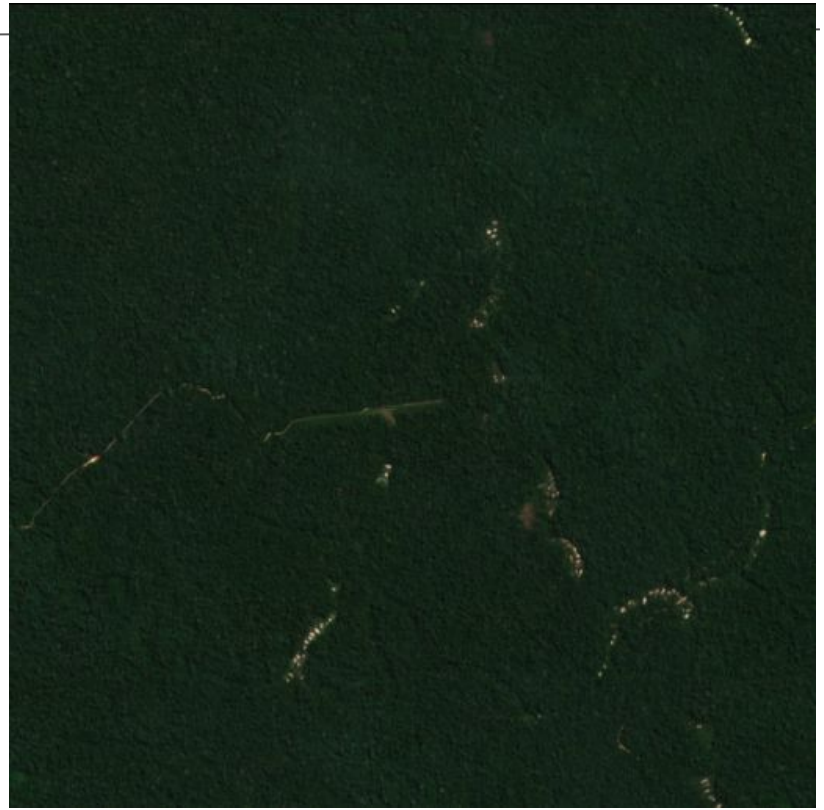
GeoAI Amazon Basin Secret Runway Detection

Detecting hidden airstrips in the Amazon rainforest using YOLOv8

By Team: Prof. Rajasekhar M, S P Suhas, Ishita Dalela, Adyansh Aggarwal, Khushi Jayaprakash
Mentor: Mr. Kodandaram Ranganath

Problem Statement

- Illegal mining, drug trafficking, and logging operations in the Amazon often use hidden runways under dense forest canopy.
- Detecting these airstrips early can help enforcement agencies and conservation groups act before damage is done.
- Current methods are prone to high false positives and low generalization to new locations.



The Original Team's Work

Approach:

- Combined **data preprocessing**, **deep learning**, and **geospatial analysis** for runway detection.
- Developed in **Google Colab**, with **Google Drive** for storage and **GitHub** for version control.
- Used **Google Earth Engine (GEE)** to download Sentinel-1 SAR (VV & VH bands) and Sentinel-2 MSI imagery (6 bands: visible, NIR, SWIR).

Data Pipeline:

- **Bands Selection:** Merged Sentinel-1 & 2 data to produce 9-band images.
- **Cropping:** 512×512 px tiles at 10 m/px for training; inference matched AOI bounding boxes.
- **Normalization:** RGB bands → ImageNet values; others → dataset mean/std.
- **Augmentation:** Random crops to 224×224, flips, rotations.

The Original Team's Work (continued)

Model Architecture:

- **U-Net** with **ResNet50 encoder**, pre-trained on ImageNet for RGB bands.
- Support for additional bands with separate learning rates.

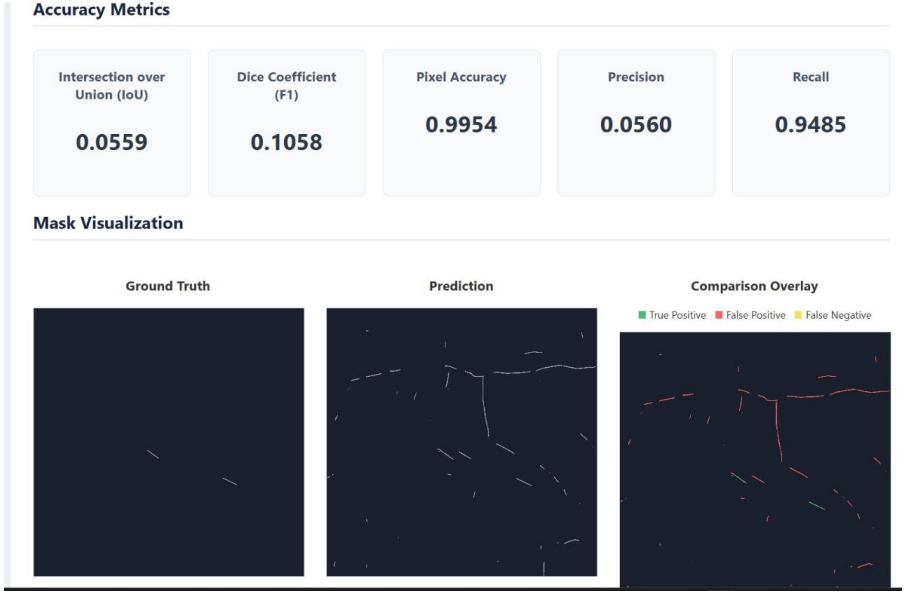
Workflow:

1. Download & visualize Sentinel data.
2. Clean GeoTIFFs and remove NaN borders.
3. Generate runway masks from shapefiles.
4. Train model in *zindi_airstrip_training.ipynb*.
5. Run inference in *zindi_airstrip_prediction.ipynb*.
6. Post-process predictions — remove roads (via OSM data), apply buffer zones.

Reference: https://github.com/liangchow/zindi-amazon-secret-runway

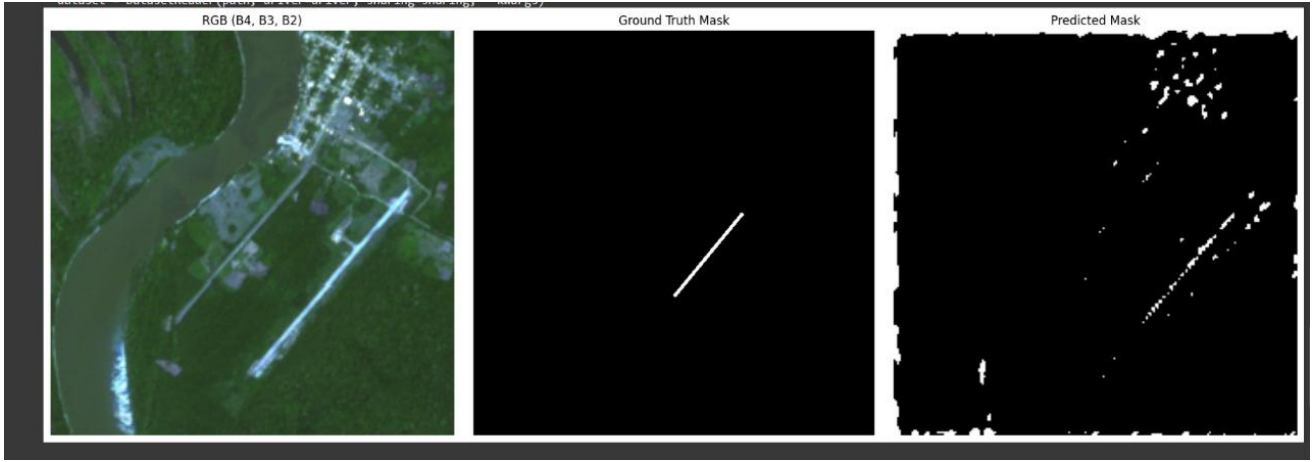
Issues with the Original Approach

- **Poor Accuracy**: Missed several real runways, especially those partially obscured by trees.
- **High False Positives**: Confused riverbanks, roads, and clearings for runways.
- **No Localization**: Patch classification didn't provide precise runway boundaries.
- **Limited Generalization**: Performed poorly on unseen geographic regions.



How We Debunked the Model

- Re-ran their code on new, unseen data (our test set).
- Compared predictions with ground truth masks annotated manually — precision dropped significantly.
- Visual inspection revealed multiple false detections in non-runway areas.
- Conclusion: Model needed both **better localization** and **more robust feature extraction**.



Our Idea

Novelty: Replace patch-based CNN with **YOLOv8 object detection** for precise localization.

Why YOLOv8?

- State-of-the-art in real-time object detection.
- Detects runways as bounding boxes, reducing background confusion.
- Better at learning elongated, narrow structures.

Additional Enhancements:

- Data augmentation targeting Amazon canopy features.
- Class balancing to reduce overfitting to majority background class.

YOLOv8 Architecture & Training Pipeline

YOLOv8 Architecture

Training Pipeline

Data Flow & Key Metrics

Dataset Structure:
- images/train_split/ (80% of data)
- images/val/ (20% of data)
- labels/train_split/ (.txt annotations)
- labels/val/ (.txt annotations)

Training Parameters:
- Epochs: 50
- Image Size: 640×640
- Device: CUDA GPU
- Model: YOLOv8n

Performance Metrics:
- mAP@0.5: 0.926 (92.6%)
- High Precision & Recall
- Class: 'airstrip' detection
- Confidence-based predictions

Model Outputs:
- Bounding box coordinates
- Class probability scores
- Object confidence scores
- Visualization with plots

File Management:
- Google Drive integration
- Timestamped result folders
- Weight file saving
- Performance metrics logs

Key Architecture Features

Backbone (C2f + SPP): Efficient feature extraction with cross-stage connections

Neck (Feature Fusion): Multi-scale feature integration via upsampling & concatenation

Decoupled Head: Separate classification and detection tasks for improved performance

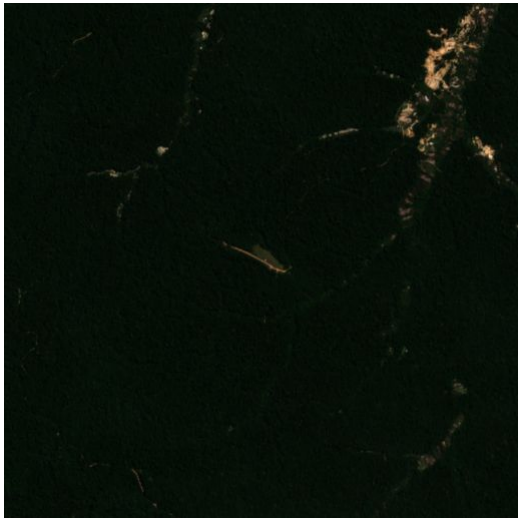
Pipeline Benefits

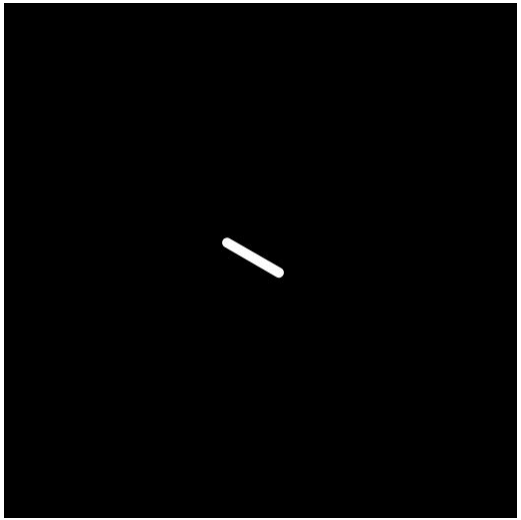
✓ Automated train/validation split prevents overfitting
✓ YAML configuration ensures reproducible experiments
✓ GPU acceleration for fast training (50 epochs)
✓ High accuracy achieved (92.6% mAP) for airstrip detection

How It Works

1. Data Preparation – *Create_yolo_bounds.ipynb*

- Converts binary mask annotations into YOLOv8-compatible bounding boxes.
- Matches mask files with corresponding satellite image chips.
- Extracts bounding boxes from non-zero mask areas, with optional padding.
- Saves:
  - Processed images (.png)
  - YOLO labels (.txt)
  - Empty .txt for false samples (no runways)
- Output structured for direct YOLO training.

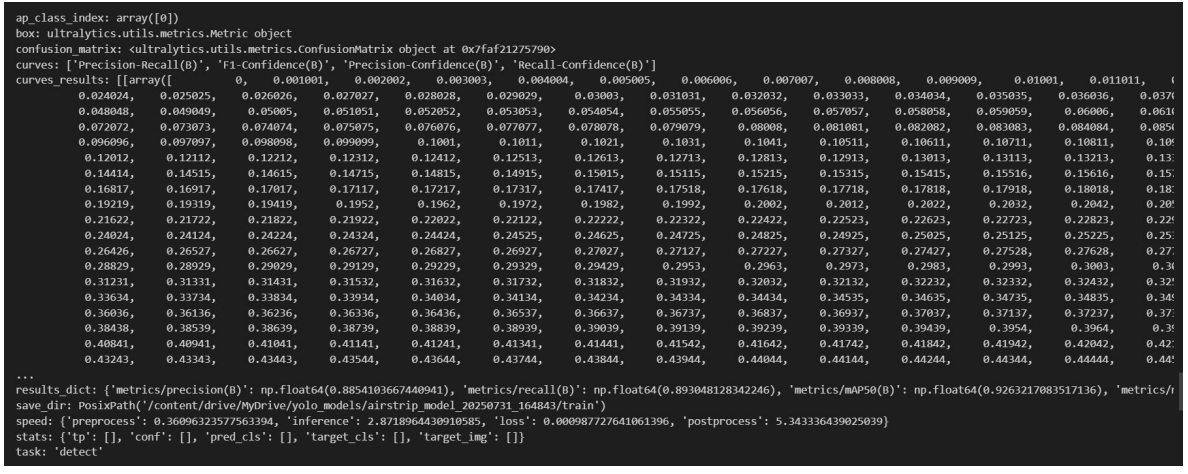




How It Works

2. Model Training – *Yolo2.ipynb*

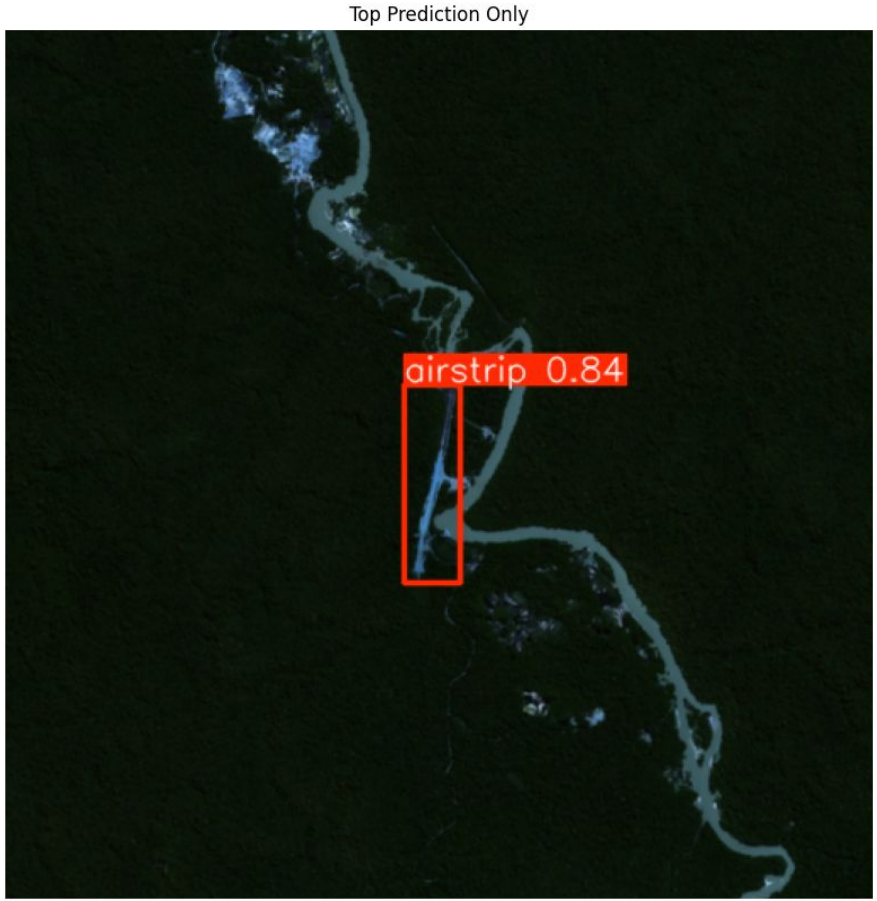
- Splits dataset into 80% train / 20% validation automatically.
- Generates dataset.yaml dynamically.
- Trains YOLOv8 for 100 epochs on 708 training images and 177 validation images.
- Saves each trained model with time stamped folder to Google Drive.
- Logs metrics (mAP, precision, recall) to metrics.txt.



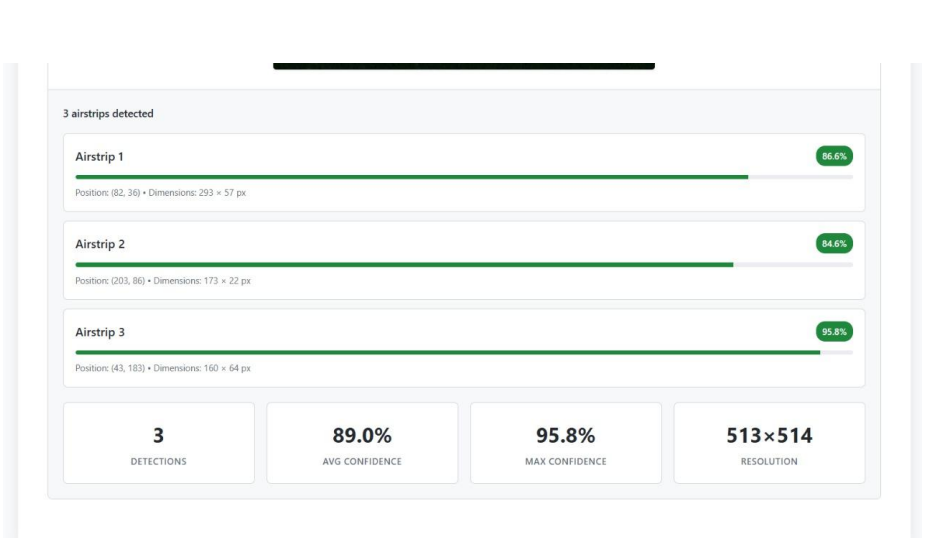
How It Works

3. Testing & Visualization

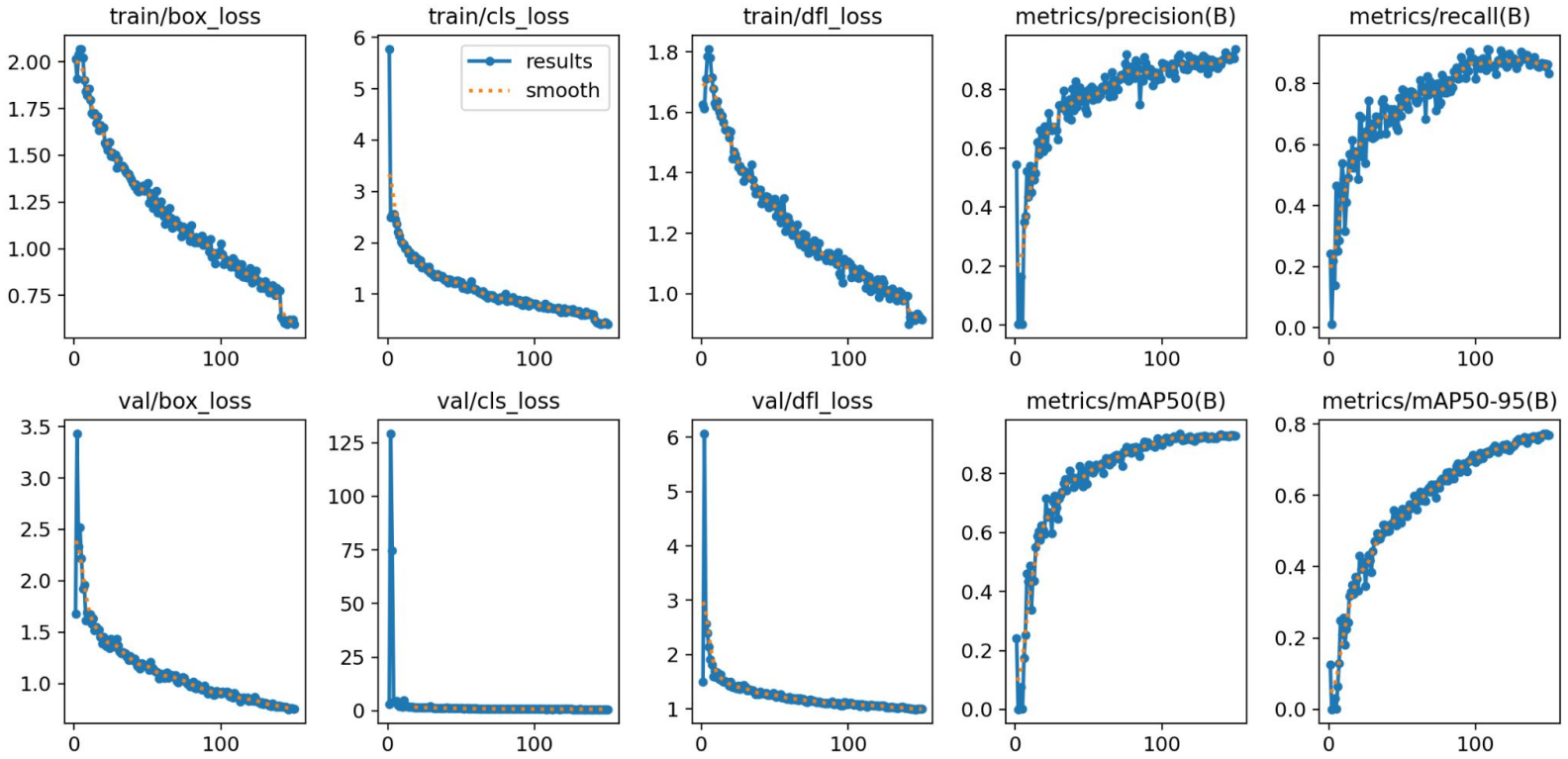
- Tests trained model on custom uploaded images.
- Displays top predictions with bounding box overlays for visual validation.



Working Demo

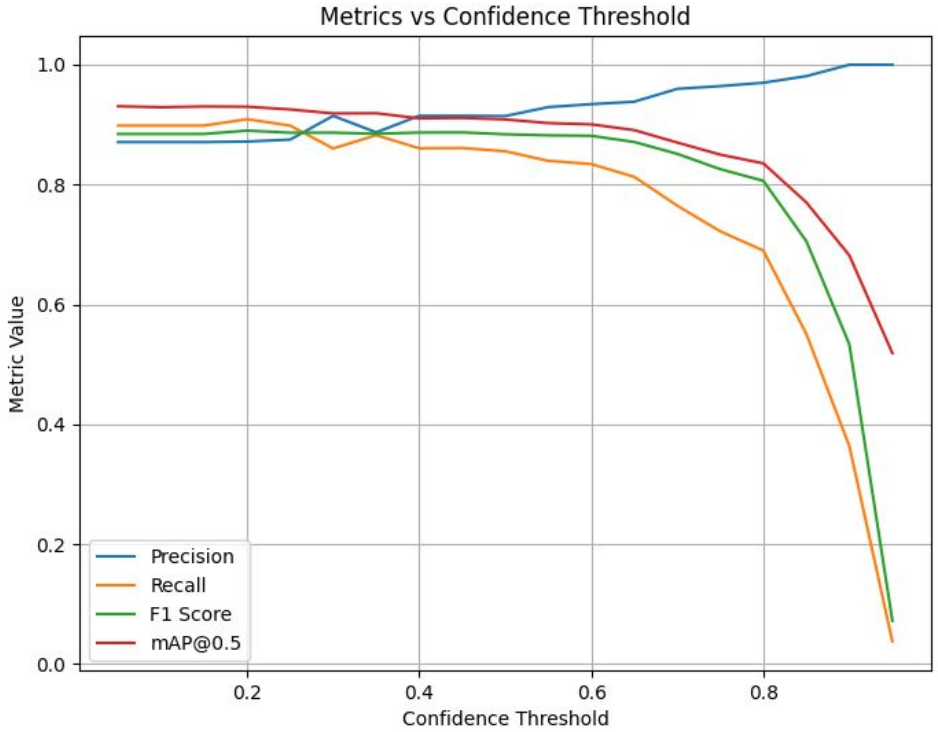


Results & Metrics



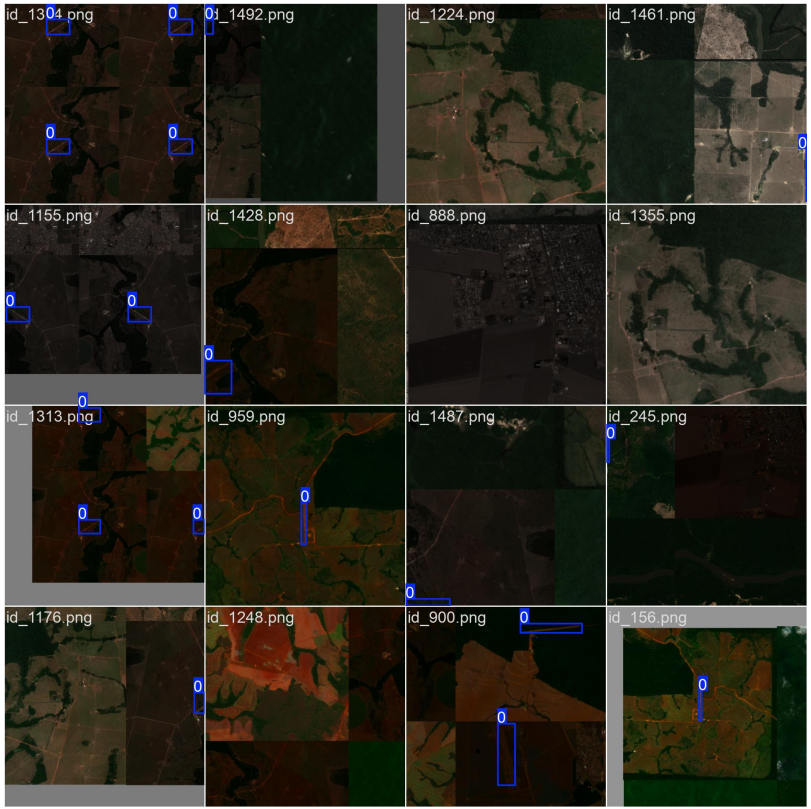
Results after Training

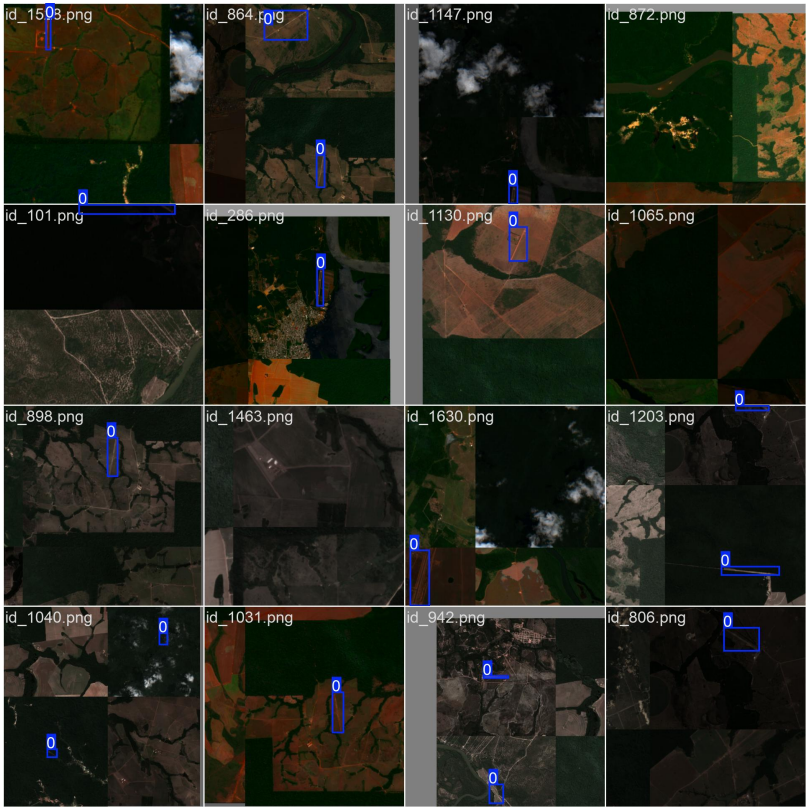
Results & Metrics



Confidence threshold data from metric maxing - final threshold used 30%

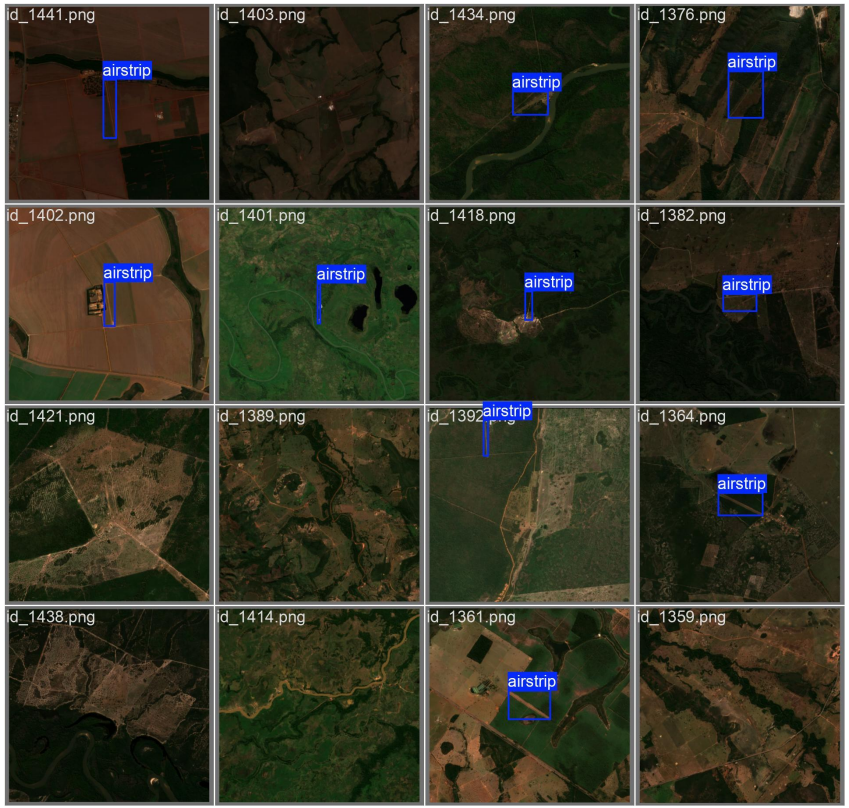
Results & Metrics



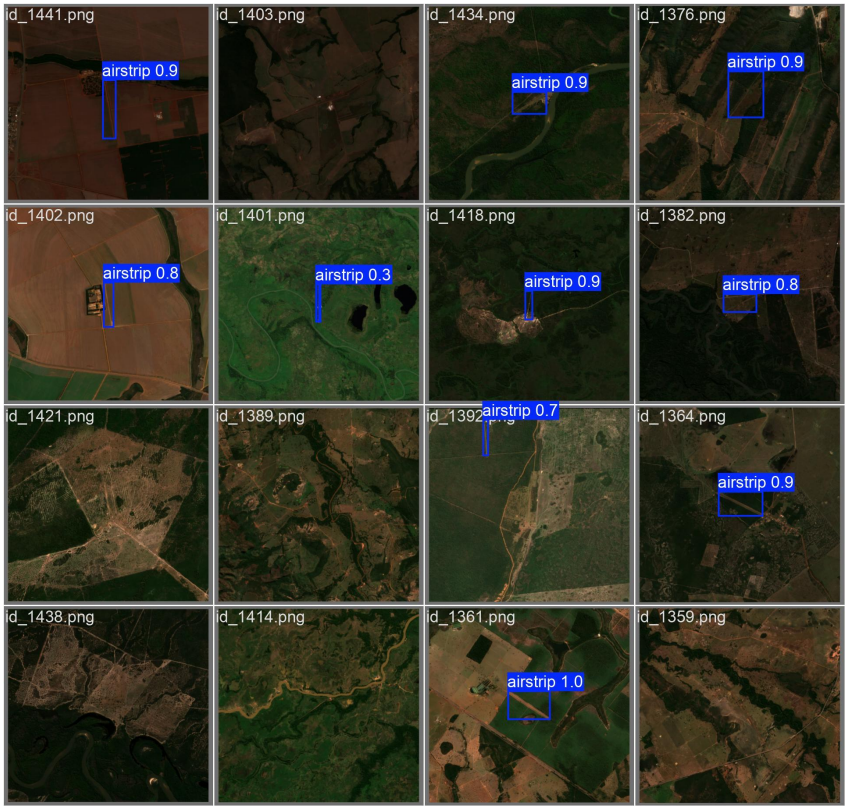


Augmented training batches

Results & Metrics



Validation batch with the bounding boxes



Predictions from yolo model

Future Scope

- Extend to other regions.
- Integrate multispectral imagery to improve detection under dense canopy.
- Build real-time monitoring dashboard using satellite feeds.
- Incorporate thermal or hyperspectral data to detect recent runway activity.
- Automated alerts to environmental authorities when new runways are detected.

Q n A

Thank You