

Computer Vision & Features Of Celestial Bodies' Landscapes

Authors:

Matvei Sotnikov, Yurii Yesypenko, Erik Brits

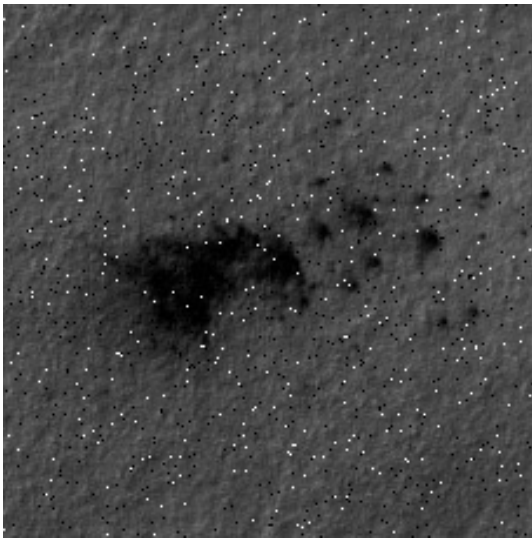


Image 1, Augmented impact ejecta with Gaussian Noise, from Mars

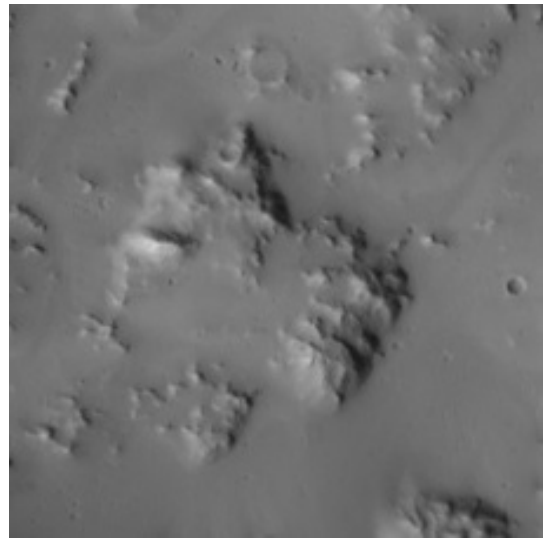


Image 2, Elevations on Mars

Abstract

We are a team of three talented developers from Lisbon. From October 4 to 5 2025, NASA held a hackathon to develop a project on space exploration, in which we participated. **Over two days**, we created, tested, and visualized a neural network based on ResNet-50, which solves one of the key tasks in the study of solid celestial bodies: the classification of landscape elements. By training the model on a NASA dataset of **more than 80,000** labeled photographs of the surface of Mars ([DOI: 10.5281/zenodo.2538136](https://doi.org/10.5281/zenodo.2538136)), we created a compact and effective neural model that predicts the features of the Martian and lunar landscapes with more than **98% accuracy**.

Based on a neural network, we have created the website “Planets-Map.select” with topographic maps from NASA of three celestial bodies: Mars, the Moon, and Vesta. Users can select any area on the map and receive predictions from the neural network about the type of landscape depicted in the selected area. **This is just one of the use cases for our neural network**: determining the type of terrain based on arbitrary input data.

Introduction

In 2025, the theme of NASA's Space Apps Challenge hackathon was “Learn, Launch, Lead.” It was quite a broad topic. After many discussions and debates, we decided on the main idea for our project: exploring the surface of Mars. We encountered a database consisting of more than 27,000 scientific records from NASA and other partner space agencies, of which more than 1,300 were directly related to Mars.

Our gaze fell upon a marked dataset of orbital images of Mars (HiRISE), created in 2019 by a group of researchers from NASA's Jet Propulsion Laboratory (JPL). However, problems were immediately detected in the dataset.

Dataset

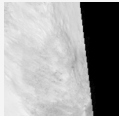

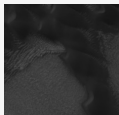
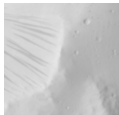
The dataset contained 10,433 landmarks extracted from 180 HiRISE photographs. Based on the original photographs, the number of elements in the dataset was increased sevenfold to 73,031 during augmentation.

The following methods were used for augmentation:

- *90 degrees clockwise rotation*
- *180 degrees clockwise rotation*
- *270 degrees clockwise rotation*
- *Horizontal flip*
- *Vertical flip*
- *Random brightness adjustment*

The dataset also contained the following naming system for surface photo labels:

Label correspondence table

Number	String	Example
0	Undefined/Other	
1	Crater	
2	Dark Dune	
3	Slope Streak	

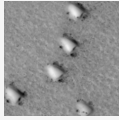
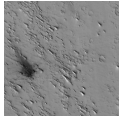


4	Bright Dune	
5	Impact Ejecta	
6	Swiss Cheese	
7	Spider	

Table 1

However, this dataset had significant issues with the distribution of instances for each class. The table below shows the distribution of the number of instances for each class and the percentage ratio:

Table of group images ratios

Class	Images	Percentage ratio
Undefined/Other	61054	83,6%
Crater	4900	6,7%
Dark Dune	1141	1,6%
Slope Streak	2331	3,2%
Bright Dune	1750	2,4%
Impact Ejecta	231	0,3%
Swiss Cheese	1148	1,5%
Spider	476	0,7%
Total	73031	100%

Table 2

Based on the data presented in Table 2, it can be seen that class 0 represents **83.6%** of the images in the entire dataset, which *is categorically unsuitable* for training. The difference between the minimum class and the second highest class is 6 percentage points, or **957%**, which also creates unrealistic conditions for the neural network training process.

Using Python, we corrected the group ratios to acceptable values by applying enhanced augmentation to the least popular groups: *Impact Ejecta*, *Spider*, *Dark Dune*, *Swiss Cheese*, and *Slope Streak* (2–7). We noticed that the NASA research team did not use the Gaussian noise augmentation method in their augmentation process. We used this method for our own re-

augmentation, **doubling** the number of images in the *Slope Streak* class, **quadrupling** the number in the *Dark Dune*, *Bright Dune*, and *Swiss Cheese* classes, and **octupling** the number in the *Spider* and *Impact Ejecta* classes.

In addition, after several training tests with the dataset, we concluded that we needed to remove the 0 class due to the large amount of noise. Since all other objects are in class 0, there are no discernible relationships between them, and therefore it was very difficult for neural networks, including those based on ResNet50, to identify patterns even after 30 training epochs.

Training results

We trained a ResNet-50 classifier on 7 Mars surface categories after removing the “undefined/Other” group. The dataset contained 21,392 images (classes moderately imbalanced; max/min count ≈ 2.65). Images were resized to 224×224 and mapped to three channels (replicated from L). A two-stage protocol was used:

Stage 1 (linear probe). The backbone was frozen and only the 7-way linear head was optimized. Validation macro-F1 rose quickly from **0.925** to **0.952** over three epochs while the validation loss decreased (0.482 \rightarrow 0.425). This indicates that neural network features already separate these textures reasonably well.

Stage 2 (full fine-tune). We unfroze the network and fine-tuned end-to-end with mixed precision. Validation macro-F1 improved further to **0.993** on the first epoch and peaked at **0.9986** by epoch S2-04, with validation loss dropping to **0.264**. A transient dip at S2-03 (F1 = 0.990, val loss \uparrow) was corrected on the next epoch, after which the best checkpoint was saved.

Representative logs (val macro-F1 / val loss):

- S1-01: **0.925** / 0.482
- S1-03: **0.952** / 0.425
- S2-01: **0.993** / 0.294
- S2-02: **0.996** / 0.273
- S2-03: **0.990** / 0.285
- **S2-04: 0.9986 / 0.264 (best)**

Training dynamics. Train and validation losses decreased together during both stages; no collapse to a single class was observed. Minority classes received stronger data augmentation, which, together with the two-stage schedule, likely prevented overfitting despite the imbalance.

Caveats and checks. Because validation performance is very high, we recommend (i) confirming that no near-duplicate tiles or scene siblings appear across splits (grouped split by scene/strip if available) and (ii) reporting per-class precision/recall and the confusion matrix on the held-out test set. These guard against optimistic estimates due to leakage or class drift.

Inference and abstention. Since “other” was excluded from training, deployment should use an abstention rule: apply temperature scaling and reject to class 0 when $\max \text{softmax} < \tau$. The threshold τ can be tuned on validation using a pool of “other” images.

Takeaway. A simple linear probe already reached ~ 0.95 macro-F1; end-to-end fine-tuning pushed performance to **0.9986** on validation. Assuming clean splits, the model appears to learn highly discriminative representations for the seven target morphologies and is ready for test evaluation and calibration.

Practical Application

We deployed the seven-class ResNet-50 as a Dockerized Python service (FastAPI + PyTorch) behind **planets-map.select** (domain via Porkbun). Unlike tile-pulling backends, our server accepts **image uploads** from the client and returns a calibrated probability vector over terrain classes.

System overview. The frontend renders NASA basemaps for **Moon, Mars, and Vesta**. Users pan/zoom, draw a region of interest (ROI), and the client **crops that ROI client-side** to a PNG/JPEG and submits it directly to the backend (multipart/form-data), optionally tagging the body/layer as metadata. This avoids server-side tile orchestration and keeps latency predictable.

Data flow.

1. **ROI capture on the client.** The browser crops the selected map region into an image (converts to grayscale inputs) and sends `{file}`.
2. **Preprocessing on the server.** The backend converts grayscale inputs to three channels (L \rightarrow RGB replication), resizes to 224 \times 224, and applies the exact normalization used at training.
3. **Inference.** The image is batched (size 1 for single requests) and run through the **7-way head** with mixed-precision when supported by the device.
4. **Calibration & abstention.** Probabilities are temperature-scaled; if the maximum class probability falls below a tuned threshold, the service returns **“unknown/other”** instead of forcing a label.
5. **Response.** The API returns the full probability vector, the top-1 class, and (optionally) top-k details for UI overlays.

API surface. A single endpoint `/api/v1/classify/image` accepts `multipart/form-data` with `image` (PNG/JPEG). The response is JSON: `{probs: [p1..p7]}`. This design keeps the backend stateless and easy to scale.

Operational notes. Server performance is dominated by preprocessing and model forward; caching is unnecessary because images are user-specific crops. Consistency with the training pipeline (channel handling, normalization, resolution) is critical to maintain the observed validation performance.

Conclusion

What it does / how it works. We trained a seven-class ResNet-50 to recognize common geomorphological patterns on airless bodies. The model was optimized in two stages (linear probe \rightarrow full fine-tune) and then deployed behind a FastAPI service packaged in Docker. The web app renders NASA basemaps (Moon, Mars, Vesta); a user draws a region of interest, the client crops it to an image, and the backend returns a calibrated probability vector over the seven classes with an abstention option for “unknown.”

Benefits. The system turns raw map exploration into immediate, quantitative feedback. It supports triage at scale (rapidly scanning large scenes), improves consistency versus ad-hoc human labeling, and exposes full probabilities so scientists can set thresholds appropriate to their use case. Because we separated “Undefined/Other” from training and handle it at inference via confidence thresholds, the model behaves sensibly on out-of-distribution inputs rather than forcing a guess.

Intended impact. Our goal is to shorten the loop between data browsing and scientific insight. The tool should help planetary scientists and students find candidate sites (e.g., dunes, Swiss-cheese terrain, spiders) faster, enable citizen-science workflows, and provide a reproducible baseline for future comparative studies across instruments and bodies.

Tools, languages, hardware, software.

- **Languages/frameworks:** Python, PyTorch/torchvision, scikit-learn (metrics), FastAPI, React JavaScript.
- **Deployment:** Docker, REST JSON API, temperature scaling for calibration.
- **Frontend:** Web app with client-side ROI cropping; NASA map layers as basemaps.
- **Hardware:** Training on NVIDIA T4 (mixed precision); local inference tested on Apple M2 (MPS).
- **Ops:** Reproducible transforms and weights, checkpointed best model, stateless API.

What’s creative about it. Two decisions made the system both accurate and usable in two hackathon days: (i) reframing the noisy “Undefined/Other” bucket as an **open-set** problem with abstention instead of training it as a class, and (ii) a **stage-wise** training schedule that leverages strong pretrained features before gentle end-to-end fine-tuning. On the product side, accepting **direct image uploads** (rather than server-side tiling) kept the backend simple and responsive while still mirroring the exact training pipeline (grayscale→RGB replication, normalization, 224×224).

Factors we considered. We explicitly addressed class imbalance (targeted augmentation, balanced sampling), the risk of split leakage (scene-level separation), calibration and thresholding for abstention, latency vs. accuracy trade-offs (batching + AMP), and reproducibility (Dockerized environment and fixed seeds). We also kept the interface transparent by returning the full probability vector, enabling downstream thresholding and uncertainty-aware use.

In sum, the project couples a validated classifier with a minimal, reliable serving stack. It delivers near-instant landscape predictions on planetary maps, with the right safeguards (calibration, abstention, clean splits) to make those predictions actionable.