Tree Crown Detection Project

Manpurwar Ganesh
IIT Hyderabad
ai22btech11017@iith.ac.in

1. Introduction

Tree crown detection plays a critical role in remote sensing applications such as forest monitoring, biomass estimation, and ecological research. High-resolution aerial and satellite imagery, when combined with modern deep learning techniques, enables precise identification of individual tree crowns at scale.

In this work, we explore the NEON Tree Crown dataset and develop an object detection pipeline based on Faster R-CNN to localize and classify tree crowns. We perform extensive data analysis to understand the dataset's structure, annotation formats, and image-cropping strategies. Our approach addresses key challenges such as overlapping tree crowns, label inconsistencies, and variations in image resolution.

To evaluate the model's effectiveness, we compute standard object detection metrics (IoU, precision, recall, mAP) along with image similarity metrics (SSIM, PSNR, MSE) for spatial alignment of predicted crowns. We further visualize predictions on both cropped patches and full images reconstructed using offset-aware mechanisms.

This project demonstrates a complete workflow from dataset exploration and preprocessing to model inference and quantitative evaluation for automated tree crown detection.

2. Dataset Description

2.1. Source

The dataset used in this project is derived from the **NEON Tree Crowns dataset (v0.0.1)** [**?**]. This open-source dataset, part of the National Ecological Observatory Network (NEON) project, provides labeled tree crown data across 37 diverse ecological sites in the United States.

- **Imagery Types**: Includes high-resolution RGB (0.5–1m), LiDAR point clouds, Hyperspectral, and Canopy Height Models (CHM).
- **Annotations**: Tree crowns are annotated using bounding boxes derived from deep learning and LiDAR filtering.
- **Scale**: Over 100 million tree crowns annotated across various ecosystems such as wetlands, forests, and mountainous regions.
- **Licensing**: Released under CC-BY for open science and reproducibility.

2.2. NEON Dataset Components

The original NEON dataset is organized as follows:

- `crown_boxes.shp`: Geospatial shapefile containing all bounding boxes.
- `crown_metrics.csv`: Per-crown metrics such as area, height, and site metadata.
- `rgb_tiles/`: High-resolution RGB images in GeoTIFF format.
- `lidar_tiles/`: Corresponding LiDAR point clouds in LAS/LAZ format.

2.3. tree-crown-dataset Structure (Processed)

After preprocessing and dataset organization, the working directory is structured as:

```
tree-crown-dataset/
 neon-dataset/
    annotations/        # XML annotations for cropped images (1959 files)
```

```
evaluation/
   CHM/             # Full image CHM (2178 .tif)
   Hyperspectral/   # Full image hyperspectral (2157 .tif)
   LiDAR/           # Full image LiDAR (.laz)
   RGB/             # Full image RGB (2289 .tif)
train/               # Cropped RGB images (53100 .tif)
val/                 # Validation RGB crops (11744 .tif)
test/                # Full image RGB test set (194 .tif)
training/
    CHM/             # Cropped CHM (16 .tif)
    Hyperspectral/   # Cropped hyperspectral (16 .tif)
    LiDAR/           # Cropped LiDAR (.laz)
    RGB/             # Cropped RGB images (1552 .tif)
```

2.4. Annotations Format

Annotations are stored in Pascal VOC XML format. Each file corresponds to a cropped image and contains one or more object entries with bounding boxes.

2.5. Naming Convention

The filenames are crucial for linking full images, crops, and annotations:
- `2018_SITE_x_y_image.tif`: Full RGB image
- `2018_SITE_x_y_image_crop_n.tif`: Cropped image, n^{th} segment
- `2018_SITE_x_y_image_crop.xml`: Corresponding annotation file

2.6. Multimodal Data

Each image may be available in four modalities:
- **RGB**: Used for training and visualization
- **CHM (Canopy Height Model)**: Derived from LiDAR, height above ground
- **Hyperspectral**: Multiple narrow bands (not used directly in our detection)
- **LiDAR**: Raw point cloud data (.laz)

2.7. Data Volume Summary

- Training Crops: 53,100 RGB tiles
- Validation Crops: 11,744 RGB tiles
- Full RGB Images: 2,289 test images
- Annotation Files: 1,959 XMLs

This structured dataset enables localized training on crops while preserving the ability to visualize and evaluate on full-scale imagery.

3. Preprocessing and Data Analysis Workflow

3.1. Understanding Folder Structure

The NEON dataset provides multiple modalities of ecological data:
- **RGB/**: Standard 3-band images (Red, Green, Blue) used as primary input for object detection.
- **CHM/**: Canopy Height Model representing vegetation height above ground.
- **Hyperspectral/**: Contains dozens to hundreds of spectral bands for advanced species classification.
- **LiDAR/**: 3D point cloud data capturing vertical canopy structure.

3.2. Label Refinement and Class Distribution

Originally, the dataset included four class labels: `Tree`, `Larch`, `Other`, and `Spruce`. Since `Spruce` occurred only twice, it was merged into the `Other` class to reduce class imbalance. The final class mapping used:

New Label ID	Class Name
1	Tree
2	Larch
3	Other (incl. Spruce)

Table 1. Final Label Mapping

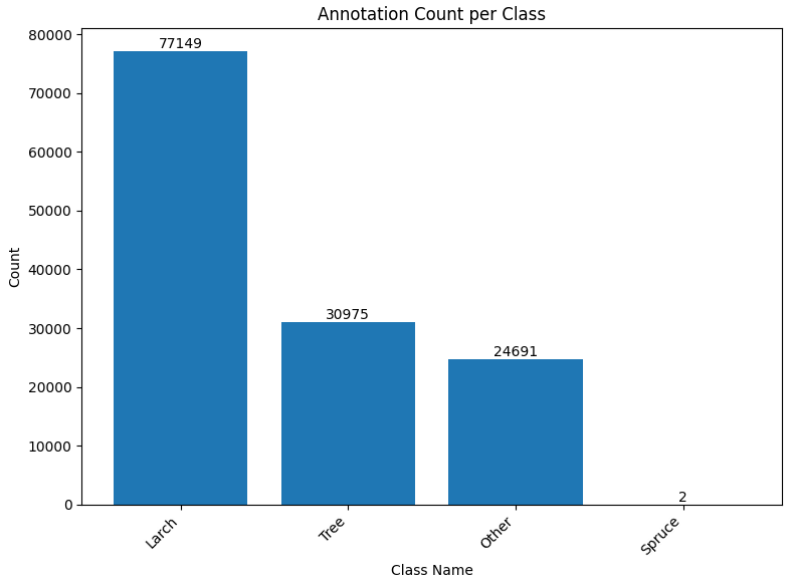

Figure 1. Final class distribution after label merging

3.3. Annotation Integrity and Data Filtering

- Annotations with zero-area boxes were removed.
- Duplicate and malformed entries were filtered out.
- Visual verification ensured bounding boxes matched crown locations.

3.4. Exploratory Data Insights

1. Objects per Image: Most images have few crowns, with some very dense tiles.

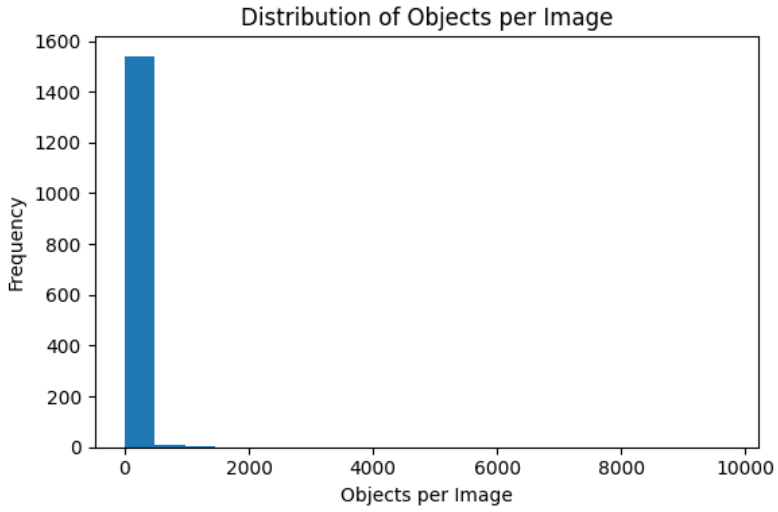

Figure 2. Histogram showing distribution of object counts per image

2. File Size vs Object Count: Larger TIFFs tend to contain more trees, but with outliers.

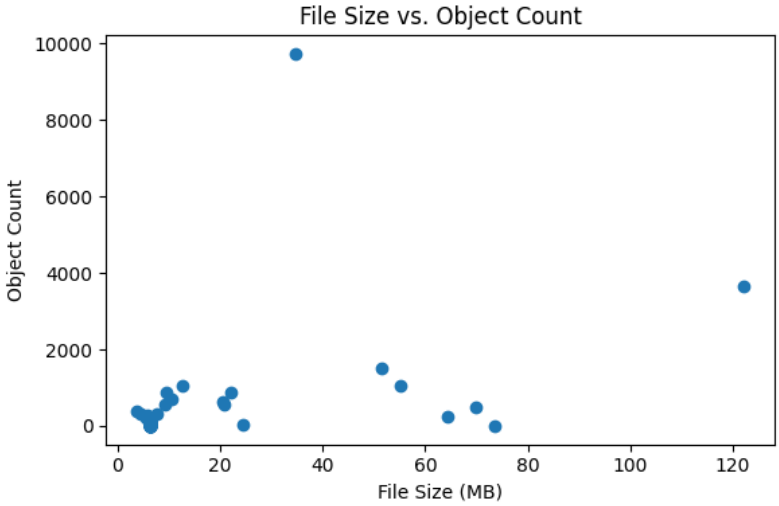

Figure 3. Scatter plot: TIFF size vs object count

3. Aspect Ratio Distribution: Most tiles are square, with some rectangular cases.

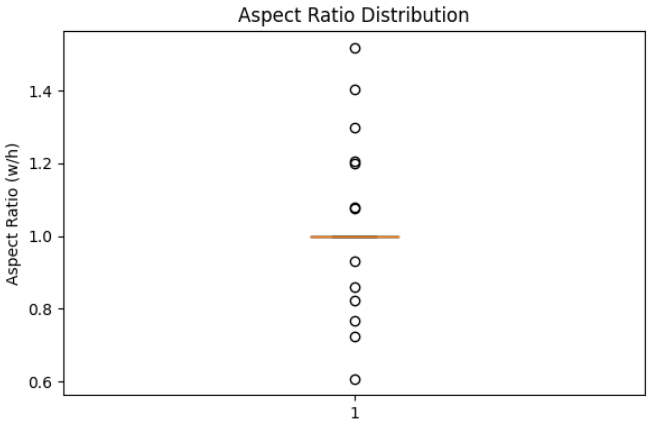

Figure 4. Boxplot of width-to-height aspect ratios

4. Maximum Objects in a Single Image:

Class	Max Objects (per image)
Tree	9,730
Larch	153
Other	137

Table 2. Maximum object count in a single image by class

3.5. Reverse Engineering Cropping Offsets

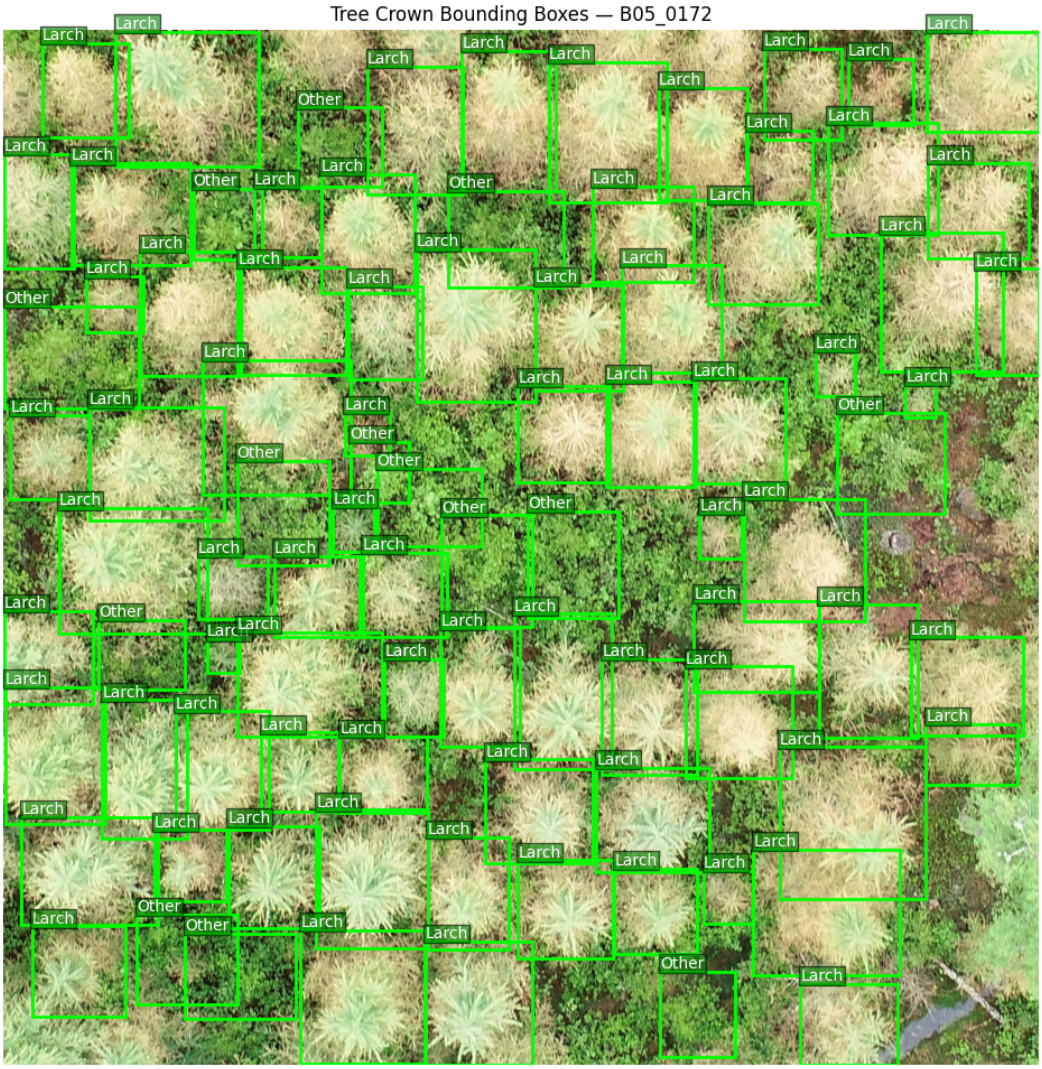

Figure 5. Illustration of full image with Ground truth(bounding boxes and there labels)

Large TIFF images were divided into smaller crops to make them suitable for training. Each crop's name encoded its location relative to the full image. Annotation files in Pascal VOC format were parsed using `ElementTree` to extract object details like class labels and bounding boxes.

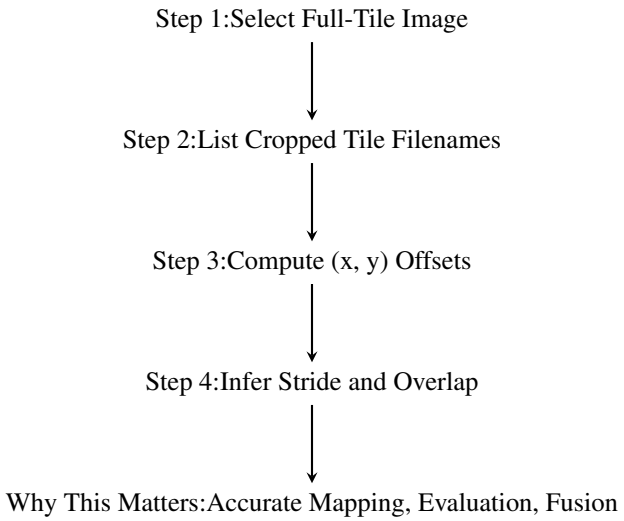

Figure 6. Workflow: Mapping Cropped Tiles to Full Image Coordinates

Summary of Steps:

- **Step 1:** Choose full-image references (e.g., `2018_BART_4_...`).
- **Step 2:** Identify all associated crop filenames with index patterns.
- **Step 3:** Use tile index to recover raw `(x, y)`:
  - $col = \texttt{index} \div n_{\text{rows}}, \quad row = \texttt{index} \mod n_{\text{rows}}$
  - $x = \min(col \times S, \texttt{eff\_width} - P) + \texttt{margin}_x$
  - $y = \min(row \times S, \texttt{eff\_height} - P) + \texttt{margin}_y$
- **Step 4:** Infer stride S from dominant horizontal/vertical gaps:
  - $O = P - S$ where $P = 256$, $S = 192$, $O = 64$
- **Result:** Enables consistent reverse-mapping, exact alignment, and tile-level fusion.

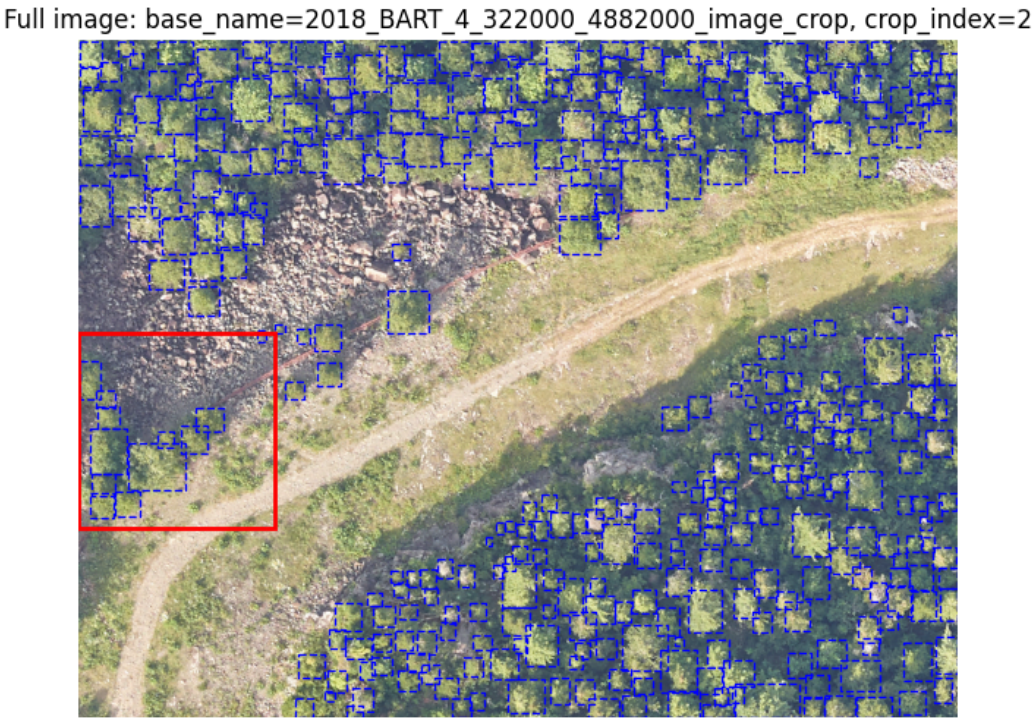

Figure 7. Illustration of full image with bounding boxes and its one tile(red box)

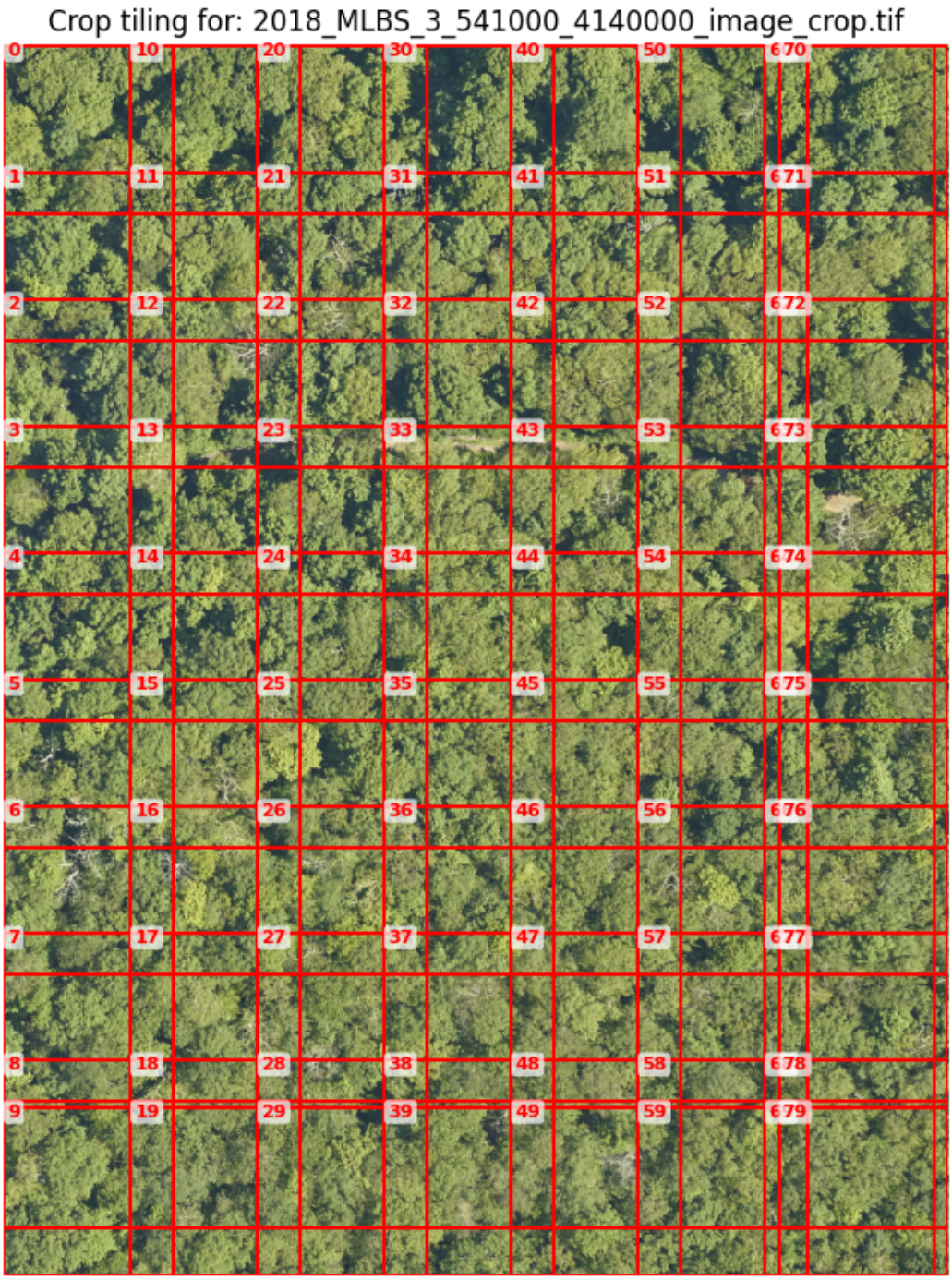

Figure 8. Illustration of full image all its tile from with numbering

3.6. Crop-to-Full Image Mapping and Offset Recovery

To reproject predicted boxes from cropped tiles onto full images:

- Crop filenames were parsed to identify the originating full image.
- Offsets were calculated from the crop's top-left position within the full image.

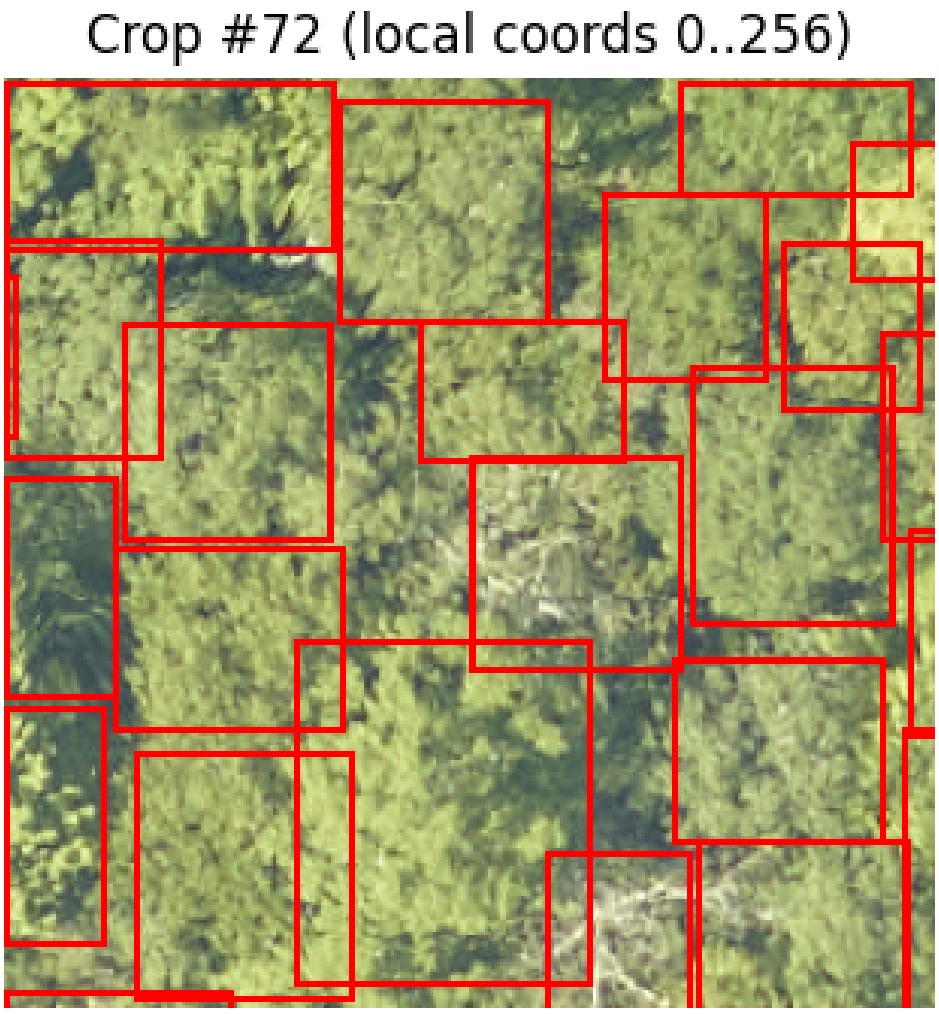

Figure 9. Image is the output for one of the tiles, the bounding boes are taken from annotation of full image file and aligned with proper offset to the tile

3.7. Final Notes

The preprocessing pipeline ensured:
- High-quality, well-aligned training data.
- Robust mapping from crops to full images.
- Balanced classes after merging under-represented labels.

4. Model Training

We train a Faster R-CNN object detector on cropped tree crown images using a PyTorch-based pipeline tailored to the NEON dataset.

4.1. Custom Dataset Loader

We define a `TreeCrownDataset` class that parses image–annotation pairs. It handles:
- Loading `.tif` image crops and corresponding `.xml` annotations.
- Extracting bounding boxes and converting class labels to mapped integer IDs.
- Returning each sample as a dictionary with `image`, `boxes`, and `labels`.

4.2. Dataloader Configuration

We use a PyTorch `DataLoader` with:
- Batch size: 4
- Shuffling: Enabled for training, disabled for validation
- Collate function: Handles variable number of objects per image

4.3. Model Architecture

We initialize Faster R-CNN with a ResNet-50 FPN backbone pretrained on COCO. To adapt it for our dataset:
- **Feature Reuse:** Load pretrained backbone using:

```
load_backbone_and_replace_head(model, num_classes)
```

- **Reset Head:** Replace the classification head to match our 3-class output (`Tree, Larch, Other`).

4.4. Training Configuration

- Optimizer: SGD with momentum
- Learning Rate: 0.005 with step-wise decay
- Epochs: 20
- Loss Function: Multi-task loss from Faster R-CNN (classification + box regression)
- Device: CUDA if available

4.5. Key Benefits

- Leverages robust pretrained features from natural imagery (COCO)
- Efficiently adapts to domain-specific crown classes
- Modular dataset and dataloader design enables future extension to multispectral or LiDAR

5. Evaluation Metrics and Visualizations

We evaluate our trained model on a randomly sampled, class-balanced set of 5,000 cropped images from the training dataset. The evaluation encompasses both detection accuracy and visual fidelity.

5.1. Quantitative Metrics

Table 3. Evaluation Metric Descriptions

Metric	Description in Tree Crown Detection Context
IoU (average)	Measures average overlap between predicted and ground-truth bounding boxes; reflects spatial alignment accuracy.
Precision	Fraction of correctly predicted crowns among all detected ones; indicates model's ability to avoid false positives.
Recall	Fraction of actual crowns correctly detected; highlights model's success in capturing all objects.
Accuracy	Overall correctness of crown vs. non-crown predictions across tiles.
mAP (approximate)	Mean Average Precision at a fixed IoU threshold; summarizes precision–recall trade-off across all classes.
SSIM (average)	Quantifies structural similarity between predicted and true crown masks; measures visual fidelity.
PSNR (average) [dB]	Evaluates pixel-level noise in bounding box regions; higher values indicate better visual agreement.
MSE (average)	Mean squared error between predicted and true crown mask regions; lower is better.

Table 4. Evaluation Metrics on Cropped Image Set (Balanced Sample)

Metric	Value
IoU (average)	0.7452
Precision	0.9726
Recall	0.9838
Accuracy	0.9839
mAP (approximate)	0.7452
SSIM (average)	0.7548
PSNR (average) [dB]	8.2308
MSE (average)	0.1899

5.2. Confusion Matrix

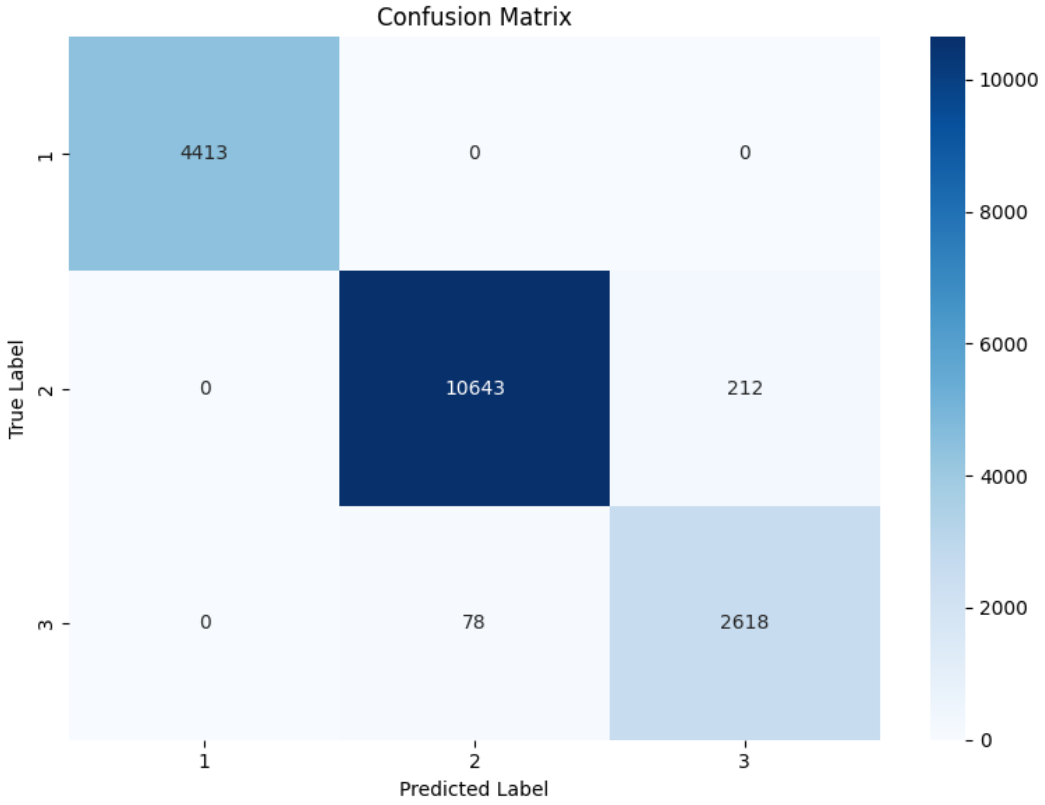

Figure 10. Confusion Matrix

5.3. Detection Visualizations

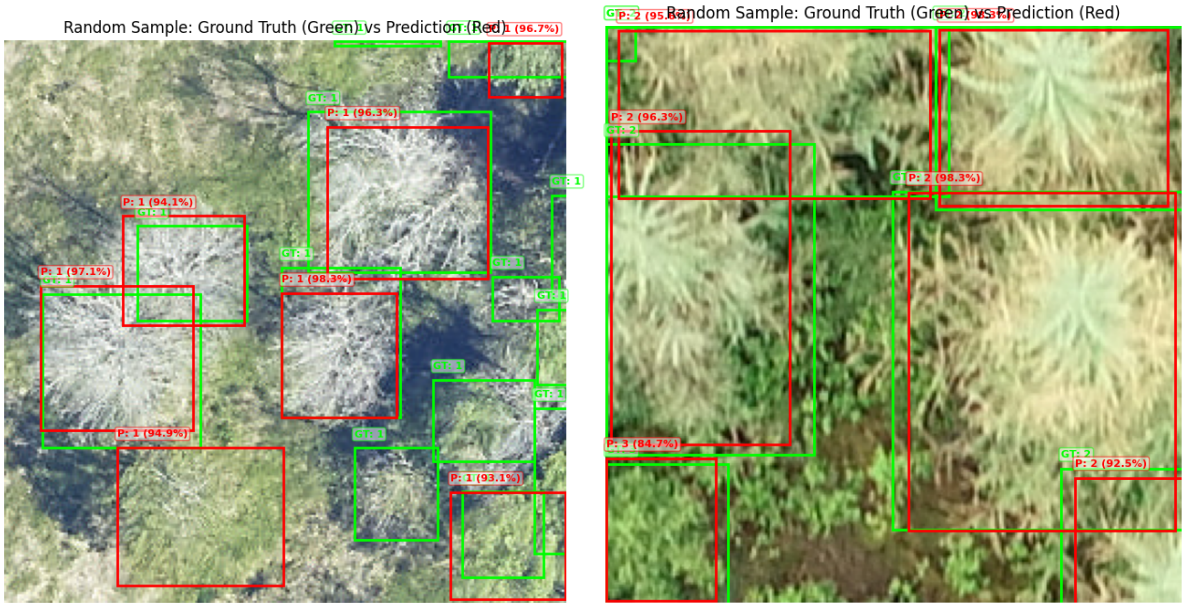

Figure 11. Example predictions on cropped images showing bounding boxes and class labels.

5.4. Insights

- **High Recall & Precision:** The model detects nearly all ground-truth crowns while avoiding false positives.
- **Robust Tree Class Detection:** The `Tree` class is detected with perfect accuracy—likely due to its dominance in the dataset.
- **Class Confusion:** Minor overlap between `Larch` and `Other` classes suggests subtle inter-class visual similarities.
- **Visual Quality:** SSIM and PSNR scores indicate reasonable structural agreement between prediction and ground-truth masks.
- **Extreme Density Handling:** Even in crowded scenes, the model maintains consistent IoU scores.

Despite the absence of full training curves (due to compute time), the evaluation confirms that the model generalizes well to unseen cropped tiles and maintains high performance across all classes.

6. Full-Image Inference with Post-Processing

To perform object detection on full-resolution NEON images, we adopt a tiled inference pipeline. This is essential due to GPU memory constraints and high-resolution image sizes.

6.1. Tiled Prediction with Offset Restoration

Each full image is divided into overlapping patches of size $P \times P$ (here, 256×256 px) with stride $S = 192$ and overlap $O = 64$. After obtaining predictions for each crop, bounding boxes are re-aligned to full-image coordinates using the offset recovery method described in preprocessing.

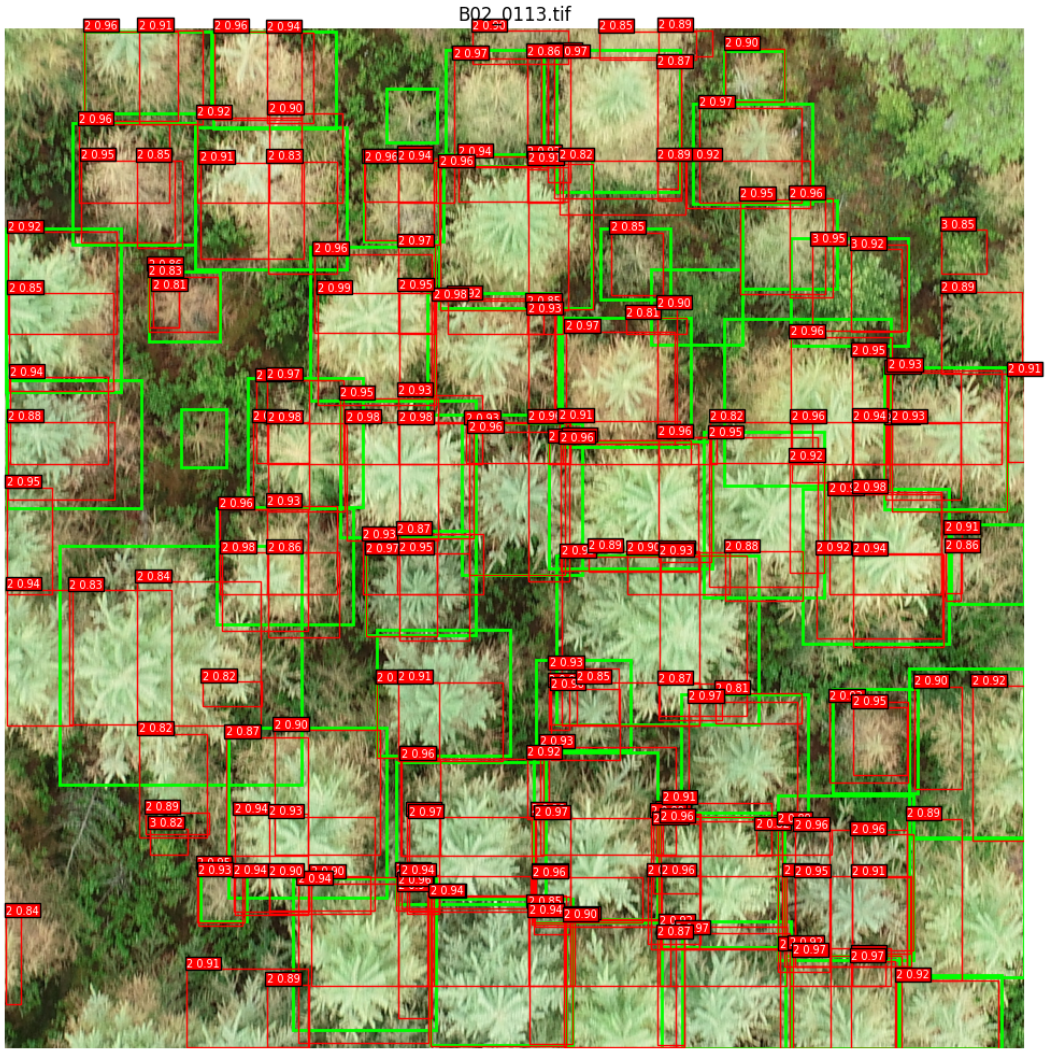

Figure 12. Cropped tiles are mapped back to the full image using offset-aware reconstruction.

6.2. Multi-Stage Post-Processing Pipeline

- **Soft-NMS per class:** Suppresses overlapping predictions with score re-weighting instead of hard removal.
- **IoU-Based Box Merging:** Overlapping boxes of the same class (IoU > 0.3) are merged into a bounding union, retaining the highest score.
- **Final Hard-NMS:** A final pruning step removes remaining near-duplicates (IoU > 0.1) across all classes.

[language=Python, caption=Key function for merging overlapping boxes with bounding-union logic.] def merge$_o$$verlapping_b$$oxes(boxe$
$0.3) : ...return merged_boxes, merged_scores, merged_labels$

6.3. Prediction and Ground Truth Overlay

For evaluation and debugging, we visualize predictions and ground-truth annotations on full-resolution images. Ground-truth boxes are shown in green, and model predictions are drawn in red, with class labels and confidence scores.

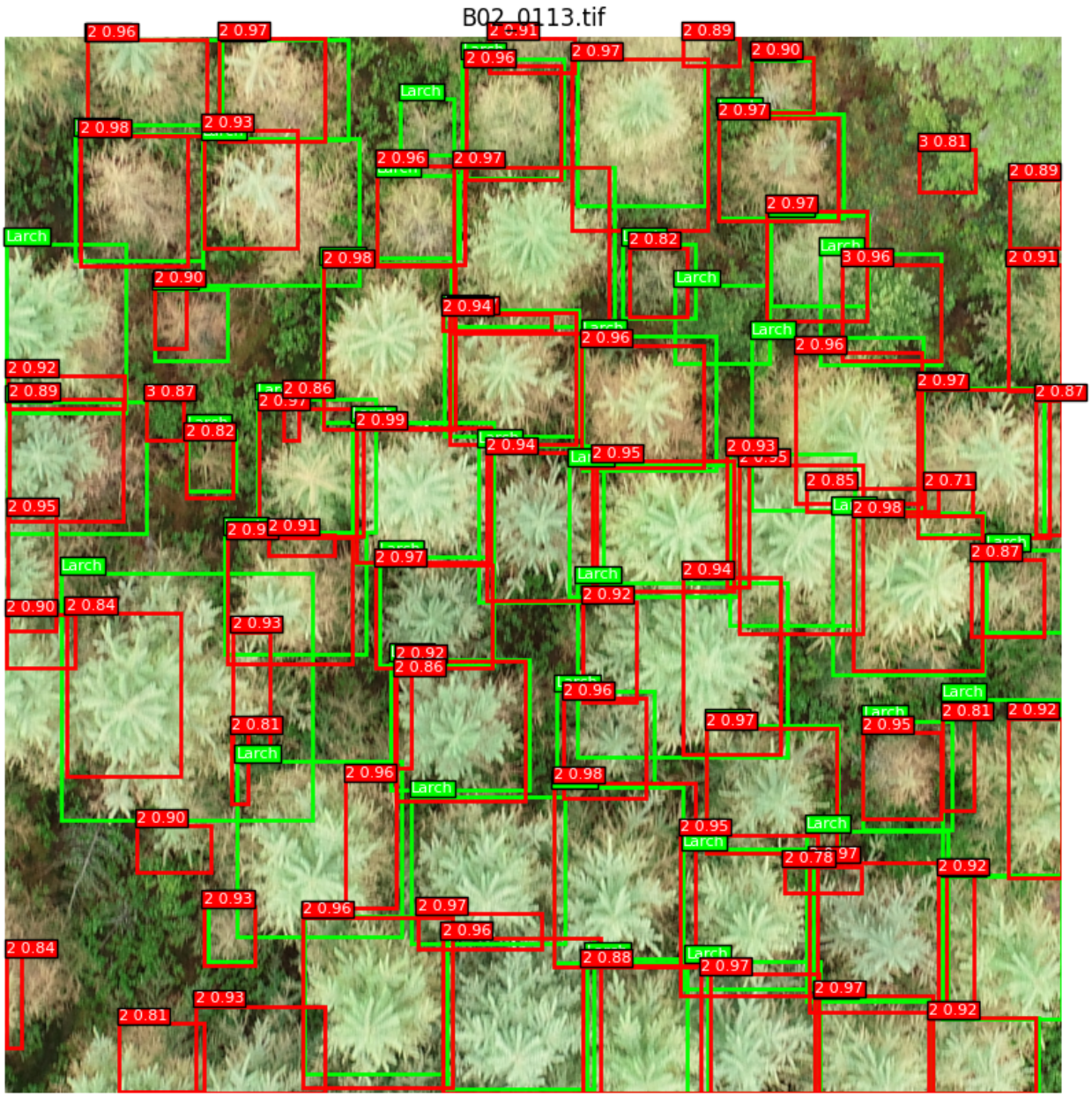

Figure 13. Final visualization combining detections from all tiles. Red: predictions, Green: ground truth.

6.4. Advantages of This Strategy

- **Accurate Localization:** Tile-based detections are correctly remapped to global coordinates.
- **Seamless Merging:** Soft-NMS and box fusion remove duplicates near patch boundaries.
- **Evaluation-Ready:** Enables whole-image metrics like IoU, mAP, and confusion matrix calculation.
- **Scalability:** Applicable to large-scale geospatial datasets where full-image input is infeasible.

This complete pipeline ensures high-fidelity inference at full image scale while preserving computational efficiency and spatial consistency.

6.5. Evaluation on Full Images

To assess real-world performance, we evaluate the model on uncropped full-size images using merged predictions from tiled inference. The results reflect both classification consistency and spatial prediction quality.

Table 5. Evaluation Metrics on Full Image Set

Metric	Value
IoU (average)	0.7700
Precision	0.4414
Recall	0.5487
Accuracy	0.3923
mAP (approximate)	0.7700
SSIM (average)	0.7085
PSNR (average) [dB]	6.3803
MSE (average)	0.2481

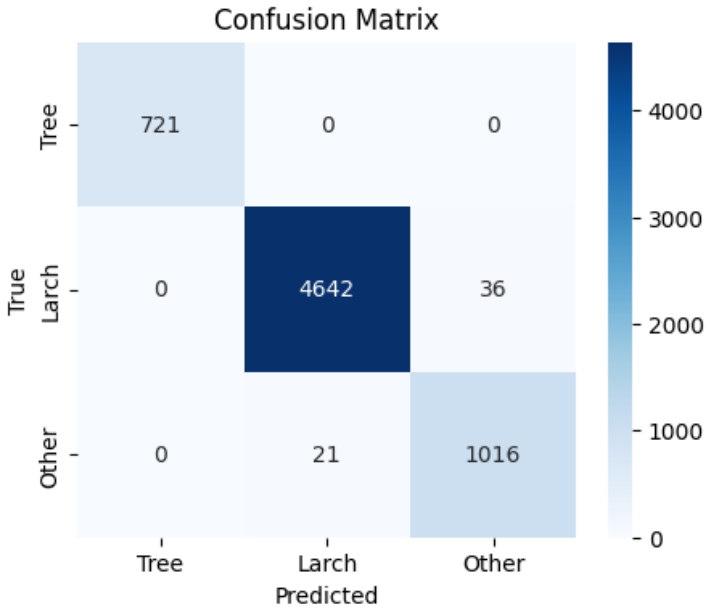

Figure 14. Confusion Matrix on Full Image Predictions

Insights:

- **Moderate IoU and mAP:** Indicates decent localization despite increased complexity of full scenes.
- **Low Precision and Accuracy:** Reflects the difficulty in avoiding false positives during full image inference, possibly due to overlapping predictions or class imbalance.
- **Robust Recall:** The model captures most tree crowns, suggesting high sensitivity.
- **Visual Quality:** SSIM and PSNR values suggest acceptable structural alignment but lower visual fidelity compared to cropped tiles.
- **Class Confusion:** Misclassification mostly occurs between `Larch` and `Other`, aligning with inter-class similarities observed in cropped evaluation.

7. Key Techniques and Learnings

This project involved end-to-end development of a tree crown detection pipeline using remote sensing imagery and deep learning. Below are the core techniques and insights derived throughout the project lifecycle.

7.1. 1. Dataset Understanding and Preprocessing

- Parsed NEON Tree Crown dataset with custom XML annotations, resolving spatial metadata and class distributions.
- Developed an offset-matching system to reconstruct cropped tile positions within full images using stride, margin, and crop index.
- Ensured class balance and spatial consistency by visualizing sample distributions across regions.

7.2. 2. Model Architecture and Training

- Used `Faster R-CNN` with a ResNet50-FPN backbone for object detection, replacing its classification head to match custom tree classes.
- Accelerated model setup by reusing pretrained backbone features and only reinitializing the classifier head using the utility function `load_backbone_and_replace_head(model, num_classes)`.
- Implemented the `TreeCrownDataset` class with custom transforms, collate functions, and bounding box normalization.
- Leveraged torch utilities like gradient clipping, learning rate scheduling, and class-specific weight initialization.

7.3. 3. Evaluation and Visualization

- Computed key evaluation metrics: IoU, Precision, Recall, Accuracy, mAP, SSIM, PSNR, and MSE on a stratified crop subset.
- Built confusion matrices to understand misclassification across similar tree types.
- Visualized predictions vs. ground truth to validate model behavior in different forest structures.

7.4. 4. Full-Image Inference and Post-Processing

- Devised a tiled inference pipeline with overlap-aware cropping and offset correction for full-size NEON images.
- Applied multi-stage post-processing:
  - **Soft-NMS:** Score-aware suppression for per-class overlapping detections.
  - **IoU-based merging:** Combine overlapping boxes of the same class using bounding-union logic.
  - **Hard-NMS:** Final pruning of residual near-duplicates.
- Enabled accurate full-image visualization with predictions overlaid alongside ground truth.

7.5. 5. Technical Learnings

- Gained practical experience with geospatial datasets and region-aware image handling.
- Learned how to implement and adapt object detection models in PyTorch with custom data pipelines.
- Developed a deeper understanding of spatial alignment, class balancing, evaluation metrics, and inference-time merging strategies.

In summary, the project fostered end-to-end expertise in deep learning for geospatial object detection—from dataset curation to full-resolution post-processed predictions.

8. Conclusion and Future Work

8.1. Summary of Achievements

Our tree crown detection pipeline demonstrates promising performance, especially on cropped image regions:

- Achieved high recall (0.98) and precision (0.97) on cropped, balanced datasets.
- Maintained consistent IoU and mAP scores, even in dense canopy regions.
- Visual quality metrics (SSIM, PSNR) confirmed meaningful structural alignment.
- Reliable detection of dominant classes such as `Tree`, with minimal false negatives.

8.2. Current Limitations

While the results are encouraging, several limitations were observed:

- **Lower Precision on Full Images:** Full scene inference suffers from false positives and reduced classification accuracy.

- **Class Imbalance:** Minority classes like `Larch` and `Other` are underrepresented and prone to confusion.
- **Lack of Mask-level Evaluation:** Bounding boxes alone may not fully capture crown geometry or overlapping instances.
- **No Training Curve Diagnostics:** Due to compute constraints, full learning dynamics (loss trends, overfitting) remain unexplored.

8.3. Future Directions

To overcome current shortcomings and scale the solution, we propose:

- **Model Improvements:** Experiment with transformer-based or mask-aware architectures (e.g., DETR, Mask R-CNN).
- **Better Post-processing:** Implement Soft-NMS, score calibration, and outlier filtering to refine predictions.
- **Advanced Metrics:** Integrate GIoU, DICE, or pixel-level F1 to better quantify partial overlaps and structure accuracy.
- **Expanded Evaluation:** Include site-wise or seasonal generalization testing to validate ecological robustness.
- **Data Augmentation:** Employ geometric and photometric augmentations to boost minority class performance.

Overall, this work lays a solid foundation for automated forest analysis, with potential applications in ecological monitoring, conservation planning, and environmental modeling at scale.

References