Aerial Image classification: identifying trees

Allan Henrique Kamimura

São Carlos School of Engineering, University of São Paulo

usp.br@usp.br

Abstract

Objectives: Training a Convolutional Neural Network model to identify trees from aerial imagery (satellite and drone)

Methods: Firstly, I used a pre-trained MobileNetV3Small model as a base and attached a fully connected layer for classification and trained it as a binary classification problem. For the training loss, I applied a class_weight based on the label distribution to get a better learning of the objective class. Also, I tried to optimize the decision threshold to maximize the F1 score. Lastly, I present an analysis over the Activation Map to help the interpretability of the classification model.

Results: I got around 0.9115 weighted F1 score on the training dataset and 0.8940 weighted F1 score on my testing dataset, with the threshold of 0.57, which I found to be optimal.

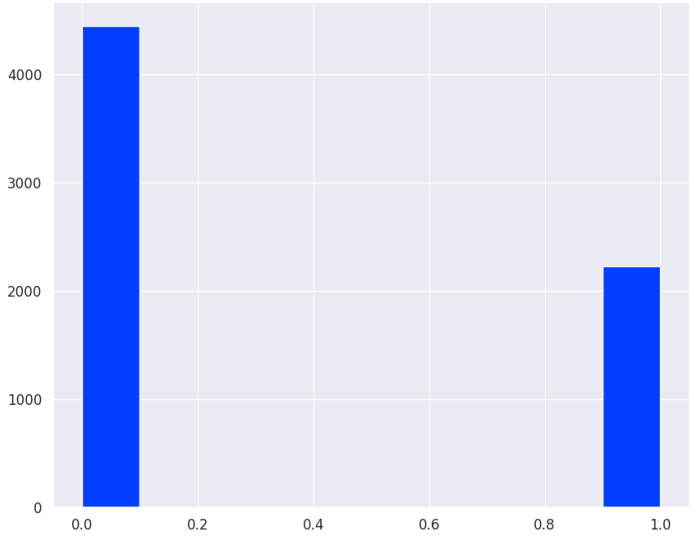
Conclusions: In this work, the model approach was nothing out of ordinary, nevertheless, the validation F1 score seems to be acceptable. I think the binary classification is not the best approach to the problem of identifying trees, but further analysis is needed in this topic.

Novelty: I think studying the effect of applying a different decision threshold to be rather unexpected, even though the improvement to the results were nothing much. And the model interpretability, which I think trying to better understand the model to be an essential part of modeling.

Report

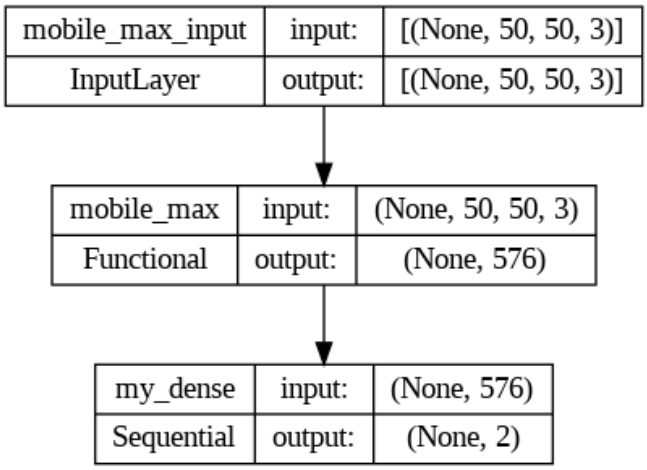
The data:

I started by taking a look at the class distribution and noticed that the class of interest (trees) represents 33% of the image dataset. To try and mitigate this, I took the inverse of the distribution value as a class_weight for the loss function.



The model:

For the base model, I used MobileNetV3Small from tensorflow applications[1], which is a standard model trained on imagenet, and added a standard fully-connected layer at the end. And fine-tuned the model with weighted categorical cross entropy loss and Adam optimizer.



[1] Instantiates the MobileNetV3Small architecture.
https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV3Small

Results Analysis:

	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90
soil	0.9471	0.9455	0.9418	0.9386	0.9380	0.9299	0.9235	0.9139	0.8928	0.8564	0.8168
tree	0.8734	0.8711	0.8747	0.8798	0.8796	0.8724	0.8597	0.8538	0.8443	0.8247	0.8129

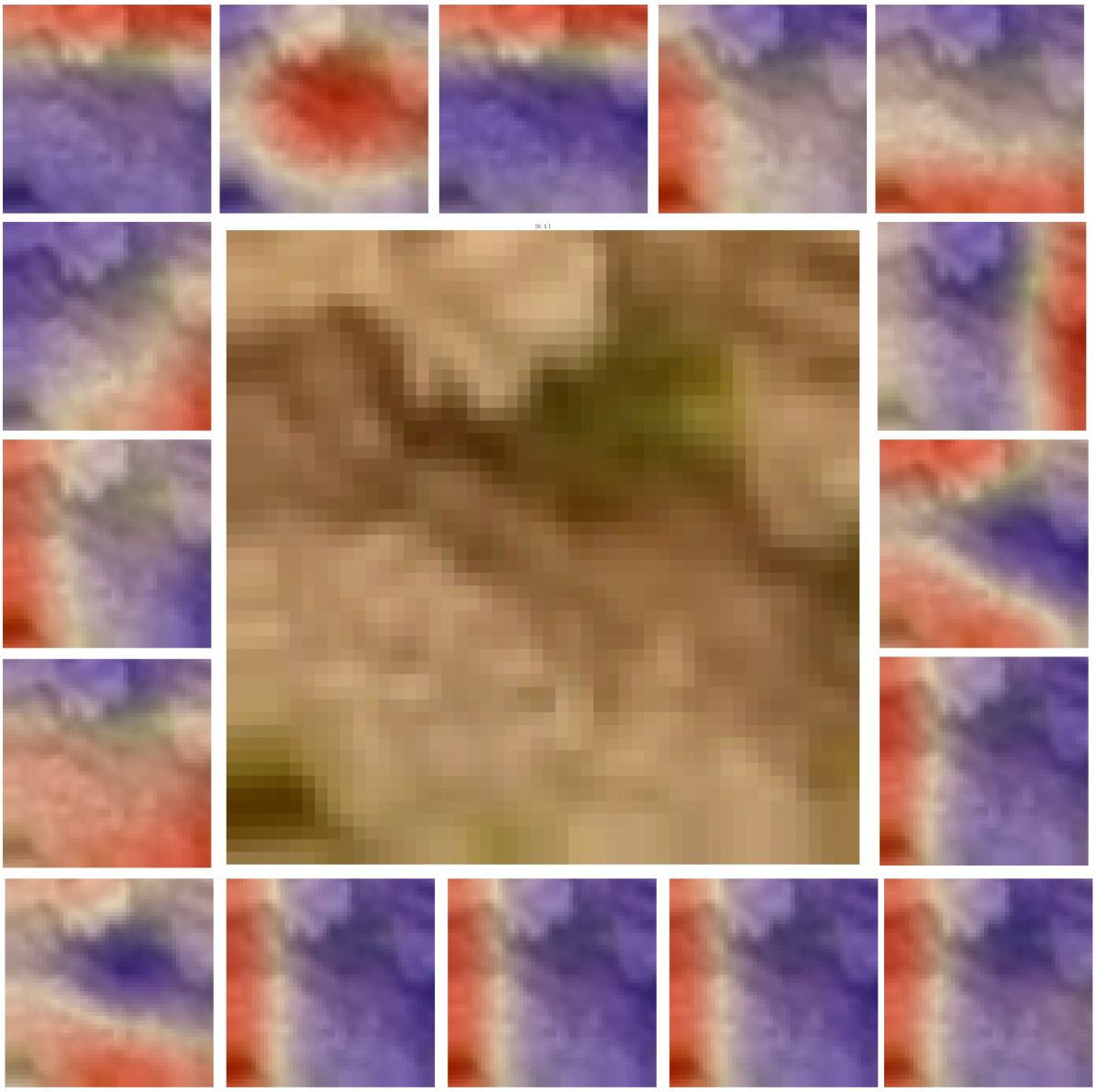
With the model trained, I analyzed the F1 score for each class separately over the epochs and, on the last epoch, with different decision thresholds, in which the value of 0.57 was giving me the highest score for the tree class, but may or may not work for other models. It's important to note that the analysis is over the validation data split.

Wrong predicts:

To learn more about the model performance, it's important to analyze the wrong predictions. Here we are going to take a look at the activation map of the early layers of the feature extraction layers, in other words, we are going to see which features are being used for the classification.

Another method, that works better with higher resolution images, is called Gradient Class Activation Map, which uses the final output (global average pooling layer) from the feature extraction model to compute the gradients which respect to the classification output and determine a weight of importance of the extracted features. This last method is not going to be used here, because our dataset uses very low resolution images.

Feature map:



Here, we have the output from the earlier layers from the feature extraction model, the image in the center is labeled as tree and is wrongly classified as soil. The red region defines the features extracted from the image.

Using the output from the global average pooling layer, we can get the feature weight being passed to the remaining out of the network and taking the dot product between the map output and the weight output, we can get a general idea of which regions of the image have a higher influence over the classification. From this, we can see that the border of the image has the higher impact on the classification, rather than the center region, where the tree lies.

Conclusion:

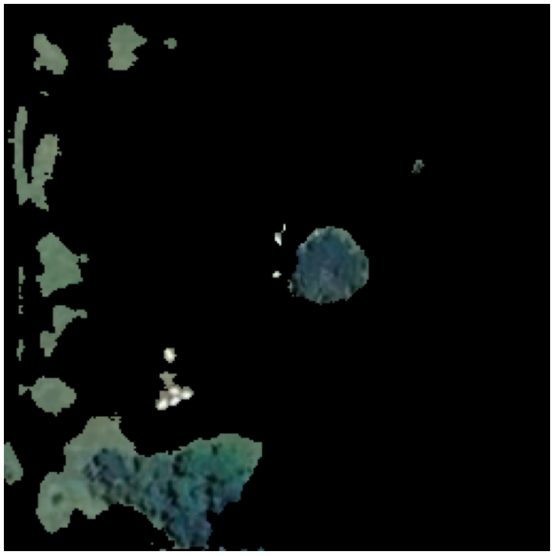
Using a pre-trained model to extract features and appending a fully connected layer seems to be the most general solution for the binary classification problem. I think the results are quite acceptable, the resolution of the data is pretty bad and in most cases even I am unsure if there's really a tree there.

I think the next steps are to try to use a different approach than the binary classification, such as a semantic segmentation model or an object detection model. The first step to do this, is to create a common dataset and to define a common metric, that way we can make a proper comparison between the models.

Bonus:

Segmentation model:

I also trained a segmentation model, on a different dataset, which uses an UNET to extract the features and a depthwise convolution to give out the pixel classification



Object Detection model:

And also, I trained an object detercion model, on yet another dataset, which uses a yoloV8 model to make the prediction. The output is not as good as expected because the dataset was not exactly “tree detection”, but rather “detect this species of tree”.

References:

1. Thresholding Classifiers to Maximize F1 Score *Zachary Chase Lipton and Charles Elkan and Balakrishnan Narayanaswamy* **2014, https://arxiv.org/abs/1402.1892**
2. Searching for MobileNetV3 *Andrew Howard and Mark Sandler and Grace Chu and Liang-Chieh Chen and Bo Chen and Mingxing Tan and Weijun Wang and Yukun Zhu and Ruoming Pang and Vijay Vasudevan and Quoc V. Le and Hartwig Adam* **2019, https://arxiv.org/abs/1905.02244v5**
3. Learning Deep Features for Discriminative Localization *Bolei Zhou and Aditya Khosla and Agata Lapedriza and Aude Oliva and Antonio Torralba* **2015, https://arxiv.org/abs/1512.04150**
4. Network In Network *Min Lin and Qiang Chen and Shuicheng Yan* **2014, https://arxiv.org/abs/1312.4400v30**
5. U-Net: Convolutional Networks for Biomedical Image Segmentation *Olaf Ronneberger and Philipp Fischer and Thomas Brox* **2015, https://arxiv.org/abs/1505.04597**
6. Real-Time Flying Object Detection with YOLOv8 *Dillon Reis and Jordan Kupec and Jacqueline Hong and Ahmad Daoudi* **2023, https://arxiv.org/abs/2305.09972**