Surface Crack Detection System

A Summer Internship Company of

APSAC (Andhra Pradesh Space Applications Center), Vijayawada

Keerthana Panchumarthi
AP23110011620
B.Tech CSE 2027

Project Title :

*“**Crack Detection in UAV Imagery Using YOLO-based Object Detection with Tiling and Image Reconstruction.”***

Why It Matters :

Early crack detection in roads, bridges, and buildings is crucial to prevent structural failures, reduce maintenance costs, and ensure public safety. UAV-based automated detection enables faster, large-scale inspection with minimal human risk.

Progress Update

Observation:
Worked on UAV-based crack detection. Cracks were mostly small and centered, but appeared across images; noisy inputs made annotation and detection difficult.

Process:
Filtered dataset, Tiled images (640×640), annotated 1300+ tiles, applied augmentation and preprocessing, used Roboflow, and trained YOLOv8/v11.

Outcome:
Achieved 66% mAP with a clean, 3000+ image dataset and improved model accuracy.

Tech Stack:

Languages: Python
Frameworks/Models: YOLOv8, YOLOv11, Roboflow Hosted MOdels
Tools: Roboflow, OpenCV
Techniques: Tiling, Data Augmentation, NMS, IoU, Image Filtering (by size)

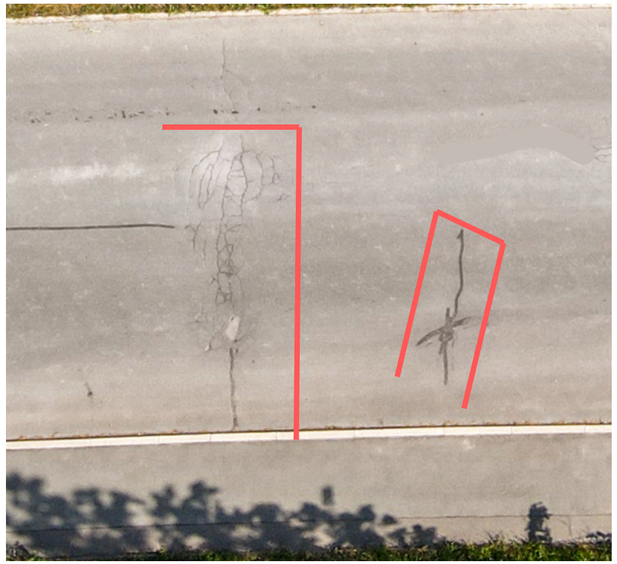
Methodology

Data Preparation

1. Image Collection
- Collected high-resolution UAV images (~5K resolution) containing visible surface cracks.

2. Data Cleaning & Preprocessing
- Removed noisy and irrelevant images (e.g., rivers, dense greenery) by checking file sizes (e.g., rivers <100 kB, trees >196 kB).
- Removed duplicates and faulty images from the original dataset.

3. Tiling (640×640 Patches)
- Each high-resolution image was split into multiple 640×640 patches.
- The object detection model (YOLO) is trained only on these tiled patches.
- This ensures uniform input size and improved crack localization.



Labeling & Augmentation

(4.) Handling Split Cracks Across Tiles

- Sometimes a single crack appears in two or more adjacent tiles, leading to duplicate detections.
- This is handled during post-processing using Non-Max Suppression (NMS) and Intersection over Union (IoU).

(5.) Image Annotation

- Used Roboflow for efficient annotation after switching from LabelMe (which had limitations).
- Annotated over 1300+ tiles, leaving a few unannotated for background learning/generalization.

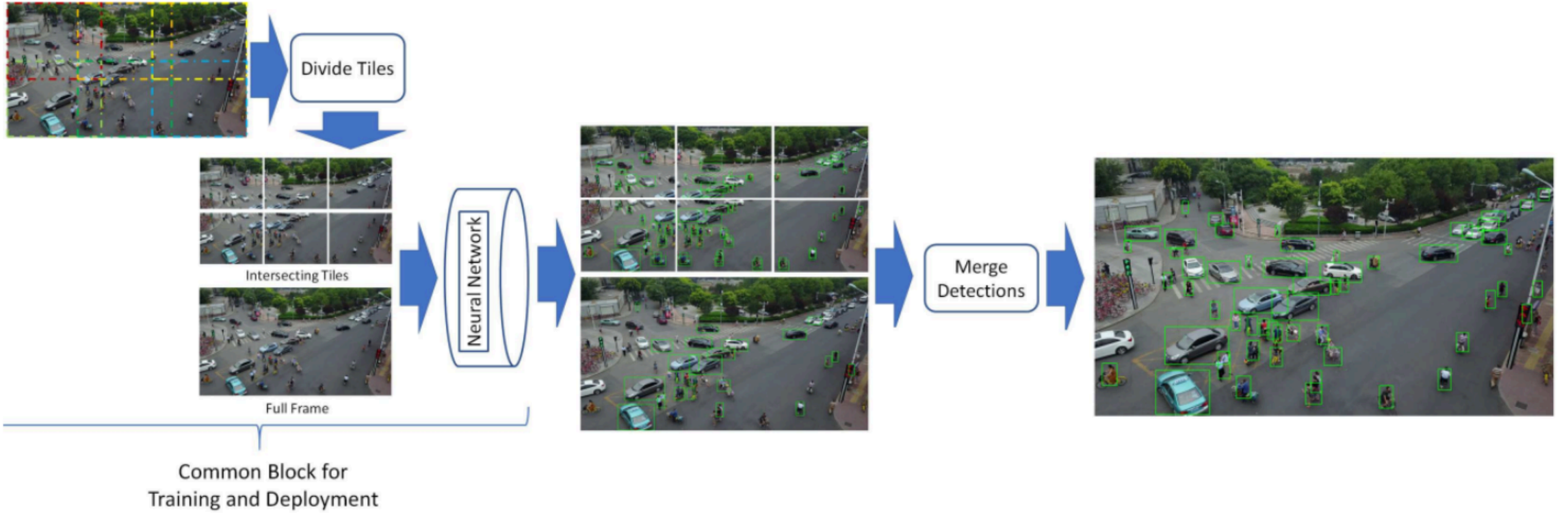
(6.) Data Augmentation

- Applied augmentations like cropping, blurring, rotation, and flips to increase dataset diversity.
- Final training set: ~5000+ images including augmentations.


(a)


(b)


(c)


Divide Tiles
Intersecting Tiles
Full Frame
Neural Network
Merge Detections
Common Block for
Training and Deployment

Training, Evaluation, and Output

(7.) Model Training

- Trained models: YOLOv8, YOLOv11, and Roboflow's hosted model.
- Trained for 150+ epochs using only the tiled images (640×640 patches).
- Images and labels were fed directly into the YOLO pipeline.

(8.) Model Evaluation

- Tracked accuracy metrics including:

✓ mAP (Mean Average Precision): 66.0%

✓ Precision: 74.2%

✓ Recall: 52.3%

- Improved over time with more data, augmentation, and better annotations.

(9.) Post-processing & Stitching

- After inference, predictions (bounding boxes) from individual 640×640 tiles were stitched back to reconstruct the full image.
- Used cv2 and custom logic to merge bounding boxes using NMS and IoU thresholds to avoid duplicate crack detections.

(10.) Model Selection & Optimization

- Compared YOLO models based on their performance.
- Confirmed no overfitting by analyzing validation metrics (precision/recall consistency).
- Final model selected based on balance of speed, accuracy, and robustness to background noise.

The objective is to detect cracks in high-resolution UAV imagery using object detection models like YOLO, while handling challenges in dataset quality, annotation, and model generalization.

Problem 1: Noisy and Irrelevant UAV Data

Observing Process 1:
UAV images contained rivers, dense greenery, and other textures unrelated to crack detection.

- Irrelevant tiles increased false positives.
- Duplicates and noise bloated the dataset.

Solution:
Removed duplicates and filtered tiles by file size using a custom script.
(e.g., rivers <100 kB, trees >196 kB).

Problem 2: Crack Splitting Across Tiles

Observing Process 2:

Tiling large 5K-resolution images into 640×640 patches often caused a single crack to appear in multiple tiles.

- Same crack annotated twice across tiles.
- Model interpreted it as two different cracks.

Solution:

Used Non-Max Suppression (NMS) and Intersection over Union (IoU) to reduce overlap issues and planned to stitch predictions post-inference.

Problem 3: Annotation & Tool Limitations

Observing Process 3:

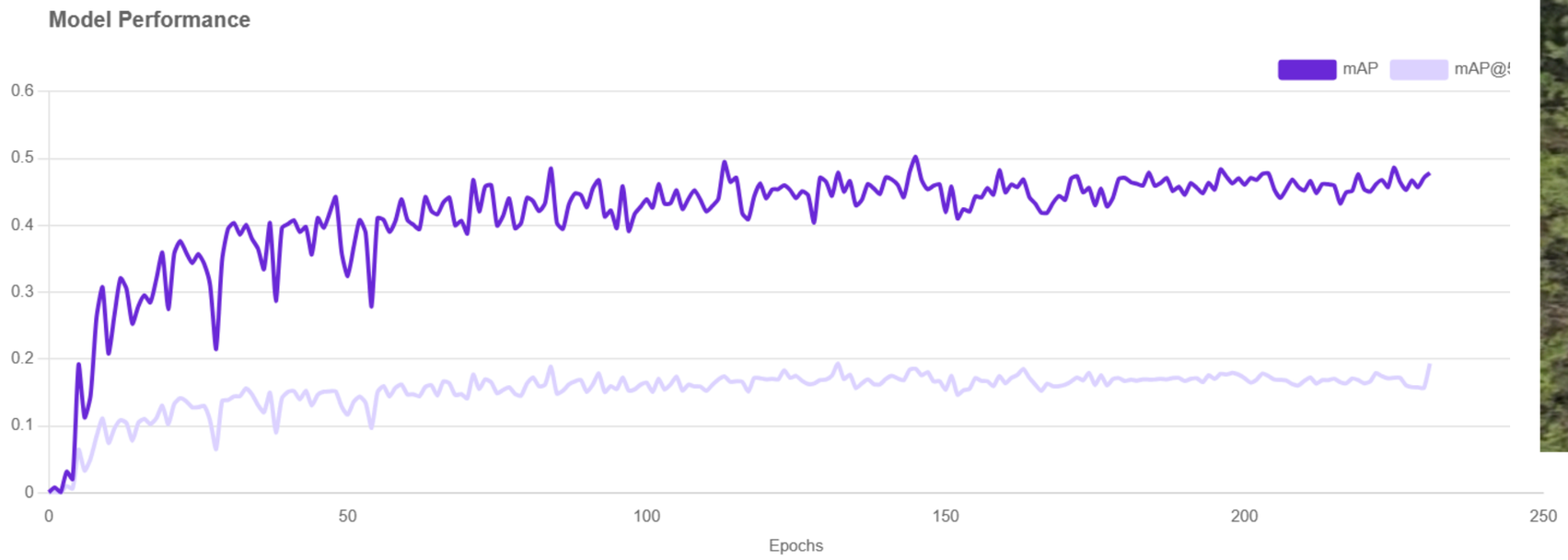
Used LabelMe initially, which had restricted access to features.

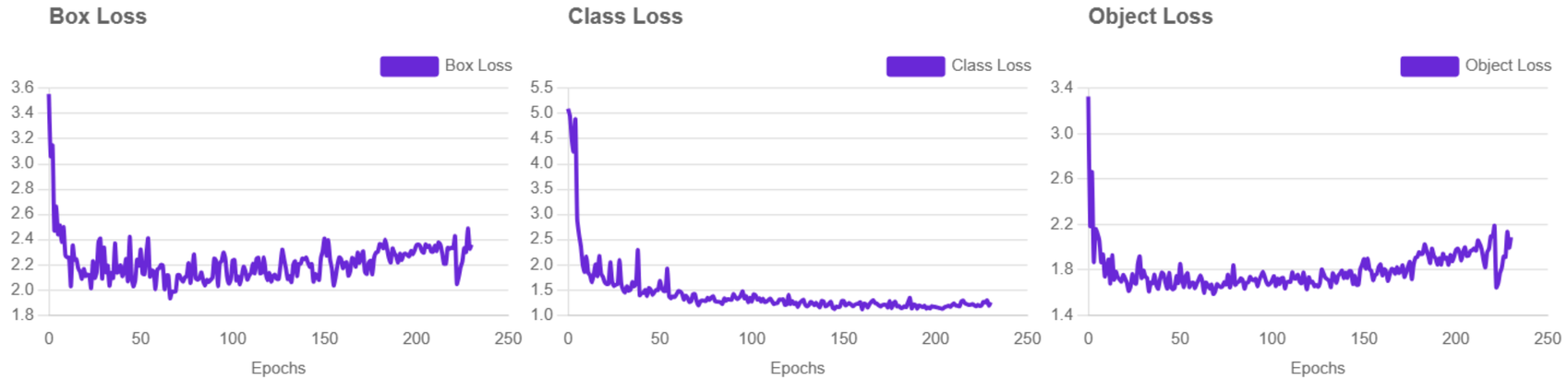
- Time-consuming annotation process.
- No built-in augmentation or training support.

Solution:

Switched to Roboflow, which provided a full pipeline including annotation, augmentation, and model training.

Results so far





Results so far

CrackDetection02 10 ID: crackdetection02/10	7/10/25 2:26 PM	mAP@50 60.8% Precision 63.7% Recall 53.8%	YOLOv11 Object Detection (Fast)
CrackDetection02 9 ID: crackdetection02/9	7/8/25 7:08 PM	mAP@50 58.3% Precision 62.5% Recall 52.8%	YOLOv11 Object Detection (Fast)
CrackDetection02 8 ID: crackdetection02/8	7/5/25 3:55 PM	mAP@50 56.6% Precision 60.7% Recall 51.3%	YOLOv11 Object Detection (Fast)
CrackDetection02 4 ID: crackdetection02/4	6/28/25 10:47 PM	mAP@50 61.4% Precision 58.9% Recall 58.2%	YOLOv11 Object Detection (Extra Large)
CrackDetection02 6 ID: crackdetection02/6	6/24/25 1:08 PM	mAP@50 61.2% Precision 57.9% Recall 56.5%	Roboflow 3.0 Object Detection (Fast)
CrackDetection02 2 ID: crackdetection02/2	6/20/25 11:13 PM	mAP@50 64.2% Precision 64.8% Recall 59.1%	Roboflow 3.0 Object Detection (Fast)
CrackDetection02 1 ID: crackdetection02/1	6/20/25 7:48 PM	mAP@50 66.0% Precision 74.2% Recall 52.3%	YOLOv11 Object Detection (Fast)

Results so far

4 objects detected

Confidence Threshold: **50%**

0% 100%

Overlap Threshold: 50

0% 100%

Opacity Threshold: 75

0% 100%

Label Display Mode:

Draw Confidence


```
{
  "predictions": [
    {
      "x": 361,
      "y": 1188,
      "width": 90,
      "height": 74,
      "confidence": 0.708,
      "class": "Cracks",
      "class_id": 0,
      "detection_id": "c7e24e0
    },
    {
      "x": 537.5,
      "y": 381,
      "width": 111,
      "height": 204
```


Copy

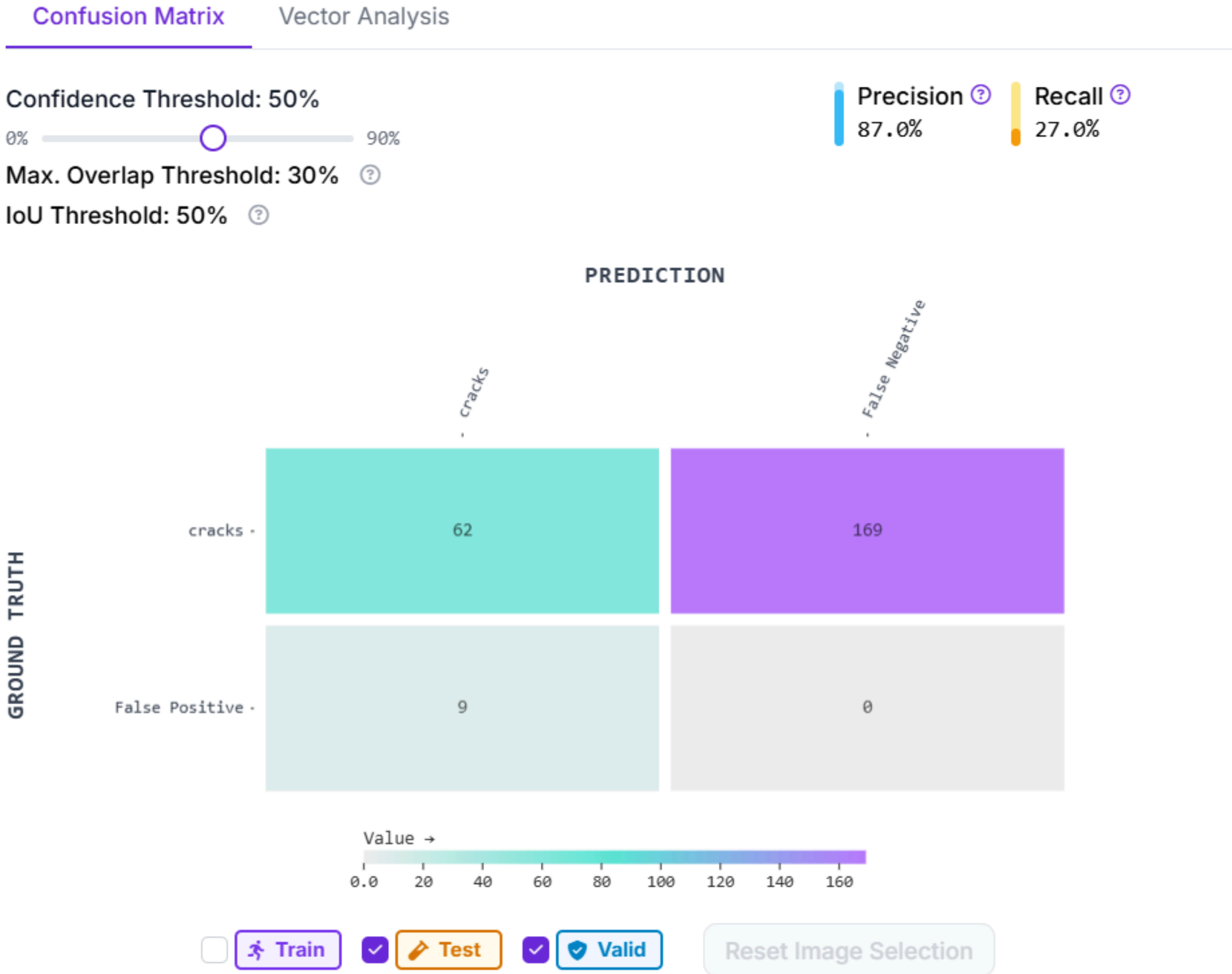
Download Weights

Deploy Mod

Metrics

mAP@50	Precision	Recall
48.0%	67.3%	43.7%

Results so far



Results so far

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS

```
Final Detections (after NMS):
Box: (401, 3076), (640, 3524)
Box: (1536, 3), (1740, 462)
Box: (772, 2658), (1121, 2888)
Box: (622, 3582), (912, 3711)
Box: (1025, 1315), (1152, 1635)
Box: (1209, 512), (1531, 1113)
Box: (1210, 1024), (1410, 1353)
Box: (1176, 1580), (1428, 2174)

✅ Saved result to: C:\Users\satya\OneDrive\Desktop\keerthana\DL_APSAC\detections_output01.jpg
PS C:\Users\satya\OneDrive\Desktop\keerthana\DL_APSAC>
```

Conclusion

Achieved 66% mAP using YOLO-based crack detection with tiled UAV images and preprocessing.
We can improve the accuracy with more data points of different surfaces/structures.

thank you,

Keerthana