# 🌕 Lunar Crater Detection using YOLOv8 & DETR Ensemble

## 1. Introduction 📃

Accurate detection of lunar surface features such as craters and boulders plays a pivotal role in space exploration and rover navigation. Craters, due to their varying scales and irregular textures, present unique challenges in automated detection. Our team, *Craternauts*, addresses this by using an **ensemble of deep learning models (YOLOv8 + DETR)** to enhance robustness and accuracy in crater detection.

Tools/libraries used: YOLOv8, DETR, PyTorch, HuggingFace, Ultralytics, OpenCV, Streamlit, Matplotlib, Numpy, Pandas

## 2. Approach 🔍

Our solution uses a **hybrid ensemble model** combining the strengths of:

- **YOLOv8**: Fast and reliable object detection for real-time predictions.
- **DETR (DEtection TRansformer)**: A transformer-based model offering improved detection of irregular or less distinct features.

We believed that combining their outputs via **Non-Maximum Suppression (NMS)** would lead to better detection accuracy, particularly in edge cases where either model might fail individually.

The overall pipeline includes:

- Training YOLOv8 on labelled lunar imagery.
- Loading pretrained DETR model (with HuggingFace).
- Performing inference with both models.
- Merging results using NMS to resolve overlapping predictions.
- Optionally visualizing results via a **Streamlit GUI app**.

🛠️ Challenges Faced During Training:

- DETR model is heavy → used pre-trained version

- YOLOv8 took long to train → reduced epochs and batch size to make it feasible within compute limits

## 3. Implementation 🛠️

We structured the project as follows:

- train.py: Trains YOLOv8 on the dataset using Ultralytics API. It automatically creates a dataset.yaml if not present.
- detr_ensemble.py: Defines detection functions for YOLO, DETR, and the ensemble. Automatically handles fallback to YOLO if DETR fails to load.
- inference.py: Performs batch inference over test images, saves results in YOLO format.
- app.py: A Streamlit GUI that allows users to upload images and view predicted bounding boxes interactively.

We trained our YOLOv8 model using a custom dataset of lunar images and labelled craters. DETR was loaded from HuggingFace's pre-trained checkpoint (converted to local folder form for offline use).

---

## 4. Experimentation and Innovations 🛸

### ✅ Ensemble Strategy

We used a **late-fusion ensemble** where DETR and YOLO predictions were merged using IoU-based NMS. This approach allowed us to:

- Improve precision where YOLO produced false positives.
- Enhance recall where DETR captured subtle crater features missed by YOLO.

### ✅ GUI for Interpretation

We built a **Streamlit GUI app** to make model usage more intuitive. This helps users visualize predictions live and better interpret model behaviour. UI Features:

- Crater count display

- Confidence threshold slider

- Crater diameter information

- Circularity score of detected craters

- Crater density heatmap (visual representation)

---

## 5. Insights and Real-world Use Cases 🌍

📌 **Observations:**

- DETR detected **larger and blurred craters** better, especially at lower resolutions.
- YOLOv8 was more confident and faster, performing best on **well-contrasted small craters**.
- The ensemble improved both **consistency** and **coverage** across varied terrain types.

🚀 **Applications:**

- **Autonomous lunar rovers**: Safer path planning around craters and boulders.
- **Satellite monitoring**: Studying crater formation or impacts over time.
- **Simulation environments**: Improving realism by automatically labelling terrain in 3D moon simulation platforms.

---

By Sameer Chakrawarti, Warren Sequeira