Multi label classification of tree species in forest satellite imagery

Miloš Medić

June 2023.

1 Abstract

As ecological and agricultural fields of study continue to advance, the need for related data increases. Here we study multi-label tree species classification for forest satellite imagery. Many similar studies have been conducted, but they mostly focus on specific regions, sets of image bands, and classes. We describe a process that can be generalized for any region with sufficient training data, and which will predict all significant tree species within the image, not just classify it to the most prominent one. For this we used the Sentinel-2 dataset, and the United States Forest Service (USFS) public database. The training was performed on images of Maine, with a binary accuracy of 82%. A conventional Convolutional Neural Network (CNN) architecture was used, as well as ResNet-50, both yielding similar results.

2 Introduction

With rising ecological concerns, the field of remote sensing has seen increased interest. As satellite data has become more widely accessible, remote sensing with satellite imagery using various Machine Learning (ML) methods has become a particular target of interest. Specifically we look at tree species classification of satellite imagery. Most of the studies concerning this perform single-label classification (i.e. classify the image to the most dominant species on it). Some exploit temporal data [1, 2], most hyperspectral data [1–7]. Many such studies do not have generalizable results, as they focus on specific regions with specific tree species with specific types of data [8]. Many different approaches have also been tried, such as: Random Forest Classifiers [1, 3, 7], Gaussian Mixture Modelling [1], k Nearest Neighbors [1], Support Vector Machines [1, 7], Neural Networks [6, 9]. Some using per-pixel classification methods [6, 7]. We applied two approaches: a simple Convolutional Neural Network (CNN), and the ResNet-50 [10] architecture. Both perform multi-label classification, outputting all the detected tree species in a particular image. We used Sentinel-2 satellite imagery, focusing only on the visible spectrum, the RGB channels. Although accuracy could probably be improved with more bands, this would make the approach less generalizable. For the land cover data, we used the United States Forest Service Forest Inventory and Analysis database (USFS FIA). Training was done on Maine, as it contains abundant forests on the USA territory. The source code for the project can be found at `https://github.com/hiddenMedic/forestClassifier`.

3 Data Preparation

3.1 Tree Data

The USFS FIA dataset was chosen due to its large number of data points, and its good documentation. The Maine tree data file (ME_TREE.csv) contains over 920 thousand instances. While there are many columns in the dataset, we are only interested in PLOT - the plot number the tree is located in, and SPCD - the species code of the tree. We will match the

species code to its name using the USFS MasterSpecies dataset. For simplicity, we assume that each tree is located in the latitute and logitude coordinates given for its plot. Now that we know the coordinates of each tree, we can assign them to their corresponding images. At this step, we make two observations: the tree data should match the image data temporally, and trees that are of a very low frequency species, if we do not wish to focus on that species specifically, will only inhibit the model in learning. Thus, we only consider trees that are within some temporal range of the moment the image was taken (in particular, the value of 5 years was used), and whose species appears above some frequency threshold (in particular, the threshold of a minimum of 1000 instances was used). For example, for an image in 2021, after selecting only the temporally viable instances, we are left with 207446 (22.4% out of the starting 923939) instances, and after further taking only those above the species frequency threshold we are left with 201720 instances, which is 21.8% of the total number of instances. This procedure is performed by the `make_dataset.py` and `merge_data.py` files in the source code. This produces a dataset of 8381 images (more on this in 3.2). For each of these images, we know how many trees of a particular species have been recorded in it. Currently there are twenty different species associated with each image. Next, we implement a threshold that says how many trees of a certain species are needed in an image, for that image to "contain" that species. This threshold was chosen to be 7. Here we also remove tree species columns which are contained in a very low number of images (less than 5%), and images that have no trees associated with them (though this choice is arbitrary). This step is performed by `data_cleanup.py`, and we are left with 7796 data points, and 13 different classes, specifically: balsam fir, red maple, red spruce, northern white-cedar, paper birch, American beech, yellow birch, eastern white pine, eastern hemlock, sugar maple, quaking aspen, northern red oak and white ash.

To confirm that the plots in the dataset cover all of Maine, the script `info/plot_coords.py` was written, see Figure 3.1. `info/inventoryyear_histogram.py` shows an overview of how many trees were recorded each year, see Figure 3.2. `species_frequencies.py` shows a histogram of species frequencies in Figure 3.3, and we can see that the data is very unbalanced. The USFS FIADB guide mentions that where there is missing location data, the latitude and longitude of the country centroid will be put. The script `info/check_centroid.py` was put in place to check how many occurances of this exist. It turns out that there are in fact no plots which have the US centroid as their coordinates. Another thing that should be noted is section 1.2.2 (Plot Locations) in the guide, it explains that, to protect the privacy of land owners, the plot coordinates have been fuzzed and swapped. This means that the real coordinates of the centers of the plots are within 0.5 miles of the coordinates given (1 mile for older data), and that some plot coordinates have been swapped (up to 20% within the same county). Although it does note that plots have been swapped on a similarity metric, this will result in some inconsistencies in the data labels.

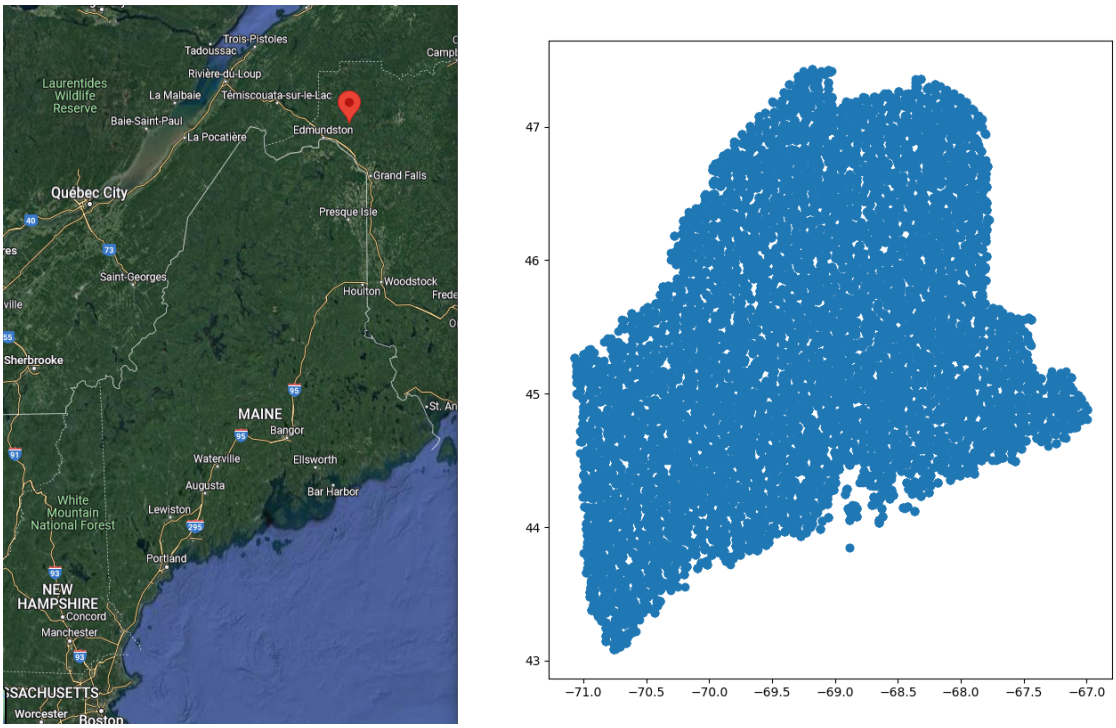

Figure 3.1: *A google maps image of Maine, next to a scatterplot of plots in the dataset arranged by their latitude and longitude.*

3.2 Image Data

While there was consideration for other data sources (like the USGS Earth Explorer), Sentinel-2 imagery from the Copernicus hub turned out to have the highest quality images, while allowing for easiest downloads of those images as well. Out of a total of nineteen potential images that cover Maine, nine that are square were selected. Each of these images covers a large landmass, their resolution is around 10.000 by 10.000, each pixel representing ten meters squared. We will break this big images up into smaller images (frames). There are two main motivations for this: we will have more training data, we will allow our model to predict a smaller area. The resolution of 100 by 100 pixels was chosen for the frames, so each covers 1km^2 of land area. We are given the coordinates of the corners of each image, so we can easily calculate the frames' coordinates by assuming that each frame covers the same latitudal and longitudal distance. In reality does is not true, because the latitudal and longitudal lines form curves on a two-dimensional image of the Earth, but on our scale this will not pose an issue. The mentioned segmentation of the images is performed by `image_breakup.py`. After matching the plots and frames, as explained in the previous section, we get a total of 1949 frames. To increase the input data size, and make the model more robust, data augmentation was performed. For each frame, three additional transformations were applied to it. These transformations are randomly sampled from the possible seven transformations belonging to the dihedral D_4 group (except for the identity). The

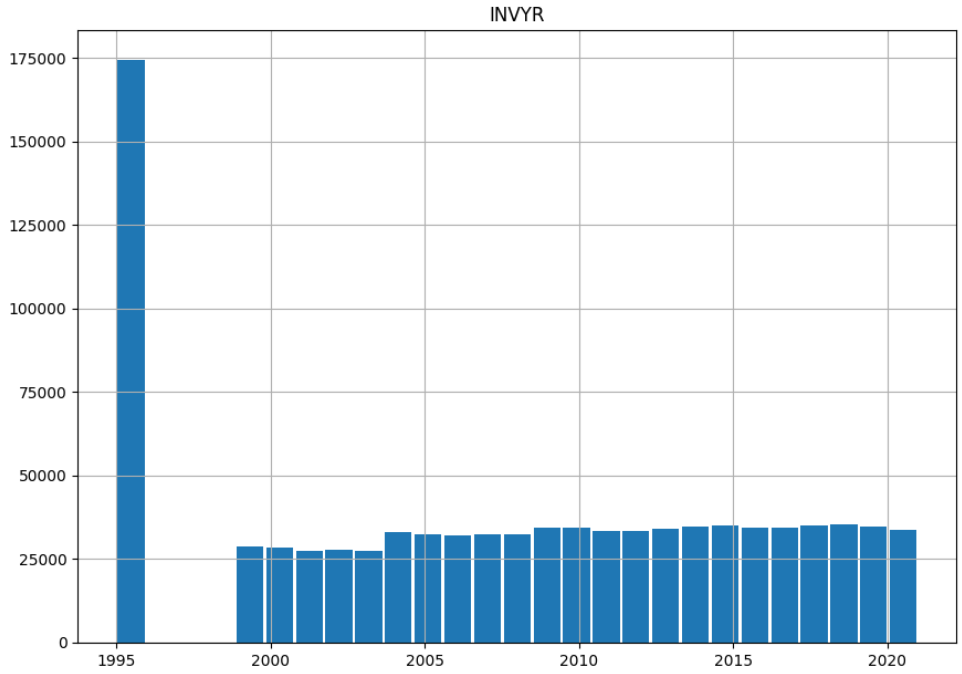

Figure 3.2: *A histogram that shows how many trees were recorded each year in ME_TREE.csv.*

image augmentation step was performed by `images_apply_aug.py`. This gives us our total of 7796 frames.

4 Learning

Now that we have labeled input data, we can move on to the machine learning. Different variations of model architecture were tried, and the best one seemed to be a simple sequential CNN with 4 convolution layers and two dense layers at the end. Max pooling, batch normalization and dropout layer were also used. The model architecture can be seen in `learn.py` and in Figure 4.1. The weights for the model are saved in `models/base_model`. The ResNet-50 architecture was also built, and performed with similar results to the simpler model. Interestingly, even as the models overfit, the validation accuracy did not suffer. The binary accuracy graphs can be seen in Figures 4.2 and 4.3. Graphs of other related metrics can be seen in `figures/batch_64_aug_basic` and `figures/aug_resnet` respectively. Both models had a sigmoid function on the last layer (common for multi-label classification), used the Adam optimizer, binary crossentropy as the loss function, had a batch size of 64, and were trained for 60 epochs. The final validation binary accuracy achieved for the simple model was 83.01%, and 82.23% for the ResNet-50 model.

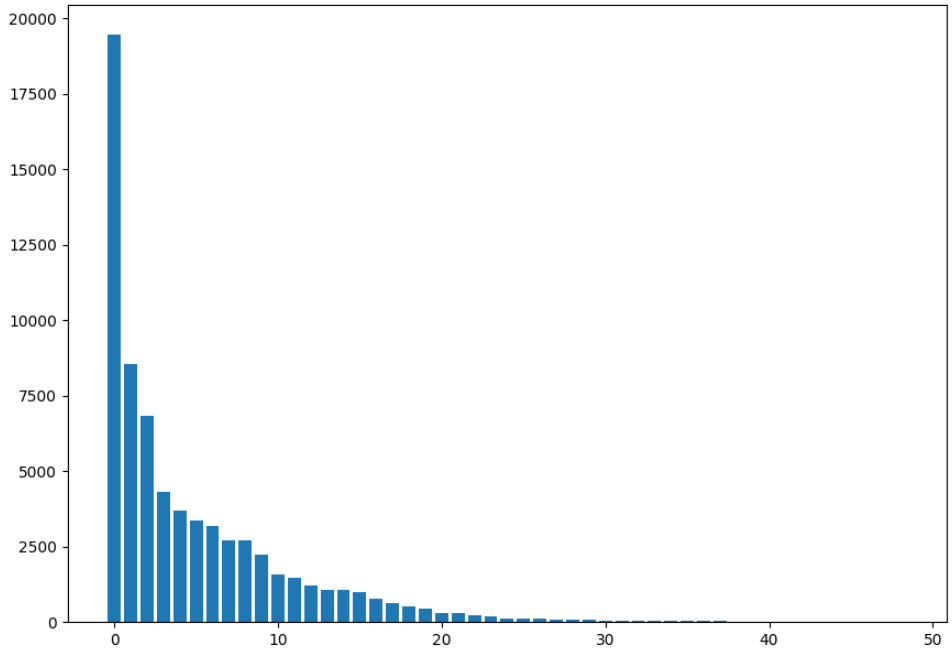

Figure 3.3: *A sorted histogram of tree species occurences in ME_TREE.csv.*

```
model = Sequential()

model.add(Conv2D(filters=16, kernel_size=(5, 5), activation="relu", input_shape=(FRAME_SIZE, FRAME_SIZE, FRAME_DEPTH)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(filters=32, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Conv2D(filters=64, kernel_size=(5, 5), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Conv2D(filters=64, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(CLASSES, activation='sigmoid'))
```

Figure 4.1: *Python code for the tensorflow library for constructing the model architecture.*

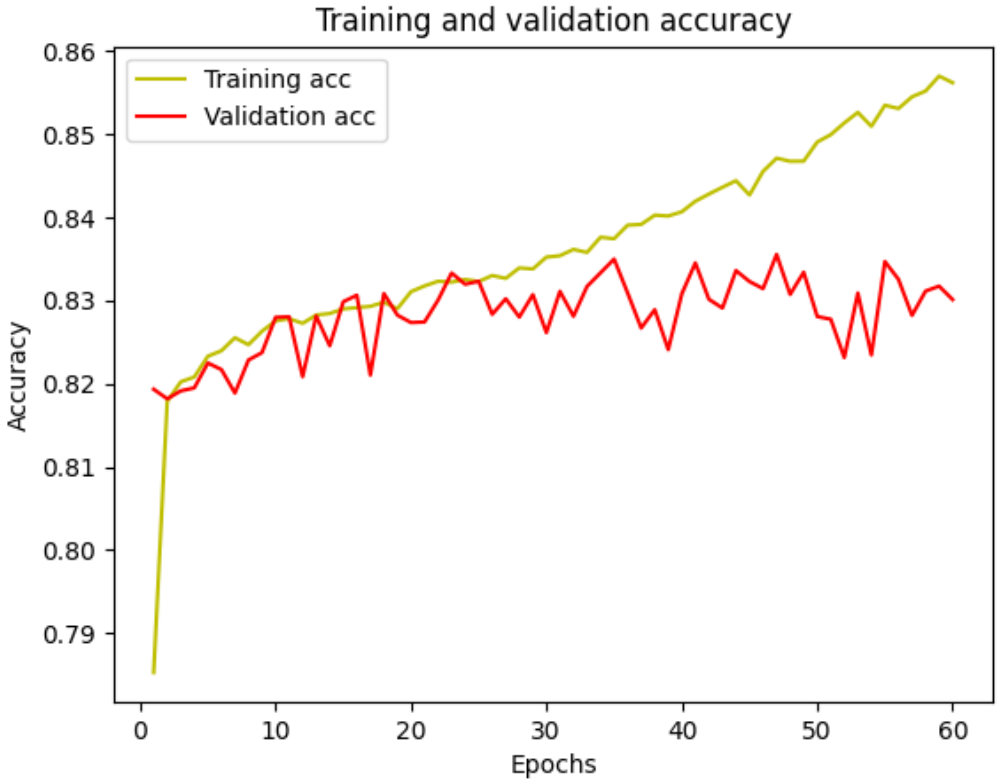

Figure 4.2: *Binary accuracy per epoch graph for the training and validation sets for the simple CNN model.*

To see how well our model does on images of a different state, we performed prediction on some frames of Alabama. The dataset for Alabama was generated in the same manner as already described. Unfortunately, there was only one tree species which appeared in both the Maine and the Alabama dataset - red maple. This greatly hinders our models ability to predict tree species from Alabama, and an example prediction can be seen in Figure 4.4. Example predictions for Maine and Serbia (with unlabeled data) can be seen in the `figures/predicts` folder. The predictions are generated by the `predict.py` and `predict_unknown.py` files respectively.

Figure 4.3: *Binary accuracy per epoch graph for the training and validation sets for the ResNet-50 model*

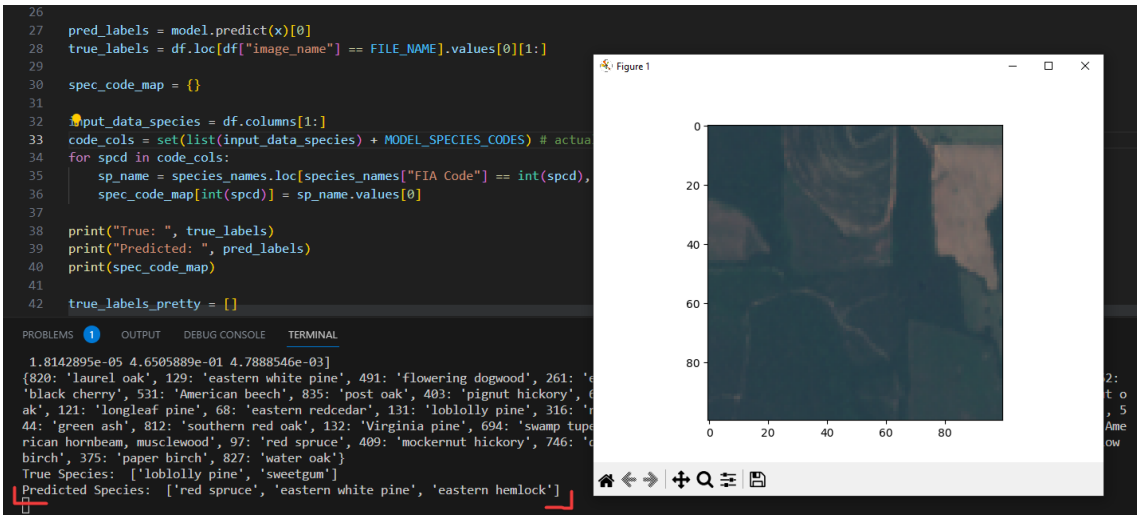

Figure 4.4: *Prediction of the simple model performed on some frame of an image of Alabama.*

5 Conclusion

As the field of remote sensing continues to grow, the methodologies put forth become better, but also more complex. Here we provide a simple solution with each step of the process clearly laid out. Our method can be easily used in other geographic regions with minimal available data, as only the RGB channels of the image, and the plot locations are needed. It should be noted that one should take care when performing predictions on a region of unknown tree species, as there may be species that were not included in the training dataset.
Some improvements to the process described here could be made. Images of higher resolution would greatly help; although these exist and some are freely available, there is still not enough of them freely available for model training. Ideally, data that is not swapped and fuzzed should be used. Image segmentation could be extended to work for non-square satellite images, and more precise latitude and longitude coordinates could be calculated. Ensamble or transfer learning could be performed to add aditional image bands in regions where they are available, or the model could simply be retrained.

References

[1] David Sheeren, Mathieu Fauvel, Veliborka Josipović, Maïlys Lopes, Carole Planque, Jérôme Willm, and Jean-François Dejoux. Tree species classification in temperate forests using formosat-2 satellite image time series. *Remote Sensing*, 8(9), 2016.

[2] Olga Grigorieva, Olga Brovkina, and Alisher Saidov. An original method for tree species classification using multitemporal multispectral and hyperspectral satellite data. *Silva Fennica*, 54(2), 2020.

[3] Markus Immitzer, Clement Atzberger, and Tatjana Koukal. Tree species classification with random forest using very high spatial resolution 8-band worldview-2 satellite data. *Remote Sensing*, 4(9):2661–2693, 2012.

[4] Minfei Ma, Jianhong Liu, Mingxing Liu, Jingchao Zeng, and Yuanhui Li. Tree species classification based on sentinel-2 imagery and random forest classifier in the eastern regions of the qilian mountains. *Forests*, 12(12), 2021.

[5] Bin Zhang, Lin Zhao, and Xiaoli Zhang. Three-dimensional convolutional neural network model for tree species classification using airborne hyperspectral images. *Remote Sensing of Environment*, 247:111938, 2020.

[6] Svetlana Illarionova, Alexey Trekin, Vladimir Ignatiev, and Ivan Oseledets. Neural-based hierarchical approach for detailed dominant forest species classification by multispectral satellite imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:1810–1820, 2021.

[7] Martina Deur, Mateo Gašparović, and Ivan Balenović. Tree species classification in mixed deciduous forests using very high spatial resolution satellite imagery and machine learning methods. *Remote Sensing*, 12(23), 2020.

[8] Fabian Ewald Fassnacht, Hooman Latifi, Krzysztof Stereńczak, Aneta Modzelewska, Michael Lefsky, Lars T. Waser, Christoph Straub, and Aniruddha Ghosh. Review of studies on tree species classification from remotely sensed data. *Remote Sensing of Environment*, 186:64–87, 2016.

[9] Hui Li, Baoxin Hu, Qian Li, and Linhai Jing. Cnn-based individual tree species classification using high-resolution satellite imagery and airborne lidar data. *Forests*, 12(12), 2021.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] Arvid Axelsson, Eva Lindberg, Heather Reese, and Håkan Olsson. Tree species classification using sentinel-2 imagery and bayesian inference. *International Journal of Applied Earth Observation and Geoinformation*, 100:102318, 2021.