MAIS 202 - DELIVERABLE 2

1. Problem Statement

This project aims to detect deforested areas using satellite imagery and semantic segmentation. The model takes Sentinel-2 satellite images as input and produces segmentation masks to highlight deforested regions. A deep learning approach is used to classify pixels as either forest or deforested land, enabling automated analysis of environmental changes.

2. Data Preprocessing

The dataset consists of Sentinel-2 satellite images and their corresponding segmentation masks, both in .tif format. It contains 16 images with 16 masks, which is fewer than initially expected. The masks, stored separately, classify regions as forest (1) or deforested (2) and were converted to a binary format (0 = deforested, 1 = forest) for training. A few adjustments were necessary due to the dataset's structure. The .tif format required specialized libraries for processing, and the images were resized to 256×256 pixels for computational efficiency. Pixel values were normalized to the [0,1] range to improve numerical stability, and data augmentation techniques such as flipping, rotation, and zooming were applied to address the limited number of samples. The processed images and masks were also converted into TensorFlow datasets to enable efficient batch processing during training of the model.

3. Machine Learning mModel

This project uses a custom-built semantic segmentation model, inspired by DeepLabV3+, to detect deforested areas in Sentinel-2 satellite images. The model processes input images and generates segmentation masks, classifying pixels as forest or deforested land.

Framework, Tools, and Model Architecture

The model was implemented using TensorFlow and Keras, with rasterio for handling .tif files. It is built using a ResNet-50 backbone for feature extraction, followed by an Atrous Spatial Pyramid Pooling (ASPP) module to capture multi-scale features. The segmentation head consists of convolutional layers, dropout (0.3), and upsampling layers, ensuring the final output is the same size as the input image.

Training, Regularization, and Optimization

The dataset was split 80% for training and 20% for validation due to the small sample size. Data augmentation (random flips, rotations, and zooming) was applied to improve generalization. Dropout (0.3) was added before the final upsampling step to reduce overfitting. The model was trained using Adam optimizer (learning rate = 0.0005), and Binary Crossentropy loss, chosen to handle pixel-wise classification.

Model Validation and Performance Analysis

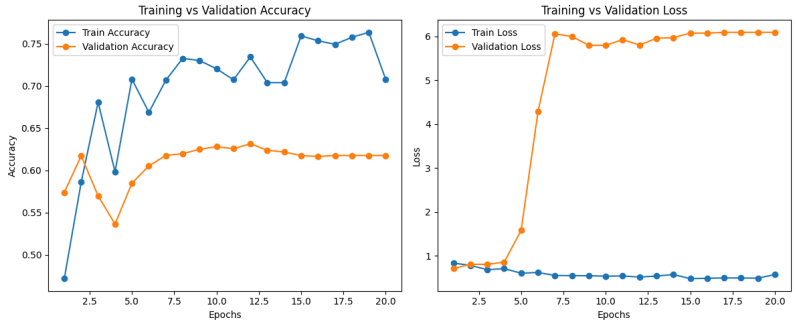
Validation was performed using accuracy and loss tracking. The model showed overfitting, with training accuracy improving while validation accuracy plateaued at ~61.7%. High validation loss (6.0+) indicated poor generalization, suggesting that the model needs improvements.

Challenges and Solutions

One challenge was understanding the dataset structure, and ensuring the model was passed the correct input. This was solved via the analysis of the files' contents before the processing. Overfitting was another issue, which was partially addressed by reducing the learning rate, applying dropout, and using data augmentation. The next steps section details solutions to the overfitting problem.

4. Preliminary Results

The model's performance was evaluated using accuracy and loss tracking over 20 epochs. The training accuracy improved steadily, reaching ~76%, while validation accuracy plateaued at ~61.7% early on. This gap indicates overfitting, meaning the model memorizes training data but does not generalize well to unseen samples. The loss graph highlights the same issue. Training loss decreased gradually, but validation loss increased sharply after epoch 6, stabilizing at 6.0+, suggesting that the model struggles to make meaningful generalizations beyond the training set.



Given these results, the model can produce segmentation masks, but its practical accuracy is limited.

5. Next Steps

To improve model performance and reduce overfitting, several adjustments will be explored. Instead of treating segmentation as a pixel-wise classification problem, we will shift to a region-based approach by replacing Binary Crossentropy with Dice Loss, which is better suited for evaluating segmentation quality. Additionally, regularization techniques such as dropout tuning and weight decay will be refined to improve generalization. Data augmentation strategies will be expanded to introduce greater variability in training samples. Finally, modifications to the model architecture will be evaluated to determine if changes in complexity or structure can lead to better results. Hopefully, these steps will help the model's ability to generalize while maintaining computational efficiency.