APPENDIX

A ARTIFACT APPENDIX

A.1 Abstract

EARTHSIGHT is a distributed framework that coordinates inference between ground stations and satellites for low-latency Earth observation. This artifact includes the full source code of the EARTHSIGHT satellite constellation simulator, which extends the Serval simulator (Tao et al., 2024). We provide Bash and SLURM scripts that prepare and launch all experiments, along with Python scripts that reproduce and visualize key results from the paper. The artifact is publicly available at `https://github.com/scai-tech/EarthSight` and archived via Zenodo at `https://doi.org/10.5281/zenodo.18826781`. EARTHSIGHT has been awarded Artifacts Available and Artifacts Functional badges. While some reviewers were able to reproduce our artifact, the reproducibility badge was not awarded due to the prohibitive computational requirements for other reviewers.

A.2 Artifact Check-list (Meta-information)

- **Program:** Python-based satellite edge computing simulator.
- **Datasets:** All required orbital data and ground-station traces are included in the artifact; no external downloads are needed.
- **Run-time environment:** Any high-memory Linux CPU node. Experiments were conducted on RHEL 9 with Slurm.
- **Hardware:** One CPU node with 256 GB RAM per experiment (any modern multi-core CPU is sufficient; memory is the bottleneck).
- **Metrics:** End-to-end image delivery latency, per-image compute time, and on-board power consumption.
- **Output:** Traces, Logs, plain-text result tables and a latency bar chart.
- **Disk space required:** ∼128 GB (simulation logs).
- **Preparation time:** ∼1 hour (environment setup and script generation).
- **Experiment time:** ∼12 hours with a Slurm cluster (jobs run in parallel); ∼3 days serially for the combined scenario only; ∼9 days serially for the full suite (18 runs).
- **Publicly available:** Yes.
- **License:** MIT.
- **Archived (DOI):** `https://doi.org/10.5281/zenodo.18826781`

A.3 Description

A.3.1 How Delivered

The artifact is available as a public GitHub repository and as a ZIP archive, at the links provided below. All code, reference data, and launch scripts are self-contained.

- Github: `https://github.com/scai-tech/EarthSight`
- Zenodo, with reserved DOI: `10.5281/zenodo.18826781`.

A.3.2 Hardware Dependencies

Experiments were conducted on a node with a singleAMD EPYC 32-core CPU and 256 GB RAM. **200+ GB of RAM is a requirement for each simulation run**; any modern CPU is otherwise sufficient. If running in parallel, each replica of the program must have its own resources.

A.3.3 Software Dependencies

Python 3.9 or 3.13 and scientific Python packages (`numpy`, `pandas`, `skyfield`, `matplotlib`, `rtree`, and others) installed from PyPI via `requirements.txt`.

A.3.4 Datasets

All required traces are bundled with the artifact:

- `referenceData/planet_tles.txt` — Planet Labs TLE orbital elements.
- `referenceData/planet_stations.json` — Ground station locations.
- `referenceData/de440s.bsp` — JPL planetary ephemeris (32 MB).

Training data for the image classification models is out of scope for this artifact; filter pass probabilities and execution costs are hard-coded from the pre-trained models.

A.4 Installation

We recommend Python 3.9 or 3.13. After cloning the repository or extracting the ZIP file:

```
python -m venv ./satsim

# Linux / macOS
source satsim/bin/activate

# Windows (PowerShell)
satsim\Scripts\Activate.ps1

python -m pip install --upgrade pip
python -m pip install -r requirements.txt
```

Navigate to the simulator directory and verify that all imports succeed:

```
cd Sat_Simulator/
python verify_imports.py
```

A.5 Experiment Workflow

Each simulation runs 48 simulated hours of satellite operation, taking approximately 12 wall-clock hours per run and requiring 256 GB RAM. Tables 4 and 5 are standalone benchmarks that require no simulation data and can be run immediately after installation. Please refer to the README for commands for easy-to-follow formatting.

A.5.1 Option A: Slurm Cluster

All simulation jobs can be submitted in parallel; total wall time is ∼12 hours regardless of suite size.

```
cd Sat_Simulator

# Generate .sbatch files
# (replace path, account, and email)
python generate_slurm_scripts.py \
    --cluster-path /path/to/EarthSight \
    --account SLURM_ACCOUNT \
    --email you@institution.edu \
    --combined-only # omit for full suite

cd batch_scripts

# Submit standalone tables immediately
# (no simulation data needed)
bash launch_tables_standalone.sh

# Submit all simulation jobs in parallel
bash launch_sims.sh

# After all simulation jobs finish:
bash launch_tables_postrun.sh
```

A.5.2 Option B: Single Machine (No Slurm)

On Linux or macOS (Windows users: use WSL or Git Bash):

```
cd Sat_Simulator

# Generate shell scripts
python generate_batch_scripts.py \
    --combined-only
# (omit flag for full 18-job suite)

chmod +x batch_scripts/*.sh \
    batch_scripts/individual/*.sh

# Run standalone tables immediately
bash batch_scripts/generate_table_4.sh
bash batch_scripts/generate_table_5.sh

# Run simulations -- choose a scope:
# combined + TPU (~36 h)
bash batch_scripts/run_combined_tpu.sh
# combined + GPU (~36 h)
bash batch_scripts/run_combined_gpu.sh
# full combined pipeline (~3 days)
bash batch_scripts/run_all_combined.sh

# After all simulations finish:
bash batch_scripts/generate_results \
        _postrun.sh
```

Individual scripts for each of the 18 simulation configurations are also provided in `batch_scripts/individual/` for fine-grained control.

A.6 Evaluation and Expected Results

The above commands, if all are executed, will replicate Tables 3-5 and Figure 6. Please expect up to 5-10% variability for all experiments. We provide expected outputs in `Sat_Simulator/results_expected/`.

A.6.1 Tables 4 and 5 — Compute Time Benchmarks

These results are produced by standalone scripts and do not depend on simulation runs. Outputs will be written to `results/table4.txt` and `results/table5.txt`.

Table 4 reports per-image compute time for Serval, EARTHSIGHT-STL, and EARTHSIGHT-MTL across Coral/TPU and Jetson/GPU targets, and should reflect the ∼1.9× speedup reported in the paper. Table 5 compares EARTHSIGHT-MTL against the exact solution and clairvoyant oracle lower bound, showing that EARTHSIGHT's multitask scheduling is near-optimal.

A.6.2 Main Result Figure — End-to-End Latency

The bar chart (`results/summary_plot.png`) reproduces Figure 3, showing 90th-percentile end-to-end latency across all scenario and hardware combinations. Reproduced values should be within ±10% of the values reported in the paper; minor variation arises from the stochastic nature of image capture timing and query-region sampling across simulation runs.

If only the combined (i.e., urban observation) script is ran, then only those results shall be available. The generation script is robust and will simply not generate the bars in the bar chart for simulations that are absent.

A.6.3 Table 3 — On-Board Power Consumption

Note on reproducibility. Table 3 was originally produced from natural disaster scenario runs of 6 simulated hours, a setting in which compute demand does not push satellites to their power budget limit. The artifact runs each configuration for the full 48 simulated hours, so the absolute power figures (compute seconds and percentage of generated power consumed) will differ, significantly, from the paper values. Additionally, `generate_table_3.py` reads only logs whose directory name contains `natural` (i.e. natural disaster scenario runs); reviewers who execute only the combined-scenario subset (`--combined-only`) will observe no output from this script. To reproduce Table 3, at minimum the natural disaster scenario runs must be completed.

The qualitative claim Table 3 supports—that EARTHSIGHT operates within its on-board power budget across hardware platforms—remains verifiable from the reproduced values regardless of these differences in simulation length.

B EVALUATION SCENARIOS DETAILS

In greater detail, the three scenarios we use for evaluation are below, with supporting sample images from PlanetScope. All images were retrieved from the PlanetScope image gallery: https://www.planet.com/latest-satellite-imagery-gallery/.

Natural Disaster Monitoring

This scenario provides early warning and impact assessment for events like floods, wildfires, earthquakes, volcanic eruptions, and oil spills. It monitors river water extent, fire thermal anomalies, ground deformation, ash plumes, and the spread of slicks at sea. Prioritization reflects threat urgency, with events like rapidly spreading wildfires or new volcanic eruptions demanding the most immediate attention.

A high-priority task would be triggered by an image showing a rapidly expanding wildfire front, identified by new thermal hotspots adjacent to previously unburned areas, especially when downwind of an urban center. Similarly, an image capturing a new, dense ash plume from a volcano would be flagged for immediate analysis to warn aviation and nearby populations. In contrast, a low-priority task would involve monitoring a known, contained wildfire with a stable perimeter, a steady lava flow far from infrastructure, or observing a river that is swollen but remains within its banks. This tiered approach enables efficient resource allocation that dynamically adapts to threat development.

Figure 8. A high-priority image from PlanetScope of a wildfire in California. Onboard analysis can flag new hotspots that signal the fire's imminent spread toward vulnerable areas.

Intelligence Scenario

This scenario delivers strategic and tactical awareness of military and logistical activities in geopolitically significant regions. It monitors key waterways, airbases, and land corridors, detecting anomalous concentrations of hardware and disruptions to critical infrastructure. Continuous operation uses tiered, risk-based priorities to provide timely intelligence on notable patterns for security and diplomatic awareness.

For instance, a high-priority event would be the detection of a significant military armament buildup, such as the one observed in Yelnya, Russia, which deviates from baseline activity (Fig. 9(a)). Another critical high-priority task is near-real-time damage assessment of key infrastructure, like a destroyed border crossing bridge, indicating conflict escalation or disrupted supply lines (Fig. 9(b)). Conversely, an image of a single container ship on a standard shipping route or typical vehicle traffic on an open highway would be a low-priority observation, logged but not prioritized for immediate downlink.

(a) Armament buildup in Yelnya, RU.

(b) Destroyed road at Kamaryn-Slavutych.

Figure 9. High-priority intelligence images. (a) shows a strategic buildup of military hardware, while (b) shows tactical damage to critical infrastructure.

Combined Global Cities Scenario

This scenario creates a comprehensive monitoring framework for major global cities, integrating disaster, security, and humanitarian elements tailored to city-specific risks. Monitoring adapts to geographic vulnerabilities and population density, addressing complex urban challenges by identifying anomalous patterns in infrastructure and population movement.

A high-priority event in this scenario could be the detection of significant urban flooding in a low-lying coastal city. Another could be the sudden, anomalous massing of vehicles at a key transportation hub, like the Gongabu Bus Terminal in Nepal. Such an event is ambiguous and requires immediate prioritization, as it could signal the start of a mass evacuation (a humanitarian crisis), civil unrest (a security issue), or a major transit failure. Low-priority data would include routine monitoring of city parks for seasonal changes or normal traffic levels at international airports. By fusing

Algorithm 1 Ground Station Schedule Generation

Require: $\mathcal{P}$ – satellite path, $\mathcal{Q}$ – query set, p^* – priority threshold
Ensure: $\mathcal{S}$ – task schedule, for upcoming satellite pass

Initialize schedule $\mathcal{S} \leftarrow []$
for each planned capture location loc $\in \mathcal{P}$ **do**
  Initialize applicable query set: $\mathcal{Q}_{\text{loc}} \leftarrow \emptyset$
  for each query $q \in \mathcal{Q}$ **do**
    if loc $\in$ AOI(q) **then**
      $\mathcal{Q}_{\text{loc}} \leftarrow \mathcal{Q}_{\text{loc}} \cup \{q\}$
    end if
  end for
  if $\mathcal{Q}_{\text{loc}} \neq \emptyset$ **then**
    Compute DNF condition Φ_{loc} over filters from $\mathcal{Q}_{\text{loc}}$ that yield priority $\geq p^*$
    if $\mathcal{S} \neq []$ **and** last entry in $\mathcal{S}$ has formula Φ_{loc} **then**
      Merge loc into the last schedule entry
    else
      Append $(\text{loc}, \Phi_{\text{loc}})$ to $\mathcal{S}$
    end if
  end if
end for
return $\mathcal{S}$

Algorithm 2 Satellite Scheduling Runtime

Require: F – Filter set; ϕ – DNF Boolean formula, over F; β – Lower confidence threshold; α – Upper confidence threshold

Initialize dictionary $\mathcal{E} \leftarrow \{\}$
Initialize confidence $\leftarrow C_\phi(\mathcal{E})$
while $\beta \leq$ confidence $\leq \alpha$ **do**
  $f^* \leftarrow \arg\max_{f_i \in F \setminus \mathcal{E}} U_\phi(f_i, \mathcal{E})$
  Execute f^* and let v^* be its Boolean outcome
  Update execution state: $\mathcal{E} \leftarrow \mathcal{E} \cup \{(f^*, v^*)\}$
  confidence $\leftarrow C_\phi(\mathcal{E})$
end while
return confidence

these diverse data streams, this scenario provides holistic awareness for densely populated urban centers.

Figure 10. A high-priority image showing an anomalous vehicle buildup at Gongabu Bus Terminal, Nepal. Such an event requires urgent analysis to determine its cause and potential humanitarian or security implications.

C EARTHSIGHT'S SCHEDULING ALGORITHMS

EARTHSIGHT's distributed decision platform contains two distinct components. The first component, the ground scheduler, integrates global constellation context in order to prepare a computation schedule for the satellite. The second component, the in-orbit runtime, locally adapts the ground-generated schedule to order filter execution based on resource availability. These components, taken together, achieve a high-performing distributed system that is globally-aware and locally optimal.

C.1 Ground Station Scheduler

After the look-ahead simulation is performed, the scheduler has access to the priority threshold p^* for every satellite, which specifies the minimum priority of an image that can be downlinked in the next downlink window.

When a satellite reaches range of a ground station, it requests the computation schedule between the contact and the next communication window. The ground scheduler than prepares the schedule for the satellite according to Algorithm 1. For each coordinate surrounding which an image will be taken, the algorithm identifies the relevant queries containing the coordinate and synthesizes them into a Disjunctive Normal Form boolean formula ϕ. The schedule, which is a list of ϕ over all coordinates of image capture, is then compressed according to Subsection 3.4 and delivered to the satellite.

C.2 In-orbit Satellite Scheduler

On each satellite, the in-orbit satellite runtime dictates how images are processed at a granular, model-execution ordering level. Algorithm 2 provides the pseudo-code for this filter execution engine of the satellite. The execution state $\mathcal{E}$ and confidence $\mathcal{C}_\phi$ are initialized, and the filters are greedily selected for execution based on Equation 4 until the confidence exceeds the upper threshold α or is less than the lower threshold β. The values of α and β are what enables the local adaptation of the compute schedule, as they are dynamically updated based on the availability of power resources on the satellite.

D EXTENDED EXPERIMENTS AND VISION TASK EVALUATIONS

Our experiments are conducted on a diverse set of public datasets for both image classification and semantic segmentation. A detailed summary of these datasets, including their

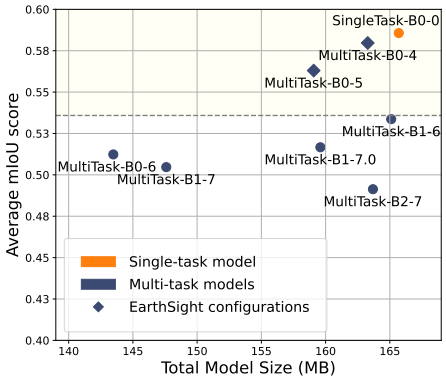

Figure 11. Segmentation: Model size vs. mean Intersection over Union (mIoU).

resolution, class count, and origin, is provided in Table 1.

Although the EARTHSIGHT pipeline is primarily designed for classification tasks that are utilized to evaluate formulas, we also performed analysis for semantic segmentation tasks. While semantic segmentation models are more complex than necessary for the requirements of prioritization, they can provide richer insights that can aid in immediate response to critical situations.

Our experimental analysis showed that for dense prediction tasks such as semantic segmentation (Figure 11), it is also viable to reduce the overall model size while maintaining comparable task prediction performance by leveraging a multi-task model paradigm. Since dense prediction requires extracting richer information from input images, employing deeper backbones than EfficientNet-B0, such as ResNet50, or utilizing vision transformer-based feature extractors may lead to more promising results.

Another way to represent the visualizations in Figure 6 is with a cumulative density function over the images, representing what fraction of the high-priority $(p > 2)$ images are downlinked by time t. See 12. The baseline (orange) appears below both EARTHSIGHTST and MT, meaning that at any given time, a greater percentage of high-value images would be downlinked by EARTHSIGHTthan the baseline approach. This is true for both Coral and Jetson configurations.

E ANALYSIS OF THE UTILITY HEURISTIC FOR FILTER ORDERING

E.1 Comparison to Exact Method and Oracle

Computing the exact solution minimizing (Eq. 3) on even one instance of ϕ is intractable for complex formulas, as the runtime is exponential in the number of filters (variables), specifically $O(3^n)$ as each filter is unevaluated, evaluated true, or evaluated false.

To study the performance of the exact solution, we artificially construct formulas from queries that have fewer than 15 filters. For simplicity, we only analyze the performance in the multi-task setting, as our single-task approach is a special case of the multi-task approach.

Lastly, we also compare to an oracle, which has access to the hypothetical results of evaluating every filter on an image, i.e., knows the ground truth. The oracle picks filters minimizing the time (actual, not expected time) to evaluate the formula.

Table 9. Average evaluation time per image across different filter ordering methods on a synthetic suite of 2473 formulas. Lower is better.

Method	Mean Time (s)	Median Time (s)
EARTHSIGHT MT	2.13	1.95
Exact Solution	1.98	1.95
Oracle	1.15	1.02

On a synthetic suite of 2473 formulas constructed by removing any filters beyond 15 from the existing formulas in our three scenarios, EARTHSIGHT Multi-task achieves an average evaluation time 2.13 seconds, the exact solution achieves an average evaluation time of 1.98 seconds, and the oracle achieves an average evaluation time of 0.85 seconds. See Table 9 The fact that filter pass probabilities are very low, most often less than 1 to 5%, contributes to the strong performance of our approximate method. The oracle result demonstrates that the stochastic nature of filter outcomes significantly increases the total compute time from a scheduling perspective.

E.2 Theoretical Analysis

The greedy scheduling policy described in Section 3.3 aims to find an approximate solution to the NP-hard Stochastic Boolean Function Evaluation (SBFE) problem stated in Equation 3. While the heuristic is not guaranteed to be optimal, primarily as it does not look ahead to future costs, its design is theoretically grounded in the property of adaptive submodularity. This provides strong justification for its quality and robustness.

The utility function U_ϕ in Equation 4 maximizes a benefit-cost ratio. The benefit component, $(1 - p_i) \cdot \text{tpr}_i \cdot n_i$, represents the expected number of DNF terms that will be correctly invalidated by running filter f_i. The quality of this heuristic can be understood by analyzing the core of this benefit term, n_i.

Theorem 1 *The benefit function representing the number of unique DNF terms covered by a set of evaluated filters is a monotone, non-decreasing, and submodular set function.*

Let F be the ground set of all filters and $\phi = \{T_1, \ldots, T_m\}$ be the set of terms in the DNF formula. Define a set function $g : 2^F \to \mathbb{Z}$ that measures the benefit of having executed a subset of filters $S \subseteq F$. Let $g(S)$ be the number of terms in

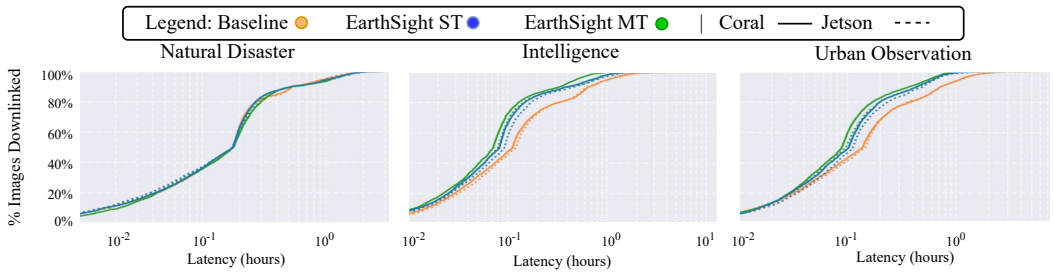

Figure 12. Distribution of avoidable image latencies for high priority ($p > 2$) images across all scenarios, measured in hours. EARTHSIGHT shows comparable or improved performance across all scenarios.

ϕ that contain at least one filter from S. Formally:

$$g(S) = \left| \bigcup_{f \in S} \{T_j \in \phi \mid f \in T_j\} \right|$$

Monotonicity. We must show that for any $A \subseteq B \subseteq F$, we have $g(A) \leq g(B)$. Let $\mathcal{T}(S) = \bigcup_{f \in S} \{T_j \in \phi \mid f \in T_j\}$ denote the set of terms covered by a set of filters S, so that $g(S) = |\mathcal{T}(S)|$. If $A \subseteq B$, then for every $f \in A$ we also have $f \in B$, so every term covered by a filter in A is also covered by a filter in B; that is, $\mathcal{T}(A) \subseteq \mathcal{T}(B)$. Therefore $g(A) = |\mathcal{T}(A)| \leq |\mathcal{T}(B)| = g(B)$, and the function is monotone non-decreasing.

Submodularity. We must show that for any subsets $A \subseteq B \subseteq F$ and any filter $f \in F \setminus B$, the following diminishing-returns inequality holds:

$$g(A \cup \{f\}) - g(A) \geq g(B \cup \{f\}) - g(B)$$

Let $\mathcal{T}_f = \{T_j \in \phi \mid f \in T_j\}$ be the set of all terms containing filter f. The marginal gain of adding f to any set S with $f \notin S$ is the number of terms in $\mathcal{T}_f$ not already covered by S:

$$g(S \cup \{f\}) - g(S) = |\mathcal{T}_f \setminus \mathcal{T}(S)|$$

Since $A \subseteq B$, monotonicity gives $\mathcal{T}(A) \subseteq \mathcal{T}(B)$. Therefore:

$$\mathcal{T}_f \setminus \mathcal{T}(B) \subseteq \mathcal{T}_f \setminus \mathcal{T}(A)$$

and taking cardinalities yields:

$$|\mathcal{T}_f \setminus \mathcal{T}(A)| \geq |\mathcal{T}_f \setminus \mathcal{T}(B)|$$

which is precisely $g(A \cup \{f\}) - g(A) \geq g(B \cup \{f\}) - g(B)$, establishing submodularity.

Implication. The submodularity of g established above justifies the design of our greedy ratio heuristic, but the correct approximation guarantee depends on how the optimization problem is stated. The satellite runtime operates under a fixed onboard power budget B, which is the binding constraint in practice (Section 3.3). We therefore state the relevant result in terms of maximization under a budget constraint.

Define $h(E)$ as the number of DNF terms whose truth value has been *determined* given execution state E—either a filter returned `False`, eliminating the term, or all filters in the term returned `True`, satisfying it. The function h is monotone and *adaptively submodular* in the sense of Golovin & Krause (Golovin & Krause, 2011): because filter outcomes are independent, conditioning on the observed state E does not introduce positive correlations between future outcomes, so the expected marginal gain of any filter is non-increasing as E grows. The satellite runtime's policy—greedily selecting the filter maximizing $\Delta h(f \mid E)/t_{\text{eff}}(f, E)$ at each step—is an instance of the adaptive greedy algorithm for maximizing a monotone adaptive submodular function subject to a cost budget. By Theorem 8 of Golovin & Krause (Golovin & Krause, 2011), this policy achieves an expected coverage of at least $(1 - 1/e) \approx 63\%$ of the coverage attained by the optimal adaptive policy within the same budget B:

$$E[h(\pi_{\text{greedy}})] \;\geq\; \left(1 - \frac{1}{e}\right) \cdot E[h(\pi^*_B)]$$

where π^*_B is the optimal policy subject to expected cost B. This provides a formal, constant-factor guarantee for Algorithm 2 under the budget-constrained interpretation that reflects the satellite's operational constraints. We note that for the complementary objective of minimizing cost to reach full term-resolution, the applicable guarantee is an $O(\log m)$ approximation (where m is the number of DNF terms) via the adaptive stochastic set-cover result of Deshpande et al. (Deshpande et al., 2016); the empirical gap of 7.5% reported in Table 5 suggests the heuristic performs substantially better than either bound requires in practice.