Lock N Code:
Dwarkadas J. Sanghvi College of Engineering:

Team Members Details:
1. Abhishek Pandey
2. Ansh Shah
3. Ashish Maurya
4. Ayush Mutha

Detailed Solution and Approach (250-300 words):

To detect and define craters and boulders from OHRC data in PDS4 format, we start by using a library like pdr to read and decode the PDS4 files, extracting the image data as a NumPy array. We then divide the large image into smaller, manageable tiles by calculating the optimal number of tiles based on a specific tile size (e.g., 1024x1024 pixels) and overlap to ensure comprehensive coverage. We apply data augmentation techniques such as rotation, flipping, and brightness adjustments to increase dataset diversity, and manually annotate a subset of tiles to create a ground truth dataset with polygonal boundaries. We choose the Mask R-CNN model for its instance segmentation capabilities, customizing it by modifying the classifier and mask predictor layers, and train it on the prepared dataset using appropriate loss functions (binary cross-entropy for masks, smooth L1 for bounding boxes) and the Adam optimizer with a learning rate scheduler. During model inference, we process each image tile to detect craters and boulders, applying non-maximum suppression to refine object boundaries and converting segmentation masks into polygonal boundaries. We combine the results from all tiles, ensuring consistent detection across tile boundaries. Finally, we calculate the selenographic coordinates and diameters of each detected crater and boulder, and convert the refined polygonal boundaries into shapefiles (e.g., Shapefile or GeoJSON) with relevant metadata, providing detailed shape-size information and locations of craters and boulders on the OHRC images.

Tools and Technology Used (50 words):

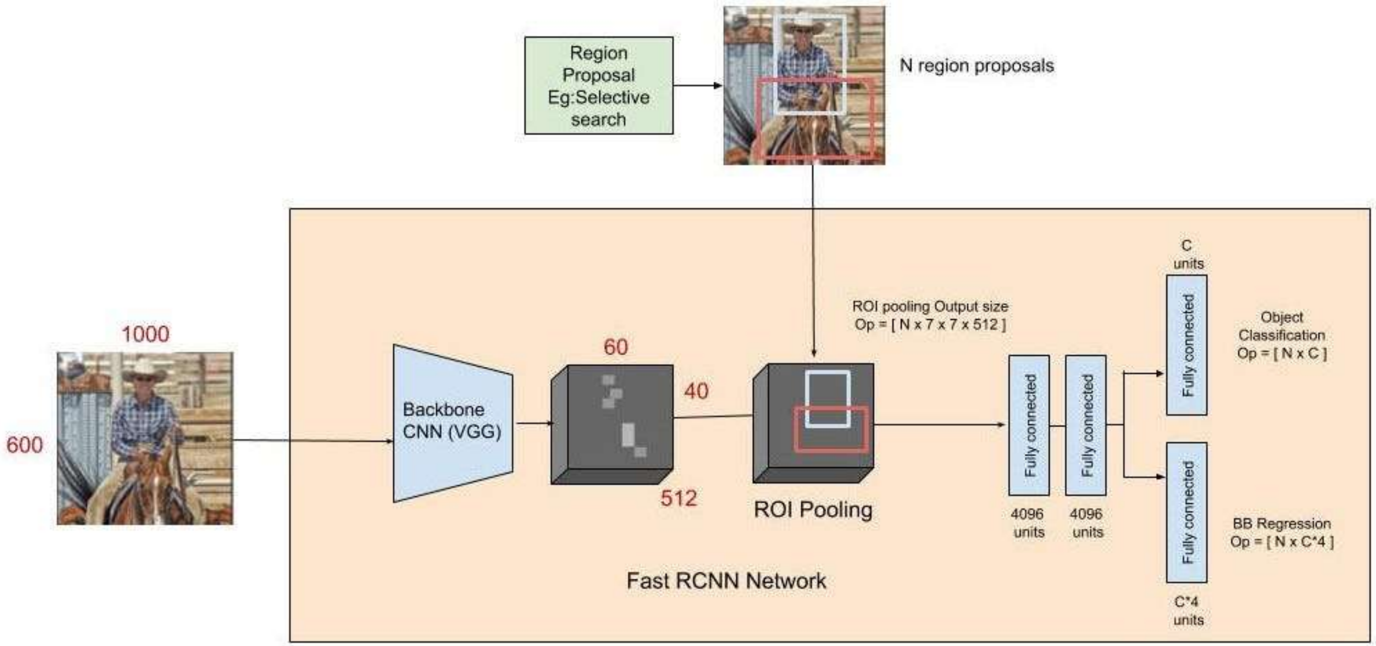
- TensorFlow and tensorflow.keras: Frameworks for building and training machine learning models.
- tensorflow-hub: Provides pre-trained model components.
- opencv-python-headless: For image processing.
- geopandas and shapely: Handle geospatial data and shapes.
- matplotlib: For plotting.
- pdr: Reads PDS4 data.
- Next.js: Manages frontend and backend, with caching for efficiency.

Opportunity should be able to explain the following:

- How different is it from any of the other existing ideas?
- How will it be able to solve the problem?
- USP of the proposed solution

Efficient Tiling and Overlap: Divides large images into overlapping tiles to ensure comprehensive coverage and avoid missing features at tile boundaries .Advanced Mask R-CNN Model: Utilizes Mask R-CNN with TensorFlow's Object Detection API for precise instance segmentation and detection. Customized Data Augmentation and Annotation: Applies extensive data augmentation and manual annotation to create a robust ground truth dataset. Refinement with Non-Maximum Suppression: Uses non-maximum suppression to refine object boundaries and reduce false positives. Geospatial Data Conversion: Converts refined polygonal boundaries into shapefiles or GeoJSON for easy integration with GIS applications. Complete Workflow Integration: Provides a full pipeline from image extraction to geospatial output, covering all stages of processing. The USP of this approach is its holistic integration of overlapping image tiling, advanced Mask R-CNN segmentation, and precise geospatial output, ensuring thorough and accurate detection of craters and boulders. This comprehensive pipeline stands out by delivering detailed and actionable geospatial data from high-resolution planetary imagery.

Proposed architecture/user diagram



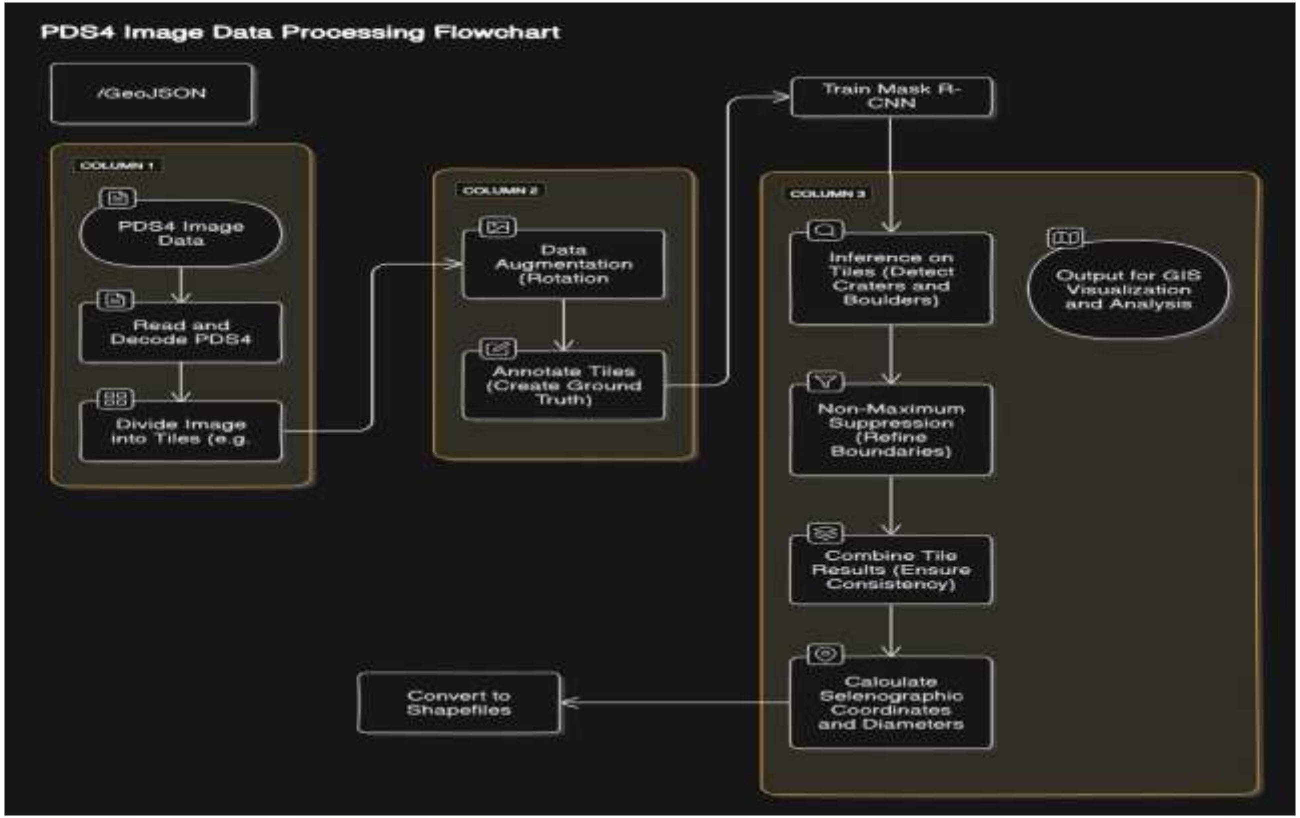
List of features offered by the solution:

It is always better to add a few visual representations (drawings/sketches/illustrations etc.) to your presentation, it adds to the power through which it reaches the audience.

1. High-Resolution Image Handling
2. Advanced Segmentation Model
3. Comprehensive Data Augmentation
4. Custom Annotation Process
5. Non-Maximum Suppression Refinement
6. Geospatial Data Conversion
7. Complete Workflow Integration
8. Selenographic Coordinate Calculation

Process flow diagram or Use-case diagram

•


PDS4 Image Data Processing Flowchart
/GeoJSON
COLUMN 1
PDS4 Image Data
Read and Decode PDS4
Divide Image into Tiles (e.g.
COLUMN 2
Data Augmentation (Rotation
Annotate Tiles (Create Ground Truth)
Train Mask R-CNN
COLUMN 3
Inference on Tiles (Detect Craters and Boulders)
Non-Maximum Suppression (Refine Boundaries)
Combine Tile Results (Ensure Consistency)
Calculate Selenographic Coordinates and Diameters
Output for GIS Visualization and Analysis
Convert to Shapefiles

Solution Brief (Overall):

To detect and define craters and boulders from OHRC PDS4 data, we use the pdr library to decode PDS4 files and extract image data as NumPy arrays. We tile the images, apply data augmentation, and manually annotate a subset. Using a customized Mask R-CNN model, we train on this dataset to segment craters and boulders, applying non-maximum suppression to refine boundaries. Results are merged, and detected features are converted into shapefiles or GeoJSON with selenographic coordinates and diameters for detailed spatial information.

BHARATIYA ANTARIKSH HACKATHON 2024

Innovation partner H2S

इसरो isro

THANK YOU