MAIS 202 - Final Results

1. Summary of the Final Model

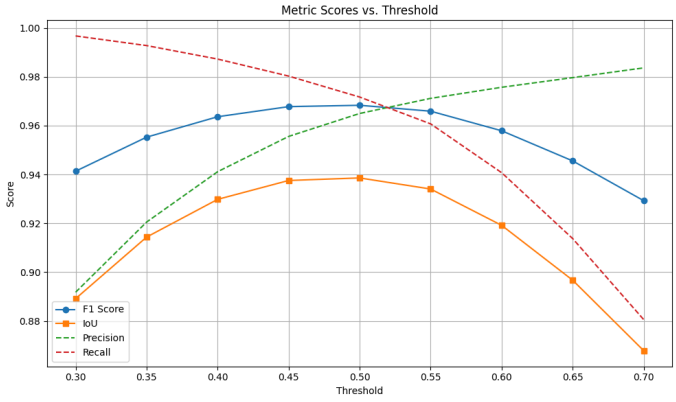
For the final iteration of my deforestation detection model, I chose to design a lightweight custom Convolutional Neural Network (CNN) architecture trained from scratch. In previous versions, I had experimented with using pre-trained base models such as ResNet. However, I found that these tended to be too powerful for the relatively small and specific dataset I was using. This often resulted in overfitting or inappropriate feature learning, where the model would memorize patterns rather than generalize from them.

The final architecture includes:

- Four convolutional layers, each followed by max pooling and ReLU activations.
- A final 1x1 convolution to reduce the feature map to a single output channel.
- A sigmoid activation to produce a soft prediction map.
- Focal Loss was used as the loss function to address class imbalance, as deforested areas are usually smaller and underrepresented.
- Inputs to the model consisted of 4-band images, combining RGB channels and the Near-Infrared (NIR) band (B8), which is known to highlight vegetation health.

By combining RGB and NIR information, the model became better at distinguishing subtle differences in forest canopy density and health, key indicators of deforestation.

To determine the best binarization threshold for the prediction map, I evaluated various threshold values on the validation set and selected the one that maximized the F1 score which unsurprisingly is 0.50.



2. Final Model Results

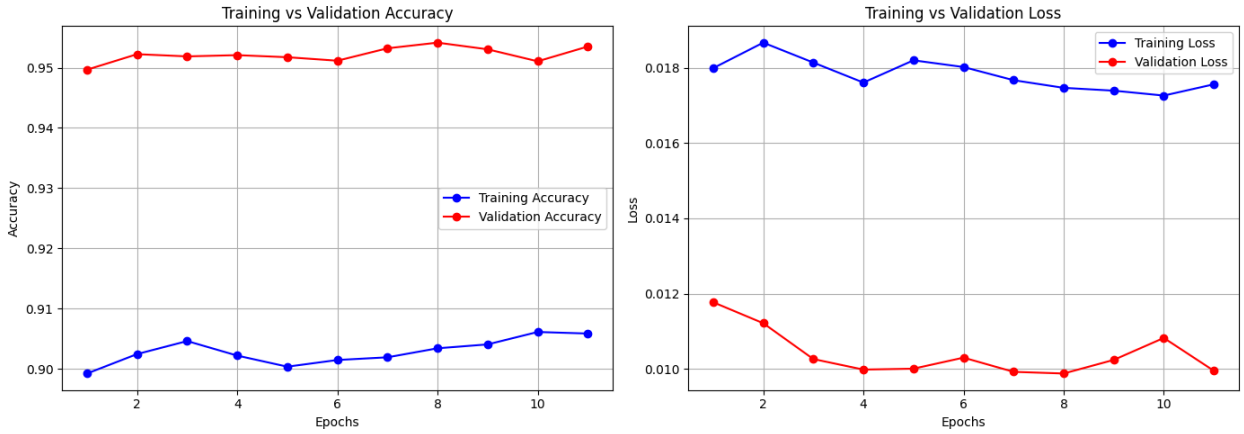
<u>Training & Validation Metrics</u>

The model was trained using an adaptive early stopping mechanism, which monitored validation loss and halted training once improvements became negligible. Although the training was set to run for up to 50 epochs, the process concluded after 31 epochs due to stabilization in the validation metrics. Over these final epochs, the model maintained high and consistent performance, with minimal variance between training and validation metrics.

By epoch 31 (the final one), the metrics were as follows:

- Training accuracy: 90.5%
- Validation accuracy: 95.4%
- Training loss: 0.0177
- Validation loss: 0.0099
- Precision: 92.3%
- Recall: 90.8%

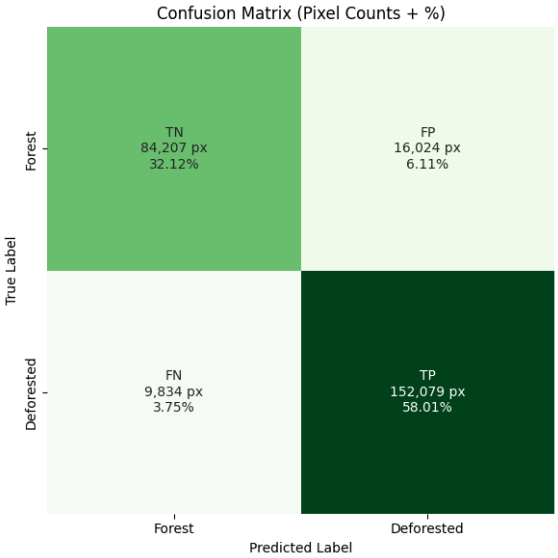
These values indicate that the model learned to generalize effectively, avoiding overfitting while achieving strong performance on unseen data.



Evaluation Metrics on the Test Set

After training, the model was evaluated on a separate test set to assess its generalization. The resulting metrics were:

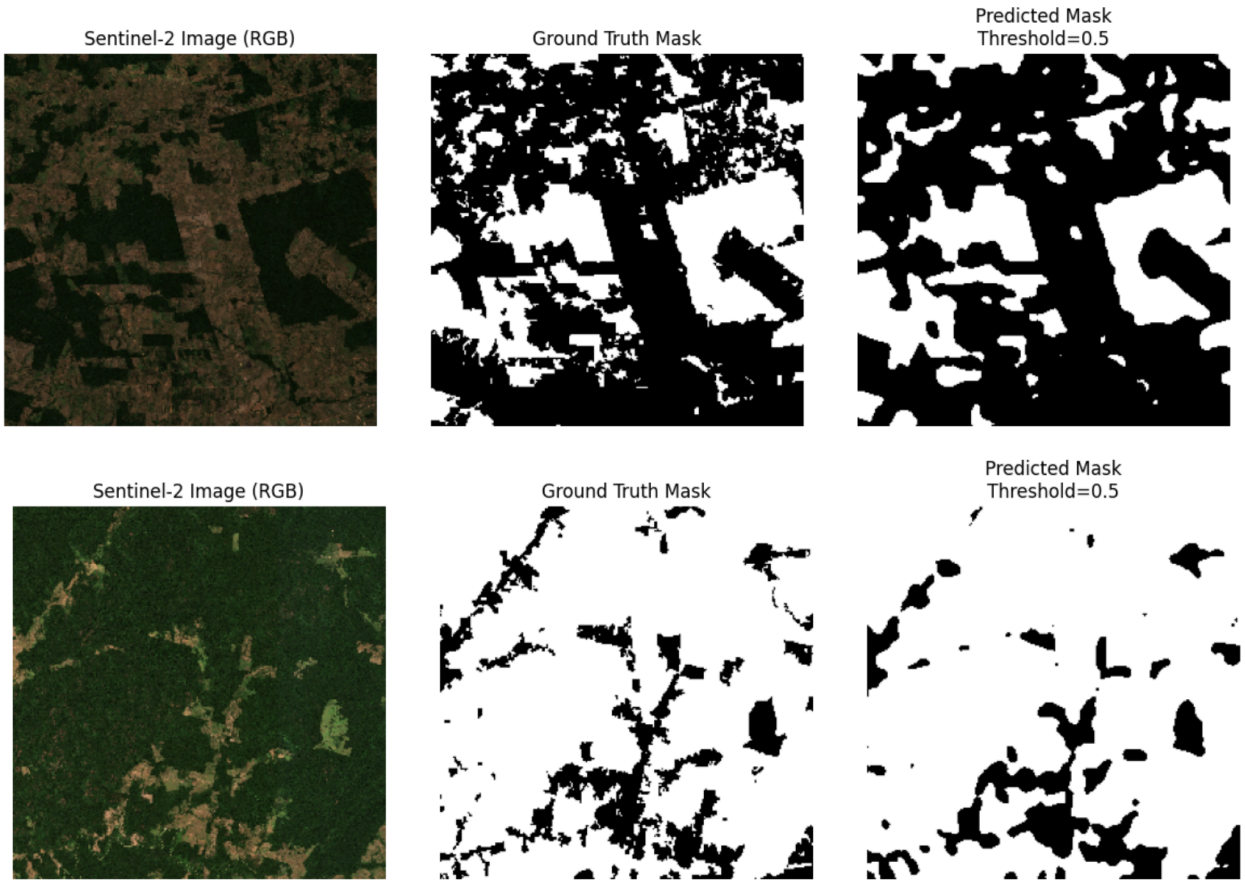
- Precision: 84.75%
- Recall: 87.99%
- F1 Score: 86.34%



Compared to the preliminary results, these metrics represent a notable improvement. In earlier versions, the validation accuracy plateaued around 61.7%, and the model struggled to learn meaningful representations. The intermediate model developed had a recall of 1.00, but indicated the whole image as deforested in all cases. These new results confirm that the model not only identifies deforestation accurately but also avoids unnecessary false positives, which is significantly better than prior versions.

Visual Analysis

The visual results further support the numerical improvements. The Sentinel-2 satellite images, when paired with ground truth masks and predicted outputs, reveal that the model is able to delineate deforested regions effectively. In complex landscapes with fragmented deforestation, the model captures edges with reasonable clarity. In more densely forested areas, it avoids excessive false detections, which had been a problem in previous attempts.



3. Final Demonstration and Integration

To showcase my model's capabilities, I built a full-stack web application using React and Flask. I chose these technologies because I've used them together in past hackathons and they allow for fast iteration and clean API integration.

Framework Overview:

- React: Frontend interface with Google Maps API integration
- Flask: Backend to manage inference and serve model predictions

- Google Earth Engine: Retrieves satellite imagery based on user-selected coordinates and time range
- TensorFlow: Runs the trained segmentation model

The final product allows users to interact with a map, select a region and time period, and receive a Sentinel-2 image alongside a deforestation prediction mask. The application handles backend preprocessing, model inference, and result visualization seamlessly.

Here's an overview of the system architecture: