Automated Tree Detection in High-Resolution GeoTIFF Imagery Using Deep Learning and Geospatial Processing

# Abstract

This paper presents an end-to-end pipeline for automated tree detection in high-resolution GeoTIFF imagery. The input GeoTIFF files are expected to originate from pre-processed images generated by photogrammetry software, ensuring that the high resolution and essential metadata, including georeference details, are preserved. The system leverages deep learning (YOLO) for object detection, geospatial libraries for image tiling and reconstruction, and web technologies for user interaction. The workflow enables users to upload large geospatial images, which are then processed to produce both annotated imagery and vector shapefiles of detected trees.

# 1. Introduction

Remote sensing and geospatial analysis are critical for environmental monitoring, forestry, and urban planning. Detecting individual trees in large aerial or satellite images is a challenging task due to image size and variability. This work describes a modular, reproducible pipeline that combines image tiling, deep learning-based detection, and geospatial data export, accessible via a web interface.

# 2. System Overview

The system consists of four main components:

* - A Flask-based web application for user interaction.
* - An end-to-end processing module for slicing, detection, and output generation.
* - Image stitching utilities.
* - Auxiliary scripts for tile processing.

# 3. Methods

# 3.1 Web Application

The web interface (webapp.py) allows users to upload GeoTIFF files. Upon upload, the file is saved, and the end-to-end processing pipeline is triggered. After processing, users can download the stitched, annotated GeoTIFF and a Shapefile containing the detected tree locations.

# 3.2 Image Slicing

Large GeoTIFFs are divided into smaller tiles using rasterio to facilitate efficient processing and detection. The function slice\_geotiff\_to\_tiffs reads the input image and writes out tiles of a specified size, preserving georeferencing information.

# 3.3 Object Detection

Each tile is processed by a YOLO model (via the ultralytics package). Tiles are converted to RGB images and saved temporarily as JPEGs. The model predicts bounding boxes for trees, which are then drawn on the tile and saved as new GeoTIFFs. The bounding box coordinates are also transformed into geospatial polygons for later export.

# 3.4 Image Stitching

After detection, the annotated tiles are stitched back together using rasterio.merge to reconstruct the full image with detection overlays. The function stitch\_tiff\_tiles ensures georeferencing is preserved.

# 3.5 Shapefile Generation

Detected bounding boxes are converted to geospatial polygons using the tile’s affine transform. All polygons are merged and saved as an ESRI Shapefile using fiona and shapely, preserving the original image’s coordinate reference system.

# 4. Implementation Details

* - The pipeline is orchestrated by the run\_end\_to\_end function in EndToEnd.py.
* - Temporary and output directories are managed automatically.
* - The system is modular, allowing for easy replacement of the detection model or adjustment of tile sizes.
* - The web interface provides user feedback and download links for results.

# 5. Results and Discussion

The pipeline enables efficient processing of large geospatial images, producing both raster and vector outputs suitable for further GIS analysis. The use of tiling allows for scalable processing, and the modular design supports adaptation to other object detection tasks.

# 6. Conclusion

This work demonstrates a practical, automated approach for tree detection in large GeoTIFFs, integrating deep learning and geospatial data handling. The system is accessible via a web interface and produces outputs compatible with standard GIS tools.

# References

* - Ultralytics YOLO: https://github.com/ultralytics/ultralytics
* - Rasterio: https://rasterio.readthedocs.io/
* - Fiona: https://fiona.readthedocs.io/
* - Shapely: https://shapely.readthedocs.io/
* - Flask: https://flask.palletsprojects.com/