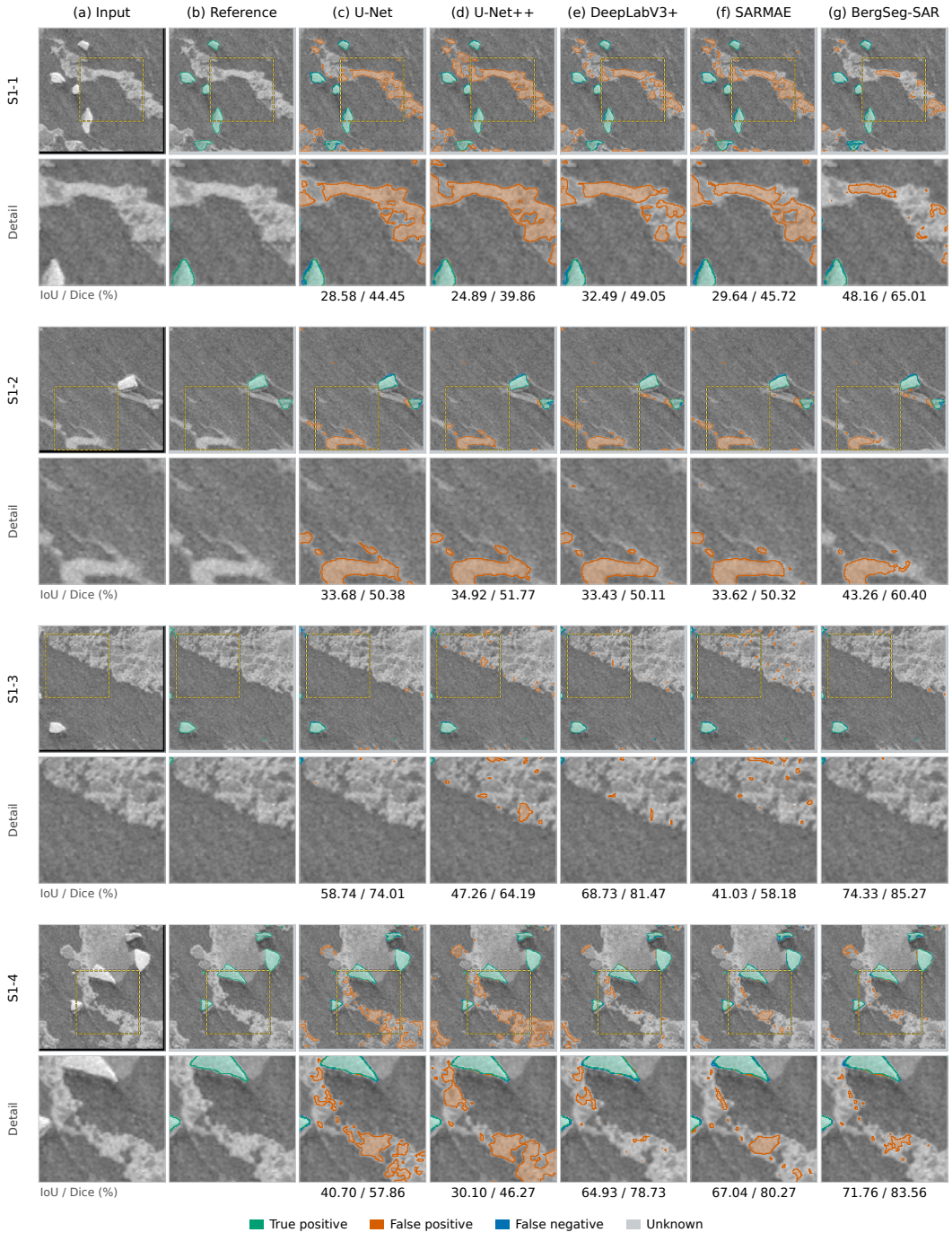


Figure 4: Selected Sentinel-1 improvement examples for BergSeg-SAR. Row pairs show full inputs and enlarged shared detail crops. The four retained cases illustrate favorable IoU differences against the four displayed comparators; all originate from the 3 April 2025 11:06:53 acquisition. Inputs use a fixed $[-35, 0]$ dB display stretch. Colors denote TP (teal), FP (vermilion), FN (blue), and unknown (gray); scores are full-input IoU/Dice (%). These examples and their error-reduction crops are post hoc selections, not representative estimates of average gains. Historical baselines use 100-epoch training; BergSeg-SAR uses 20 epochs and different training data. Full-test scores are in Table 9.

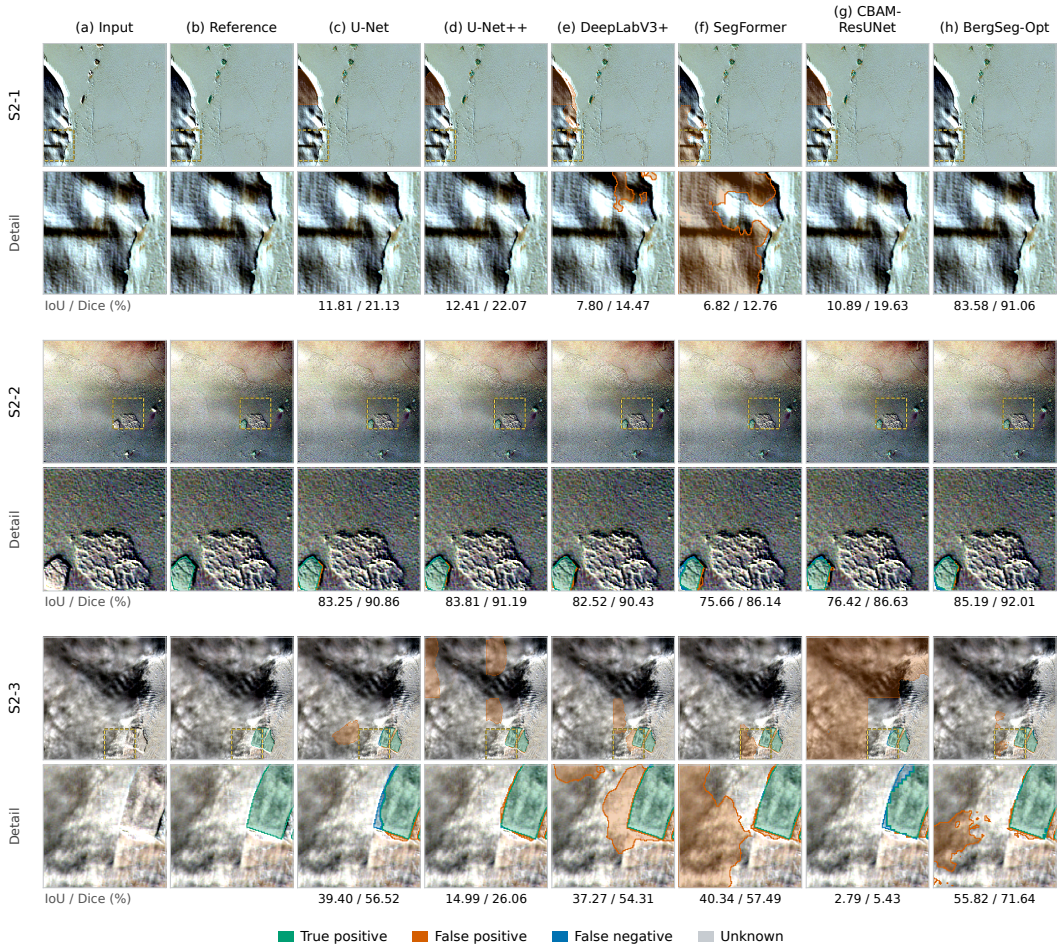


Figure 5: Selected Sentinel-2 improvement examples for BergSeg-Opt. The three complete 512×512 parents illustrate improvements over SegFormer. S2-1 and S2-3 are retained from a previous improvement ranking; S2-2 is chosen for visual clarity and the highest BergSeg-Opt IoU among displayed models. Each is paired with a shared 128×128 error-reduction detail crop enlarged fourfold. Per-channel second-to-98th-percentile B08/B04/B02 display stretches are identical across models for each parent. Scores use the full parent; colors follow Figure 4. These post hoc selections show improvements over SegFormer, not necessarily over every comparator in each row. U-Net and CBAM-ResUNet use batch size 8; other runs use 32. Full-test results are in Table 10.

C.3 Sea-ice and Oil-spill Segmentation Tools

Sea-ice segmentation. IceSeg uses the two HH/HV channels of DSI Sentinel-1 quicklooks to segment sea ice from water. SeaIceNet adds polarization-specific local-contrast features, a coarse context path, and gated detail refinement to a SegFormer-B2 backbone. The quicklooks are display-scaled uint8 images; their channels are not calibrated backscatter. The current 891 training, 144 validation, and 171 reserved test inputs are grouped by acquisition.

Table 11: Sea-ice development validation on the same 144 full inputs. Scores are foreground percentages from each run's validation-selected checkpoint (seed 42); parameters are in millions. The lower block tests the polarization gate.

Model	Parameters	Ice IoU	Dice
U-Net / ResNet-34	24.43	95.09	97.49
DeepLabV3+ / ResNet-34	22.43	94.87	97.36
SegFormer-B2	24.72	95.07	97.47
SeaIceNet (full)	24.78	**95.15**	**97.52**

The complete model shows a certain improvement in IoU compared to the listed general baseline model. Figure 6 uses two validation examples selected randomly by reference ice fraction before inspecting predictions.

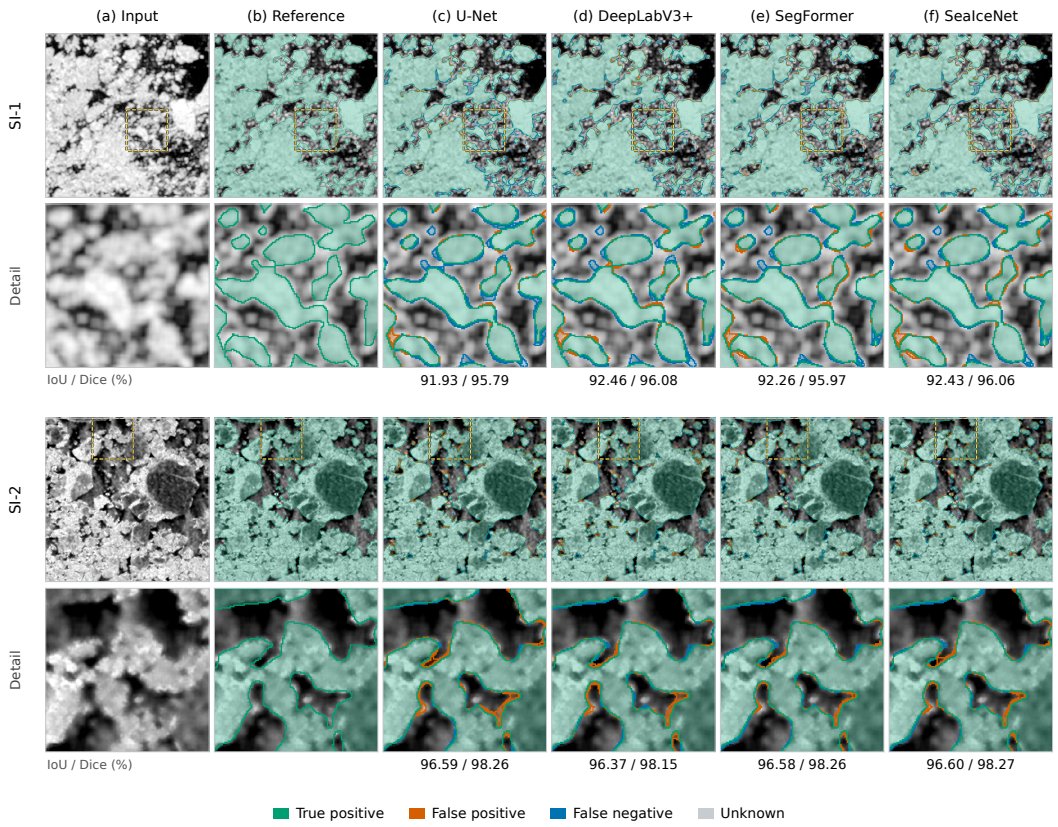


Figure 6: Sea-ice validation examples. Columns share each HH quicklook, reference mask, and display stretch. Teal, vermilion, blue, and gray mark TP, FP, FN, and unknown pixels, as in Figure 4. Yellow boxes identify shared 128×128 detail crops chosen by reference-boundary density; printed IoU/Dice are for each complete input. Examples are illustrative, while the full validation comparison is in Table 11.

Oil-slick segmentation. OilContrast processes paired VV/VH calibrated dB patches with local contrast descriptors, multiscale context, and a scale-gated decoder. We compare it with three established architectures on the same nonpolar Sentinel-1 validation data: 77 source images yield 308 fixed 256×256 patches. Training uses 1,030 source images.

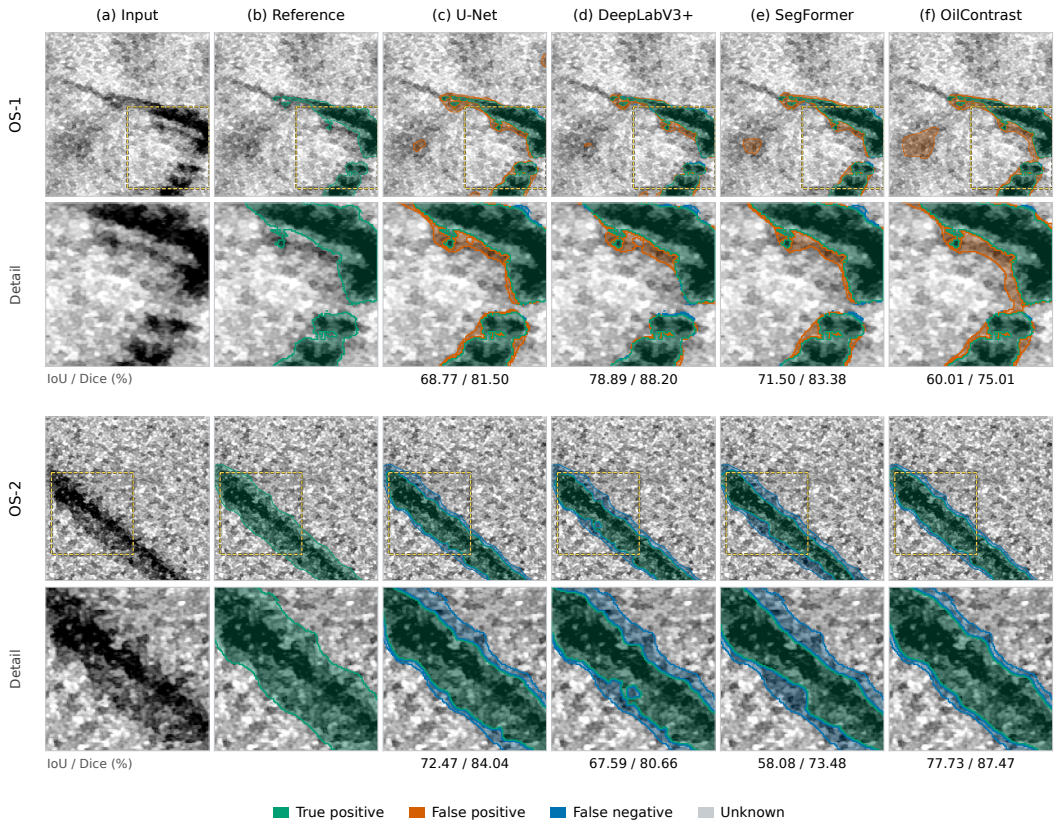


Figure 7: Oil-slick validation examples with the same VV display and reference for every model. Colors and shared detail crops follow Figure 6; scores refer to complete 256×256 patches.

C.4 Tools List

The current source registry comprises 113 analysis-tool interfaces across five categories (38, 14, 27, 26, and 8, respectively). Each table summarizes tool functionality, high-level inputs and outputs, and the implementation backend and status. **Backend:** Py denotes Python numerical or geospatial processing; Rule denotes threshold or connectivity rules; Model denotes a learned-model adapter; SNAP denotes an external SNAP processing adapter; and SAM2 denotes the local SAM 2.1 tiny model. VLM denotes a vision-language-model-dependent pipeline. **Status:** I denotes an implemented interface, B a limited baseline, and C an adapter requiring compatible model weights or external software. These labels describe implementation availability rather than predictive accuracy. The two separate evaluation interfaces are excluded from this list. The interface count is reconciled against the current source registry.

Four task-specific segmentation systems have been trained for the study: BergSeg-SAR, BergSeg-Opt, SeaIceNet, and OilContrast. They correspond to three tool interfaces because the two iceberg branches share `IcebergSegment`. Table 12 distinguishes these research checkpoints from older weights already packaged for inference; training a model does not itself establish deployment or independent-domain validity. The externally trained LeadNet backend is not included among these four systems.

Table 12: Four trained segmentation systems and their corresponding tool interfaces. "Research checkpoint" refers to the model evaluated in this paper; the bundled inference adapter may still select an earlier checkpoint.

System	Input and output	Tool interface	Current evidence and deployment
BergSeg-SAR	Calibrated single-channel SAR → iceberg mask	`IcebergSegment`	Trained and compared on common SAR test inputs; the bundled H05 weight is an epoch-14 pilot snapshot.
BergSeg-Opt	Sentinel-2 B08/B04/B02 reflectance → iceberg mask	`IcebergSegment`	Trained with component ablations; the bundled SegFormer weight is an earlier baseline snapshot.
SeaIceNet	Sentinel-1 HH/HV quicklooks → ice/water mask	`IceWaterSegment`	Development validation only; current bundled S1/S2 ice-water weights use different legacy/weak-label protocols.
OilContrast	Calibrated Sentinel-1 VV/VH dB → oil-candidate mask	`OilSpillCandidateClassifier`	Non-polar development validation only; the bundled oil weight is an earlier prototype, not this full checkpoint.

C.4.1 Data Access and Preprocessing Tools

The data access and preprocessing toolkit prepares heterogeneous observations for subsequent analysis. Its interfaces support product inspection, validity checks, radiometric decoding, grid alignment, tiling, and sensor-specific processing. Catalog and asset operations use local resources.

Table 13: Data Access and Preprocessing Tools (38 interfaces).

Tool	**Function**	**Input**	**Output**	**Backend / status**
`RasterInspect`	Inspect raster structure and georeferencing.	Raster	Dimensions, bands, CRS and validity	Py / I
`BandSelect`	Select and reorder raster bands.	Raster; band indices	Multiband raster	Py / I
`ImageCrop`	Crop by pixel window or georeferenced polygon.	Raster; window or polygon	Georeferenced crop	Py / I
`ProductRead`	Decode values using explicit scale, offset and invalid codes.	Raster or NetCDF; decode rules	Decoded raster	Py / I
`OpticalDecode`	Decode optical reflectance; no atmospheric correction.	Optical DN; scale and offset	Reflectance raster	Py / I
`SARUnit Convert`	Convert declared linear SAR power and decibels.	SAR raster; source/target units	Converted raster	Py / I
`ValidMask Build`	Combine data validity, quality and land masks.	Raster; optional masks	Valid-pixel mask	Py / I
`Observation Check`	Check required bands, CRS and valid-data coverage.	Raster; check criteria	Check report	Py / I
`RasterRegrid`	Align a raster to an explicit reference grid.	Raster; target grid; resampling method	Aligned raster	Py / I
`ImageTile`	Split a raster with optional overlap.	Raster; tile size and halo	Tiles and manifest	Py / I
`TileMerge`	Reassemble tiles on the source grid.	Tiles; manifest	Merged raster	Py / I

Continued on next page

Table 13 (continued)

Tool	Function	Input	Output	Backend / status
SARSpeckle Filter	Filter speckle in linear-power SAR.	SAR raster; filter window	Filtered raster	Py / I
TableRead	Read structured tabular records.	CSV, JSON or JSONL	Records	Py / I
SAFEInspect	Inspect SAFE metadata and product assets.	SAFE manifest	Metadata and asset map	Py / I
CatalogSearch	Search a local catalog by metadata, time and region.	Local catalog; filters	Matching entries	Py / I
AssetFetch	Copy an allowed local asset; optionally verify its checksum.	Local asset; optional checksum	Copied asset	Py / I
Reference Retrieve	Search local references with source attribution.	Text/Markdown files; query	Text hits and source lines	Py / I
ModelInput Prepare	Apply declared band order, units and normalization.	Co-grid bands; input specification	Normalized raster	Py / I
SARCalibrate	Calibrate Sentinel-1 GRD backscatter.	SAFE; polarization and noise settings	Linear sigma-nought raster	SNAP / C
SARTerrain Correct	Calibrate and terrain-correct a SAR product.	SAFE; DEM; target grid	Geocoded backscatter	SNAP / C
ImageRegister	Estimate a global translational image alignment.	Reference and moving rasters	Shift, score and aligned raster	Py / B
Raster Coverage	Measure valid-pixel count and coverage.	Raster	Coverage statistics; optional mask	Py / I
RasterWindow Export	Export pixels with masks and georeferencing preserved.	Raster; optional window	GeoTIFF	Py / I
RasterPoint Map	Map raster pixel positions to geographic coordinates.	Raster; pixel points; target CRS	Map coordinates	Py / I
ClassMap Decode	Map class codes to foreground, background and invalid.	Class raster; code mapping	Binary/invalid mask	Py / I
GCPRaster Reproject	Reproject a GCP-georeferenced raster.	GCP raster; target grid	Reprojected raster	Py / I
MaskRestore Grid	Restore a preview-grid mask to its source-scene grid.	Preview mask; source template	Full-grid mask	Py / I
CFGridExport	Decode and export a regular CF field on its declared grid.	NetCDF field; axes; quality rules	Float64 GeoTIFF and validity statistics	Py / I
DateSeries Read	Select dated CSV observations in an explicit interval.	CSV; date/value columns; interval	Ordered dated values	Py / I
Reference Search	Search a frozen evidence corpus with citations.	Local index; query	Ranked source passages	Py / I
ReferenceRead	Read a cited passage and surrounding source lines.	Local index; passage ID	Verified source text	Py / I
S1LUT Calibrate	Apply native Sentinel-1 calibration/noise LUTs to a crop.	GRD crop; source origin; XML LUTs	Calibrated backscatter	Py / I
S1Preprocess	Run a configured Sentinel-1 GRD processing chain.	SAFE; orbit/noise/DEM options	Sigma-nought product or plan	SNAP / C

Continued on next page

Table 13 (continued)

Tool	Function	Input	Output	Backend / status
`DatedRaster Composite`	Compose dated rasters on their declared common support.	Dated rasters; composition rules	Composite raster	Py / I
`Conservative PolarRegrid`	Regrid polar fields by area overlap.	Source field; target grid	Area-conserving raster	Py / I
`OrthoCutline`	Cut an orthomosaic using a georeferenced polygon.	Orthomosaic; cutline	Alpha-preserving raster crop	Py / I
`OrthoTile Mosaic`	Reassemble aligned orthomosaic tiles with overlap checks.	Tiles; target grid	Orthomosaic and coverage	Py / I
`OrthoValid Coarsen`	Aggregate RGBA cells while respecting transparency.	Orthomosaic; block size	Coarsened raster and valid counts	Py / I

C.4.2 Object Perception and Region Extraction Tools

The object perception and region extraction toolkit produces masks, connected components, boundaries, and object candidates. It combines explicit image-processing rules with learned-model adapters and prompt-based segmentation.

Table 14: Object Perception and Region Extraction Tools (14 interfaces).

Tool	Function	Input	Output	Backend / status
`Threshold Segment`	Segment values using an explicit threshold.	Numeric raster; threshold	Binary mask	Rule / B
`DarkRegion Extract`	Extract dark SAR regions as candidates.	SAR raster; threshold	Candidate mask	Rule / B
`Instance Extract`	Label connected regions; touching targets may merge.	Class mask; connectivity	Instance mask and attributes	Rule / B
`MaskVector Convert`	Convert raster masks to georeferenced polygons.	Mask raster	Vector features	Py / I
`IceEdge Extract`	Extract ice boundary pixels adjacent to observed water.	Ice concentration; threshold	Ice-edge raster	Rule / I
`Iceberg Segment`	Segment icebergs with a selected SAR or optical model.	Calibrated SAR or B08/B04/B02 reflectance; model ID	Candidate probability and mask	Model / C
`IceWater Segment`	Run a selected bundled ice-presence or weak-label ice/water model.	Raster; sensor-specific bands; model ID	Ice/water class map	Model / C
`LeadSegment`	Call the published pretrained LeadNet model (Yang et al., 2026).	Five author-defined SAR feature bands; model ID	Lead mask	Model / C
`SeaIceType Classify`	Classify sea-ice types.	Raster; model specification	Ice-type class map	Model / C
`OilSpill Candidate Classify`	Map oil-like appearance with an experimental bundled prototype.	Calibrated VV/VH dB; model ID	Candidate probability and mask	Model / C
`ShipDetect`	Detect candidate vessels.	Image; detector settings	Boxes and scores	Model / C

Continued on next page

Table 14 (continued)

Tool	Function	Input	Output	Backend / status
`PromptSegment`	Segment a prompted region without assigning a semantic class.	Image; points or box	Prompt mask	SAM2 / I
`InstructSAM`	Propose text-requested instances using VLM, SAM2 and RemoteCLIP.	Image; text prompt; model configuration	Candidate instances and union mask	VLM / C
`Segmentation Capabilities`	Inspect installed segmentation backends and input contracts.	Optional tool name	Model availability and scope	Py / I

C.4.3 Measurement, Statistics, and Spatial Analysis Tools

The measurement, statistics, and spatial analysis toolkit quantifies the outputs of perception tools. It supports area and shape measurements, object counting, weighted statistics, spatial relationships, and explicit uncertainty propagation.

Table 15: Measurement, Statistics, and Spatial Analysis Tools (27 interfaces).

Tool	Function	Input	Output	Backend / status
`Calculator`	Evaluate a restricted arithmetic expression.	Numeric expression	Number	Py / I
`AreaMeasure`	Measure polygon area under a stated CRS/method.	Polygon; CRS	Area in square metres	Py / I
`ShapeMeasure`	Measure perimeter, length and width.	Polygon; metric CRS	Shape measurements	Py / I
`Zonal Statistics`	Calculate area-weighted summary statistics.	Values; area weights; valid mask	Zonal statistics	Py / I
`IceAreaExtent`	Compute sea-ice area and thresholded extent.	Ice concentration; area weights; threshold	Ice area and extent	Py / I
`Distribution Statistics`	Summarize a numeric distribution.	Values; bins	Distribution and histogram	Py / I
`ObjectFilter Count`	Filter objects by size and boundary rule, then count them.	Object records; filter criteria	Filtered count	Py / I
`Spatial Relation`	Evaluate spatial predicates between geometries.	Two geometries in a common CRS	Relation result	Py / I
`Distance Bearing`	Calculate distance and bearing or nearest separation.	Points or shapes; CRS; method	Distance and bearing	Py / I
`TransectWidth`	Measure a region's width along a specified transect.	Polygon; line; metric CRS	Intersected width	Py / I
`NetworkBuild`	Build a skeleton graph with width estimates.	Binary mask; validity; pixel size	Width-annotated graph	Py / B
`NetworkPath`	Find a width-constrained graph path.	Graph; endpoints; minimum width	Candidate path	Py / B
`Uncertainty Propagate`	Propagate specified absolute error bounds.	Values; errors; operation	Conservative interval	Py / I
`EvidenceCheck`	Check evidence against time, region and field requirements.	Records; applicability criteria	Check report	Py / I
`Product Consistency`	Quantify incompatible product combinations.	Co-grid layers; rules; weights	Inconsistency statistics	Py / I

Continued on next page

Table 15 (continued)

Tool	Function	Input	Output	Backend / status
RasterMath	Apply arithmetic to rasters over their common valid domain.	Co-grid rasters; operation	Result raster	Py / I
ProductPoint Query	Query a raster value at a geographic point.	Raster; coordinates; decoding rules	Value and validity	Py / I
AreaAggregate	Aggregate by source–target polygon overlap.	Values; polygons; metric CRS	Area-weighted values	Py / I
FieldSample	Retrieve field observations near a place and time.	Records; location/time tolerances	Nearby records	Py / I
MaskRegion Statistics	Summarize target pixels within grid regions.	Mask; grid partition	Counts, fractions and locations	Py / I
MaskTopology	Measure connected-region topology.	Mask; connectivity and filters	Component statistics	Py / I
MaskTransect	Report contiguous mask intervals along a row or column.	Target mask; transect index	Pixel intervals	Py / I
MaskClearance	Estimate clearance using a distance transform.	Binary mask; metric grid	Clearance proxy	Py / I
RasterClass Contrast	Compare SAR backscatter across mask classes.	SAR raster; class mask	Per-class contrast statistics	Py / I
PixelPolygon Statistics	Filter and summarize pixel-space polygons.	Pixel polygons; filter criteria	Counts and pixel areas	Py / I
MaskObject Metrics	Count supplied-mask components and optional geodesic areas.	Mask; connectivity and area rules	Object counts and areas	Py / I
RasterSummary	Summarize finite pixels with explicit interval rules.	Raster; validity; quantiles	Pixel statistics	Py / I

C.4.4 Temporal Change and Motion Analysis Tools

The temporal change and motion analysis toolkit compares repeated observations and summarizes changes, displacement, and trajectories. Matching interfaces produce candidates; trajectory construction uses explicitly supplied associations.

Table 16: Temporal Change and Motion Analysis Tools (26 interfaces).

Tool	Function	Input	Output	Backend / status
Motion Statistics	Calculate displacement, path length and speed.	Geolocated positions; timestamps	Motion statistics	Py / I
Trajectory Build	Build tracks from explicit observation associations.	Observations; associations	Track graph and branches	Py / I
ChangeMeasure	Measure gains, losses and persistence on common support.	Mask pair; validity; area weights	Transition areas	Py / I
Temporal Aggregate	Aggregate observations by day, month or year.	Times; values; period	Group means and counts	Py / I
Threshold Event	Find a calendar-consecutive threshold event.	Dates; values; threshold; duration	Event interval	Py / I

Continued on next page

Table 16 (continued)

Tool	Function	Input	Output	Backend / status
`TrendAnomaly`	Estimate trend and reference-period anomaly.	Annual values; reference period	OLS/Theil–Sen slopes and anomalies	Py / I
`ObjectMatch`	Propose distance-gated associations and retain ambiguity.	Object sets; metric CRS; distance limit	Candidate associations	Py / B
`IceMotion Estimate`	Estimate local integer-pixel shifts by correlation.	Co-grid image pair; windows	Displacement candidates and scores	Py / B
`MotionField Transform`	Convert pixel shifts to east/north velocity.	Vectors; georeferencing; time interval	Velocity components	Py / I
`PixelTrackTo Geo`	Map supplied pixel centers to a geographic track.	Images; pixel centers; times	WGS84 track	Py / I
`TrackRegion Events`	Bracket observed entries into and exits from a region.	Track; region polygon; CRS	Inside states and event intervals	Py / I
`Motion Interval Bounds`	Bound motion using specified position errors.	Positions; times; error radii	Distance and speed bounds	Py / I
`TrackTurn Analysis`	Measure track turning and segment motion.	Track; projected metric CRS	Turn angles and motion metrics	Py / I
`Temporal Raster Composite`	Count threshold occurrences and consecutive observations.	Aligned rasters; times; threshold	Count, run-length and first-occurrence raster	Py / I
`RasterSeries Metrics`	Compute same-support concentration time-series metrics.	Aligned rasters; pixel areas; times	Mean concentration, extent and ice area	Py / I
`ImageTemplate Match`	Search rotated templates within an explicit window.	Image pair; template/search windows	Match candidates and scores	Py / B
`NetCDFWindow Compare`	Compare named fraction fields on common valid pixels.	NetCDF fields; threshold	Pixel-weighted statistics	Py / I
`Observation PairAudit`	Check acquisition order and time gaps without asserting identity.	Observation times; gap limit	Pair chronology report	Py / I
`RasterCommon Support`	Intersect valid pixels across aligned rasters.	Co-grid rasters	Common-support mask	Py / I
`RasterMean`	Average aligned rasters over their common support.	Co-grid raster series	Mean raster	Py / I
`Series Correlation`	Correlate explicitly paired finite series.	Two numeric series	Correlation coefficient	Py / I
`FeatureMatch AKAZE`	Generate cross-image feature correspondences.	Image pair; masks; match settings	Match candidates	Py / B
`MotionVector Filter`	Reject spatial outliers among georeferenced matches.	Match records; neighborhood rules	Filtered vectors	Py / B
`MotionVector FilterPixels`	Reject spatial outliers in pixel-space matches.	Match records; pixel thresholds	Filtered pixel vectors	Py / B
`Track Kinematics`	Compute observed-track displacement and speed.	Dated positions; coordinate convention	Track metrics	Py / I
`CFSeries Diagnostics`	Audit common support and differences among CF time slices.	CF fields; time selection; validity rules	Series comparison diagnostics	Py / I

C.4.5 VISUALIZATION AND RESULT EXPORT TOOLS

The visualization and result export toolkit presents observations and analysis results as images, charts, and structured files. Preview generation and overlay rendering retain the mapping between display coordinates and source data.

Table 17: Visualization and Result Export Tools (8 interfaces).

Tool	Function	Input	Output	Backend / status
`ImagePreview`	Render a preview using explicit bands and stretch limits.	Raster; bands; display limits	PNG and coordinate mapping	Py / I
`Coordinate Pick`	Map a display point to source/map coordinates.	Display point; crop/scale mapping	Pixel/map coordinates	Py / I
`MapRender`	Overlay pixel-space points, lines and polygons.	Preview; geometries	Annotated image	Py / I
`MatchRender`	Show cross-image links with explicit match states.	Image pair; match records	Match visualization	Py / I
`ChartRender`	Plot supplied values as line, bar or error-bar charts.	Numeric series; chart settings	Chart image	Py / I
`ResultExport`	Export computed records.	Records; output format	JSON, GeoJSON or CSV	Py / I
`GeoRasterMap`	Render georeferenced imagery with CRS-aware overlays.	Raster; overlays; style	Geographic map image	Py / I
`DriftTrackMap`	Map observed drift tracks with declared coordinates.	Track positions; map settings	Track map image	Py / I

D EXPERIMENTAL DETAILS

D.1 DETAILS OF EVALUATION METRIC

Let $S_i^m \in \{0,1\}$ denote whole-task success for mode m. An eligible unsuccessful run includes budget exhaustion or an invalid final submission; unresolved references and infrastructure failures remain outside this denominator. Let $\mathcal{P}$ contain compatible GS/TA pairs for one model. We compute

$$\text{Accuracy}_m = 100\,|\mathcal{P}|^{-1} \sum_{i\in\mathcal{P}} S_i^m, \tag{4}$$

$$\text{Tool-Gain} = 100\,|\mathcal{P}|^{-1} \sum_{i\in\mathcal{P}} (S_i^{TA} - S_i^{GS}). \tag{5}$$

The second measure is a paired accuracy difference in percentage points; it equals $100(n_{TA-\text{only}} - n_{GS-\text{only}})/|\mathcal{P}|$. It is descriptive rather than an additional independent measure of correctness.

For task i, let A_i be the independently assessable checkpoints and p_{ij} the binary outcome of checkpoint j. Define I as tasks with at least one assessable checkpoint. Then

$$\text{Critical-Step} = \frac{100}{|I|} \sum_{i\in I} \frac{\sum_{j\in A_i} p_{ij}}{|A_i|}. \tag{6}$$

We report both the number of contributing tasks and the number of assessed checkpoints. Missing artifacts or unavailable evidence are left unscored rather than treated as either success or failure. An execution error does not by itself imply that a scientific checkpoint failed; conversely, a syntactically valid call does not establish correct units, valid-pixel support, or coordinate handling. Checkpoint annotations cover all 710 tasks, while valid stage evidence is available for a model- and mode-dependent subset. Critical-Step is computed only over tasks with assessable checkpoint evidence, and the corresponding coverage is reported separately.

Let $\mathcal{C} \subseteq \mathcal{P}$ contain jointly correct pairs with positive, finite elapsed times. The main timing measure and its companion are

$$r_i = 100 \left(1 - \frac{T_i^{TA}}{T_i^{GS}}\right), \tag{7}$$

$$\text{Time-Saving} = \text{median}_{i \in \mathcal{C}}\, r_i, \tag{8}$$

$$\text{TA faster share} = \frac{100}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \mathbf{1}[T_i^{TA} < T_i^{GS}]. \tag{9}$$

Ties count in the denominator but not as faster. Empty pair sets produce missing values. Elapsed time covers actual solving, including model and tool waits, but excludes queueing, initialization, and scoring. Timing comparisons are descriptive because historical endpoint and concurrency conditions vary. The median is taken over individual relative savings; it is not the saving computed from two mean times. Neither metric describes failed tasks.

For a valid run with $c_i > 0$ observed function attempts and f_i execution failures, we use

$$\text{Failure-Rate} = \frac{100}{|E|} \sum_{i \in E} \frac{f_i}{c_i}. \tag{10}$$

E contains runs with an available execution denominator. Input/output functions and unsuccessful attempts are included; infrastructure or timeout calls excluded by the logged execution-metric policy are kept separate. An additional trace audit removes the explicitly identified worker file-policy rejection from both numerator and denominator (374 records, 1,434 calls). A direct answer with no observable call denominator is unscored for this metric. Task averaging prevents long, repeatedly failing trajectories from dominating all short trajectories. Parameter-format, runtime, and scientific errors should be diagnosed separately; this metric measures execution failures, not semantic parameter correctness or cross-run stability.

Tolerance and provenance. Scoring retains each task's frozen checks. Numeric comparisons use the declared absolute and relative tolerances, typically $|\hat{y} - y| \leq a + r|y|$, rather than applying a new global five-percent rule. Counts, categories, and pixel memberships can require exact equality. Numeric representations such as 0 and 0.0 are equivalent; strings and file paths are validated according to their typed contract. Spatial artifacts require their own grid, validity, topology, and overlap checks. A plausible path without a readable artifact is not sufficient evidence.

The analysis pipeline first selects active versioned runs, then checks pair compatibility and aggregates existing evaluator outputs. It does not ask a language model to grade another language model. Each checkpoint score must retain its oracle version and evidence source. Errors discovered in evaluators require re-scoring the corresponding frozen artifacts and retaining the old score for audit. Unknown evidence remains unknown.

Taxonomy reconciliation. Version v8 assigns one primary objective to every question and retains secondary capability tags. Raw-image segmentation belongs to perception, whereas supplied-mask inventory and topology belong to object/spatial analysis. Product decoding and support-aware physical summaries belong to quantitative diagnostics even when multiple dates are involved; correspondence and displacement belong to temporal change. This moves 61 segmentation questions into perception, 60 inventory questions into object/spatial analysis, and 80 product-diagnostic questions out of the broad change category. Assignment uses question content, not model outcomes. The resulting categories remain unequal, and reclassification creates no new independent observations.

D.2 Evaluation Models and Scale Details

The current snapshot covers the 13 model configurations listed in Table 18.

Gemma4 and InternVL3.5 models are deployed locally using their released open-source checkpoints, while all other models are evaluated through their official API endpoints. Parameter counts, architecture details, and model specifications are collected from the corresponding technical reports, model cards, and official documentation.

Table 18: Models evaluated under the GS and TA settings. Parameter counts follow official model documentation. For sparse MoE models, active parameters denote the parameters activated during inference. For Gemma E-series models, effective parameters follow the official definition. Dense models use total parameters as effective scale.

Family	Model	Architecture	Params	Access / Source
GPT	GPT-4o	Dense MLLM	–	OpenAI API
GPT	GPT-6-Luna	Dense MLLM	–	OpenAI API
GPT	GPT-6-Sol	Dense MLLM	–	OpenAI API
Claude	Claude Sonnet 5	Dense MLLM	–	Anthropic API
DeepSeek	DeepSeek Flash	Sparse MoE	552B / 8–16B active	API / Model card
Gemma	Gemma4-E2B	Dense + PLE	5.1B / 2.3B effective	Open checkpoint
Gemma	Gemma4-E4B	Dense + PLE	8.0B / 4.5B effective	Open checkpoint
Gemma	Gemma4-12B	Dense	11.95B	Open checkpoint
MiMo	MiMo-V2.5	Sparse MoE	310B / 15B active	Open checkpoint
MiMo	MiMo-V2.6-Flash	Sparse MoE	309B / 15B active	Open checkpoint
MiMo	MiMo-V2.6-Pro	Sparse MoE	1.02T / 42B active	Open checkpoint
InternVL	InternVL3.5-4B	Dense MLLM	4.7B	Open checkpoint
InternVL	InternVL3.5-8B	Dense MLLM	8.5B	Open checkpoint

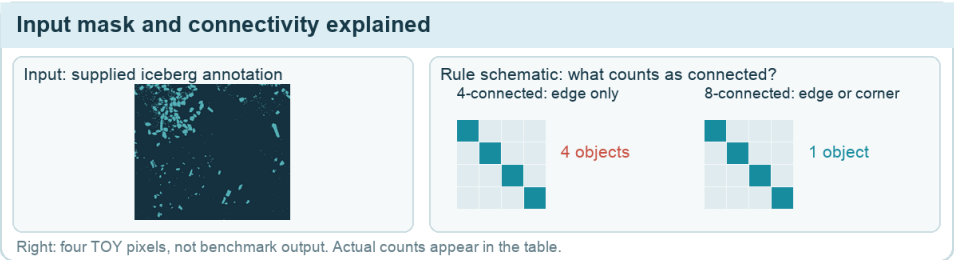
E Case Study

The following ten illustrative cases use version-matched recorded answers from DeepSeek and GPT-6-Sol. They are selected for workflow diversity, not to estimate error frequencies. Times are descriptive; full record identifiers are retained in the analysis export.

Case 1: Correct objects, wrong geodesic areas (PC600-301)

Task. Crop a supplied iceberg mask to a polygon with a hole; compare 4- and 8-neighbour components, excluding boundary-limited objects.

Inputs. supplied_iceberg_annotation.tif; study_footprint.json



Visual: Actual input mask and connectivity-rule schematic.

Model outputs and decisive checks.

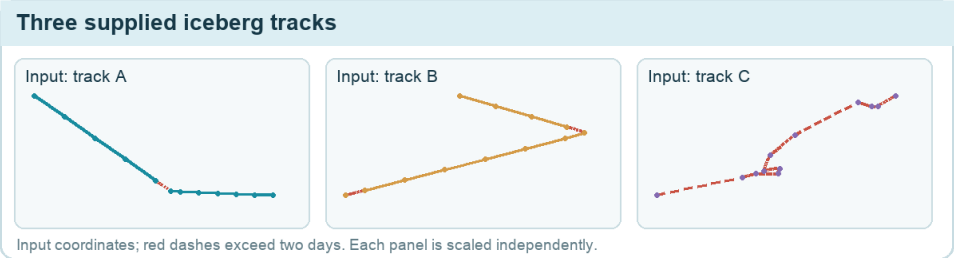
Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	FAIL	36.8 s	Both component counts = 212; area lists and derived area/density checks fail.
DeepSeek Flash / TA	NO SUBMISSION	68.9 s	No accepted answer: output limit reached.
GPT-6-Sol / GS	PASS	576.7 s	All 27 recorded checks pass.
GPT-6-Sol / TA	FAIL	623.1 s	Only the 8-neighbour area list fails; other checks pass.

What this case shows. A correct component count does not establish correct area computation. Tool access did not guarantee success in this case.

Case 2: Motion reports need more than plausible tracks (PC600-461)

Task. For three iceberg tracks, report connected geodesic segments, uncertainty bounds, signed turns and gap-aware totals.

Inputs. track_A.json; track_B.json; track_C.json



Visual: Actual input coordinates; red dashes denote gaps over two days.

Model outputs and decisive checks.

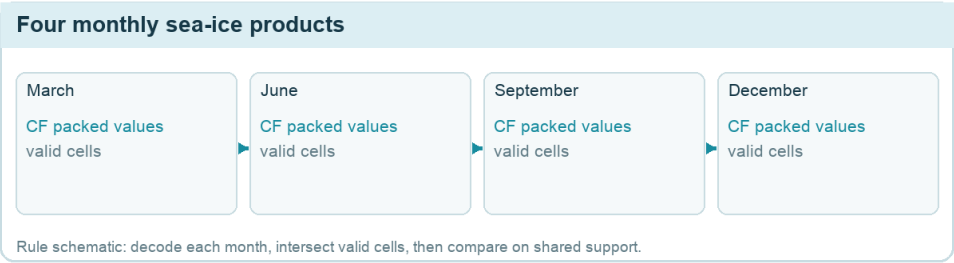
Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	**FAIL**	9.0 s	One distance bound, turn-array lengths and track-C connected totals fail.
DeepSeek Flash / TA	**FAIL**	13.3 s	The same six recorded checks fail despite tool access.
GPT-6-Sol / GS	**FAIL**	67.8 s	Several signed turn values fail; most other fields pass.
GPT-6-Sol / TA	**PASS**	168.6 s	All 654 recorded fields pass with TrackKinematics.

What this case shows. The specialist motion operation helped GPT-6-Sol on signed turns, but DeepSeek did not realize the same gain. Failed fields identify symptoms, not the first causal error.

Case 3: Correctness and speed must be separated (PC600-521)

Task. Decode four monthly CF sea-ice products and calculate statistics on exactly their shared valid cells.

Inputs. month_1.nc through month_4.nc



Visual: Four-product common-support rule schematic.

Model outputs and decisive checks.

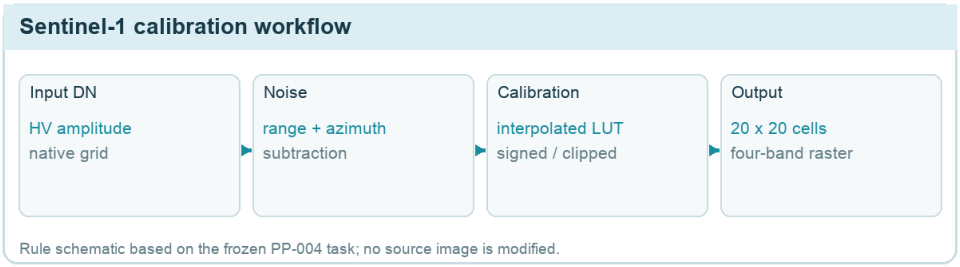
Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	**PASS**	9.4 s	All 53 scientific checks pass.
DeepSeek Flash / TA	**PASS**	17.4 s	All 53 scientific checks pass.
GPT-6-Sol / GS	**PASS**	54.7 s	All 53 scientific checks pass.
GPT-6-Sol / TA	**PASS**	24.3 s	All 53 scientific checks pass using CFSeriesDiagnostics.

What this case shows. Both modes solve this task. TA was faster for GPT-6-Sol and slower for DeepSeek in these recorded runs.

Case 4: Sensor-aware calibration changes raster values (PP-004)

Task. Use native Sentinel-1 DN and annotation XML to create four HV calibration diagnostics on 20-by-20-pixel cells.

Inputs. HV DN crop; annotation, calibration and noise XML



Visual: Sensor-calibration rule schematic.

Model outputs and decisive checks.

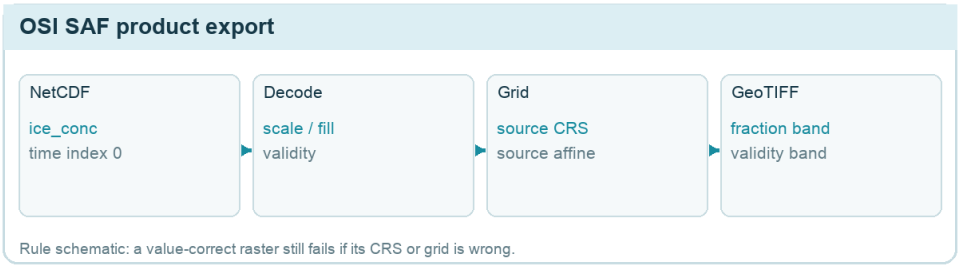
Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	**FAIL**	79.5 s	All four bands fail; signed-mean matching fraction is 4.7%.
DeepSeek Flash / TA	**PASS**	32.7 s	All four bands pass; every checked cell matches.
GPT-6-Sol / GS	**FAIL**	505.3 s	All four bands fail; signed-mean matching fraction is 0%.
GPT-6-Sol / TA	**PASS**	115.3 s	All four bands pass; every checked cell matches.

What this case shows. GS preserved the validity layout yet failed on scientific raster values. The specialist preprocessing path passed for both models.

Case 5: A plausible GeoTIFF can have the wrong grid (PP-026)

Task. Export Antarctic ice concentration to a fraction band and a validity band while preserving its CRS and affine grid.

Inputs. ice_conc_sh_polstere-100_amsr2_202509151200.nc



Visual: CF decoding and grid-preservation rule schematic.

Model outputs and decisive checks.

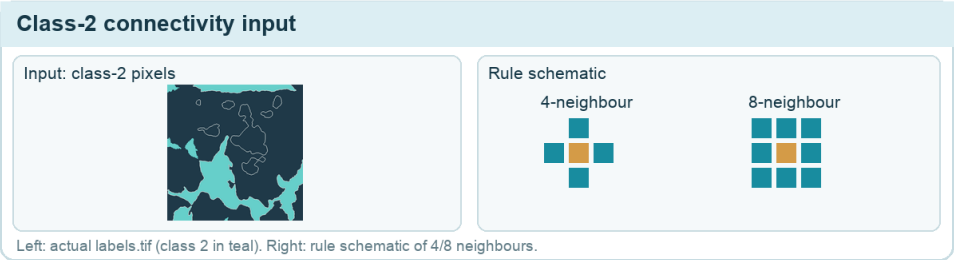
Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	**FAIL**	12.7 s	GeoTIFF submitted, but the scorer reports wrong CRS or affine grid.
DeepSeek Flash / TA	**NO SUBMISSION**	64.0 s	No accepted answer: call limit reached.
GPT-6-Sol / GS	**PASS**	248.9 s	Both bands and the grid pass.
GPT-6-Sol / TA	**PASS**	274.7 s	Both bands and the grid pass.

What this case shows. A spatial-reference error differs from incorrect pixel values. An unsubmitted run must be kept distinct from a scored scientific failure.

Case 6: Self-written code can succeed (PC300-004)

Task. Count class-2 components under both 4- and 8-neighbour connectivity in the supplied label raster.

Inputs. labels.tif



Visual: Actual label raster and connectivity-rule schematic.

Model outputs and decisive checks.

Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	**PASS**	11.4 s	Both analyses pass: seven components each.
DeepSeek Flash / TA	**PASS**	13.7 s	Both analyses pass: seven components each.
GPT-6-Sol / GS	**PASS**	360.8 s	Both analyses pass: seven components each.
GPT-6-Sol / TA	**PASS**	326.8 s	Both analyses pass: seven components each.

What this case shows. Competent GS code succeeds here. This case measures a limit of tool gain rather than presuming tools are always necessary.

Case 7: Geodesic and planar measurements differ (PC300-071)

Task. Measure the supplied iceberg polygon on WGS84 and a projected plane, including boundary and shape diagnostics.

Inputs. SAR image; supplied GeoJSON polygon

Iceberg polygon and SAR context

Input: SAR image



Measured quantities

Geodesic area	Projected area
945,822 m2	980,800 m2

Relative difference: 3.70%

Input SAR crop shown with contrast stretch; coordinate polygon displayed separately, not overlaid.

Visual: Actual SAR input and checked output values.

Model outputs and decisive checks.

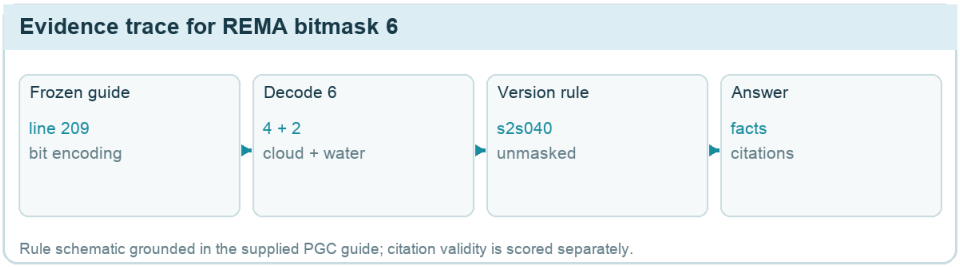
Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	**PASS**	14.9 s	Correct: geodesic area 945,822 m2; projected area 980,800 m2.
DeepSeek Flash / TA	**PASS**	14.2 s	Correct: geodesic area 945,822 m2; projected area 980,800 m2.
GPT-6-Sol / GS	**PASS**	90.1 s	Correct: geodesic area 945,822 m2; projected area 980,800 m2.
GPT-6-Sol / TA	**PASS**	68.0 s	Correct: geodesic area 945,822 m2; projected area 980,800 m2.

What this case shows. The 3.70% area difference is expected between the specified methods, not a model error. All four runs passed.

Case 8: Right facts can still have weak citations (PKR10-008)

Task. Use a frozen DEM guide to decode REMA bitmask 6, explain filtering and identify the first unmasked strip version with line citations.

Inputs. Frozen PGC DEM guide in the supplied corpus



Visual: Bitmask and evidence-rule schematic grounded in the supplied guide.

Model outputs and decisive checks.

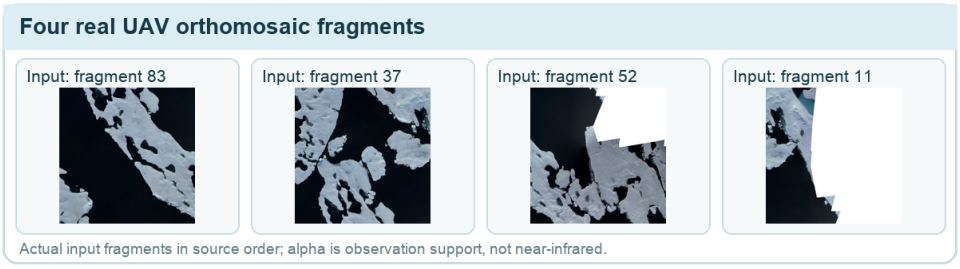
Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	**PASS**	7.3 s	All five facts and citation evidence pass.
DeepSeek Flash / TA	**PASS**	8.6 s	All five facts and citation evidence pass.
GPT-6-Sol / GS	**PASS**	289.4 s	All five facts and citation evidence pass.
GPT-6-Sol / TA	**FAIL**	264.0 s	All five facts pass, but citation evidence score is 0.5.

What this case shows. The GPT-6-Sol TA result is an evidence-attribution failure, not a factual knowledge failure: all five facts were correct.

Case 9: Mosaic tool changes the execution path (PC600-362)

Task. Reassemble four georeferenced UAV fragments without resampling and produce an RGBA mosaic plus per-pixel observation support.

Inputs. fragment_83.tif; fragment_37.tif; fragment_52.tif; fragment_11.tif



Visual: Four actual UAV fragment inputs.

Model outputs and decisive checks.

Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	**NO SUBMISSION**	175.5 s	Tool timeout; no accepted mosaic to score.
DeepSeek Flash / TA	**PASS**	27.8 s	Correct mosaic/support: 325,180 overlap and 1,841,124 valid pixels.
GPT-6-Sol / GS	**PASS**	107.3 s	Correct mosaic and support raster.
GPT-6-Sol / TA	**PASS**	48.2 s	Correct mosaic and support raster via OrthoTileMosaic.

What this case shows. TA completed the mosaic quickly in these runs. The GS timeout is an execution outcome, not evidence of wrong submitted pixels.

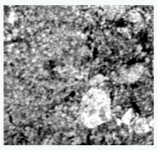
Case 10: Open-ended segmentation remains difficult (PC300-001)

Task. Find the largest complete iceberg in a candidate region, delineate it on the source grid and report WGS84 ellipsoidal area.

Inputs. SAR iceberg image; candidate search GeoJSON

Candidate region is not an iceberg boundary

Input: SAR image



Required result

Image-grounded boundary

+ WGS84 geodesic area

Search polygon: location only

Actual SAR crop with contrast stretch; interpretation must follow the discernible iceberg pixels.

Visual: Actual SAR input; candidate-polygon use explained schematically.

Model outputs and decisive checks.

Model / mode	Result	Time	Selected output or failed check
DeepSeek Flash / GS	NO SUBMISSION	184.8 s	No accepted answer: call limit reached.
DeepSeek Flash / TA	NO SUBMISSION	83.5 s	No accepted answer: call limit reached.
GPT-6-Sol / GS	FAIL	581.8 s	Submitted 0.556 km2; boundary and area checks both fail.
GPT-6-Sol / TA	FAIL	1082.4 s	Submitted 0.595 km2; boundary and area checks both fail.

What this case shows. Tool access did not solve this perception task. The two submitted boundaries fail the independent reference, so their areas cannot validate segmentation.

F Additional Analysis of Model Scale

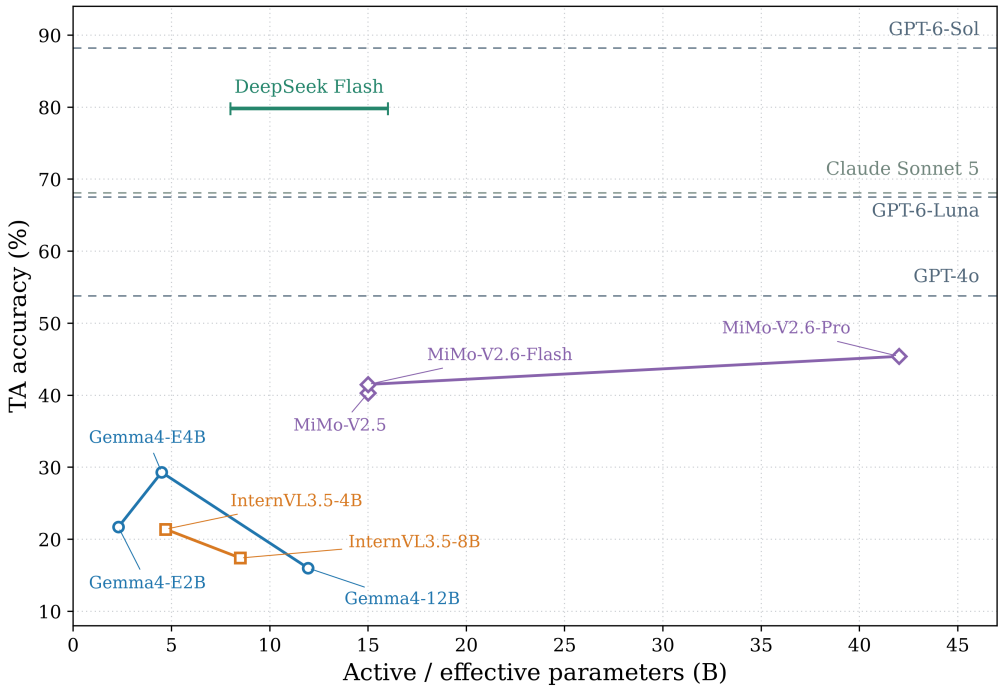


Figure 8: Relationship between active/effective model parameters and Tool-Augmented Solving (TA) accuracy. Each point represents a model configuration evaluated with PolarTools. Larger parameter scale does not consistently correspond to higher TA accuracy, suggesting that tool utilization depends on model capability beyond model size alone.