Geospatial Big Data Analysis - Lab 3

Konstantinos Papadakis - Student of MSc Data Science and Machine Learning - 03400149

Abstract

In this document we will discuss and evaluate various data loading and machine learning approaches for the classification on the HyRANK dataset.

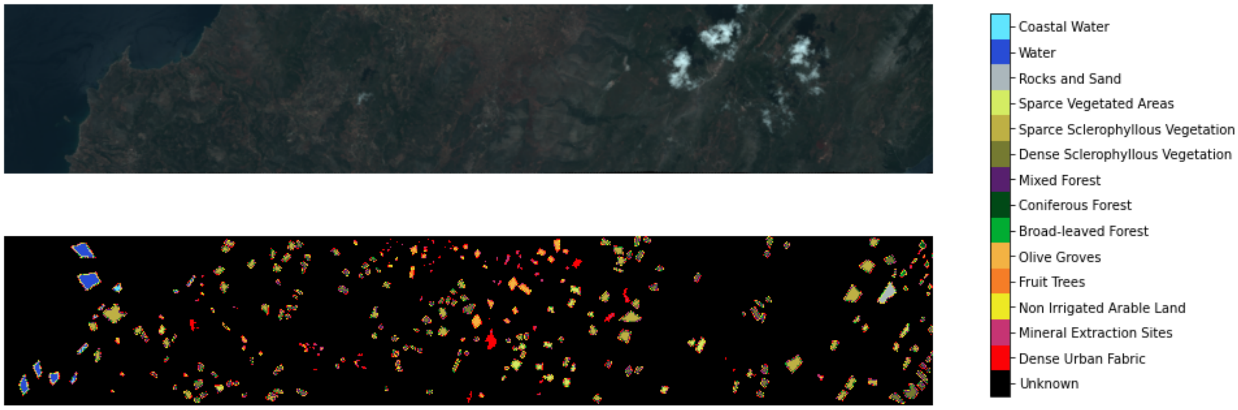
We consider three approaches to image classification:

- A *pixel-based* approach where we classify each pixel one by one, without using any information from its surrounding pixels.
- A *patch-based* approach where we again classify each pixel one by one, but this time instead of the pixel itself, we use a small surrounding image patch of that pixel.
- A *semantic segmentation* approach, where we decompose the image into medium size patches and perform image segmentation on each one of them.

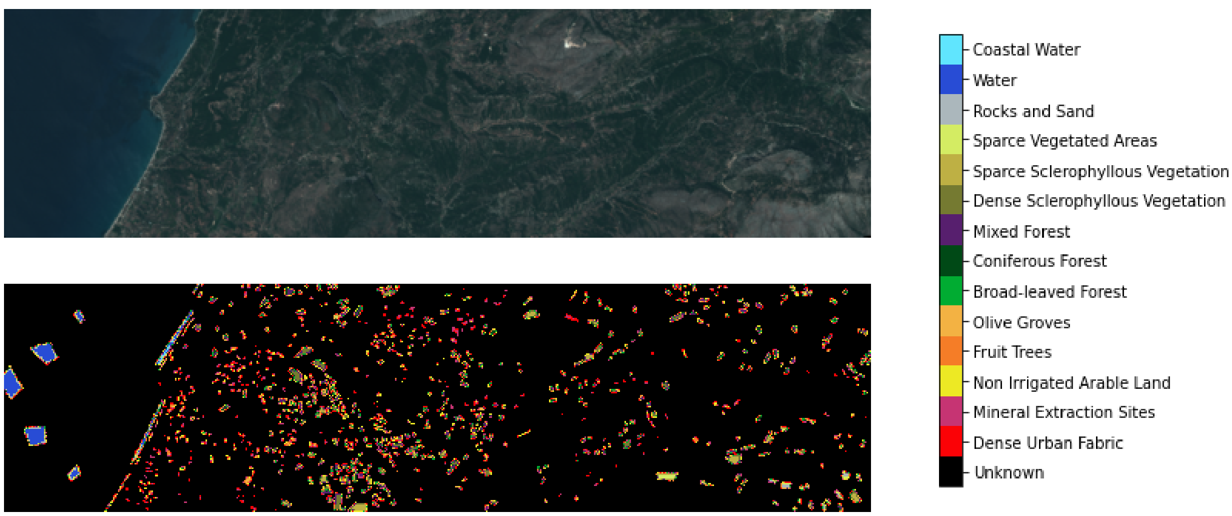
The Dataset

The HyRANK dataset is made up of 5 large images composed of 176 channels each. Of those images, 2 are (partially) labeled (Dioni, Loukia) and 3 are unlabeled (Erato, Kirki, Nefeli). Dioni and Loukia are displayed below. We will train our models on the labeled images and predict the unlabeled ones.

Dioni



Loukia



Training and Evaluation

The training is done with the main.py file which utilizes the lab3 package containing the modules datasets, models and utils. The outputs of main.py can be found in the output directory. The final inference results are included in the inference_results directory.

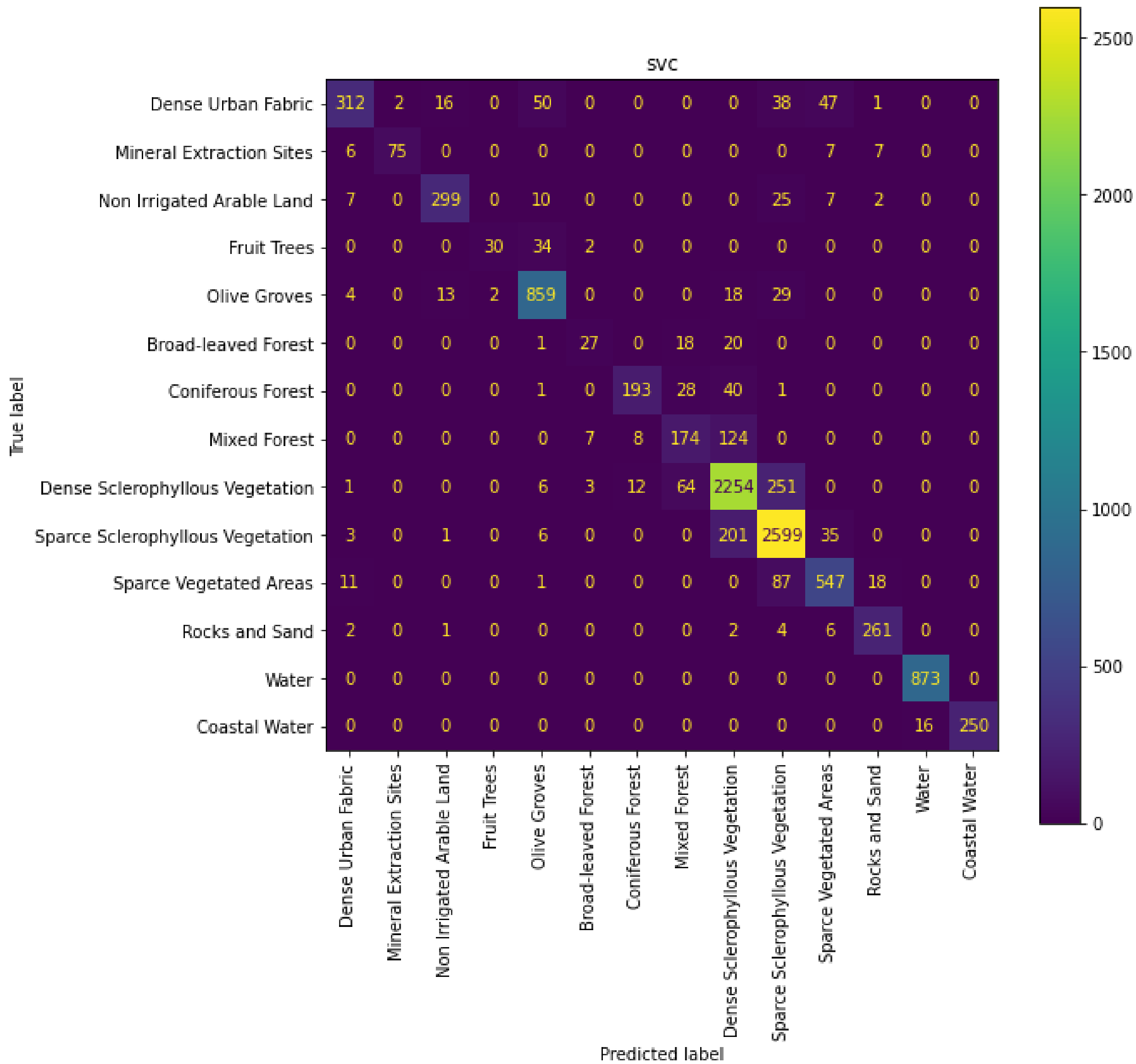
Pixel Based Approach

We create a pixel dataset composed only of labeled pixels. We train and test a support vector machine, a random forest, and a multilayer perceptron. For the support vector machine and the random forest we use a 70-30 train-test split, while for the multilayer perceptron we use a 70-15-15 train-val-test split. **This split produces very biased evaluation results that should only be used to compare between pixel-based approaches**. The results are biased because image data are approximately locally constant. This means that neighboring pixels tend to be roughly the same, which makes a random train-test split produce effectively the same dataset. Considering the results, all algorithms performed roughly the same, with the random forest having a slight edge as it is often the case.

SVM

The hyperparameters used were C=1.0 and a radial basis kernel.

	precision	recall	f1-score	support
Dense Urban Fabric	0.90	0.67	0.77	466
Mineral Extraction Sites	0.97	0.79	0.87	95
Non Irrigated Arable Land	0.91	0.85	0.88	350
Fruit Trees	0.94	0.45	0.61	66
Olive Groves	0.89	0.93	0.91	925
Broad-leaved Forest	0.69	0.41	0.51	66
Coniferous Forest	0.91	0.73	0.81	263
Mixed Forest	0.61	0.56	0.58	313
Dense Sclerophyllous Vegetation	0.85	0.87	0.86	2,591
Sparce Sclerophyllous Vegetation	0.86	0.91	0.88	2,845
Sparce Vegetated Areas	0.84	0.82	0.83	664
Rocks and Sand	0.90	0.95	0.92	276
Water	0.98	1.00	0.99	873
Coastal Water	1.00	0.94	0.97	266
accuracy			0.87	10,059
macro avg	0.88	0.78	0.81	10,059
weighted avg	0.87	0.87	0.87	10,059

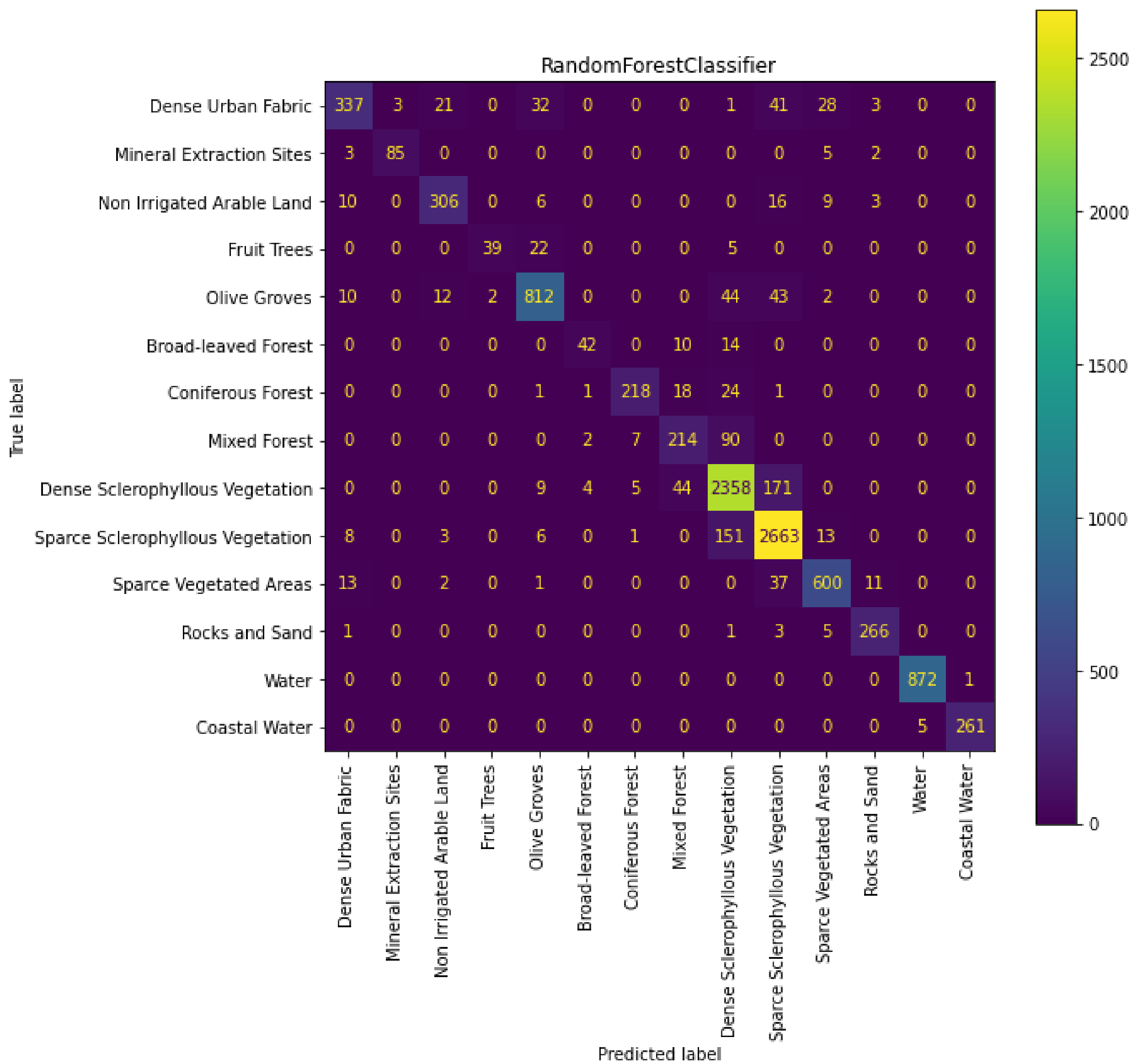


Random Forest

Random forest composed of 100 decision trees.

	precision	recall	f1-score	support
Dense Urban Fabric	0.88	0.72	0.79	466
Mineral Extraction Sites	0.97	0.89	0.93	95
Non Irrigated Arable Land	0.89	0.87	0.88	350
Fruit Trees	0.95	0.59	0.73	66
Olive Groves	0.91	0.88	0.90	925
Broad-leaved Forest	0.86	0.64	0.73	66

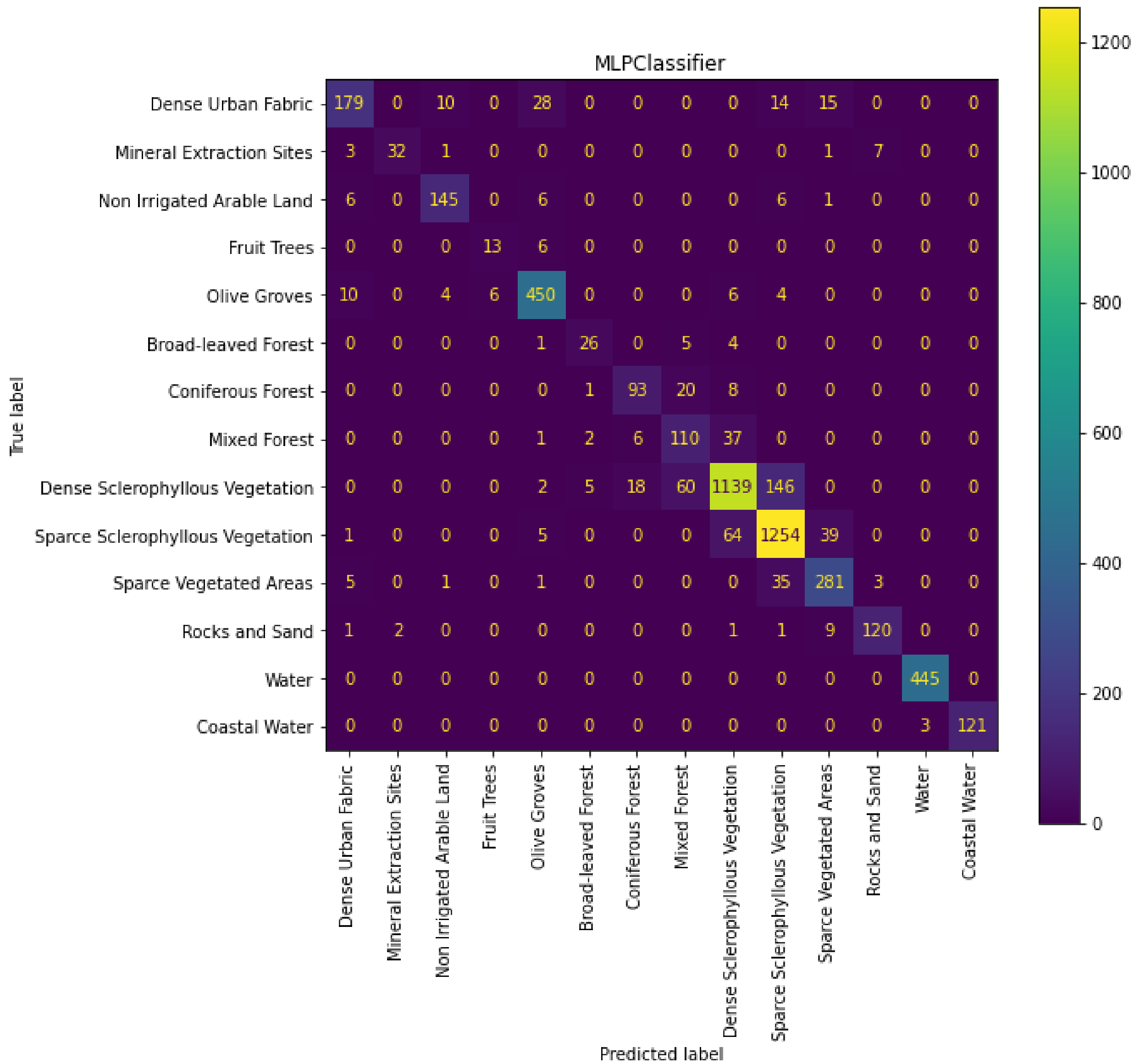
Coniferous Forest	0.94	0.83	0.88	263
Mixed Forest	0.75	0.68	0.71	313
Dense Sclerophyllous Vegetation	0.88	0.91	0.89	2,591
Sparce Sclerophyllous Vegetation	0.90	0.94	0.92	2,845
Sparce Vegetated Areas	0.91	0.90	0.90	664
Rocks and Sand	0.93	0.96	0.95	276
Water	0.99	1.00	1.00	873
Coastal Water	1.00	0.98	0.99	266
accuracy			0.90	10,059
macro avg	0.91	0.84	0.87	10,059
weighted avg	0.90	0.90	0.90	10,059



MLP

A simple feed forward neural network with 3 hidden layers and with relu as activation function. Dropout is performed between the hidden layers, and L2 regularization is also used. The network was trained for 300 epochs, with the best validation loss achieved on epoch 291.

	precision	recall	f1-score	support
Dense Urban Fabric	0.87	0.73	0.79	246
Mineral Extraction Sites	0.94	0.73	0.82	44
Non Irrigated Arable Land	0.90	0.88	0.89	164
Fruit Trees	0.68	0.68	0.68	19
Olive Groves	0.90	0.94	0.92	480
Broad-leaved Forest	0.76	0.72	0.74	36
Coniferous Forest	0.79	0.76	0.78	122
Mixed Forest	0.56	0.71	0.63	156
Dense Sclerophyllous Vegetation	0.90	0.83	0.87	1,370
Sparce Sclerophyllous Vegetation	0.86	0.92	0.89	1,363
Sparce Vegetated Areas	0.81	0.86	0.84	326
Rocks and Sand	0.92	0.90	0.91	134
Water	0.99	1.00	1.00	445
Coastal Water	1.00	0.98	0.99	124
accuracy			0.88	5,029
macro avg	0.85	0.83	0.84	5,029
weighted avg	0.88	0.88	0.88	5,029



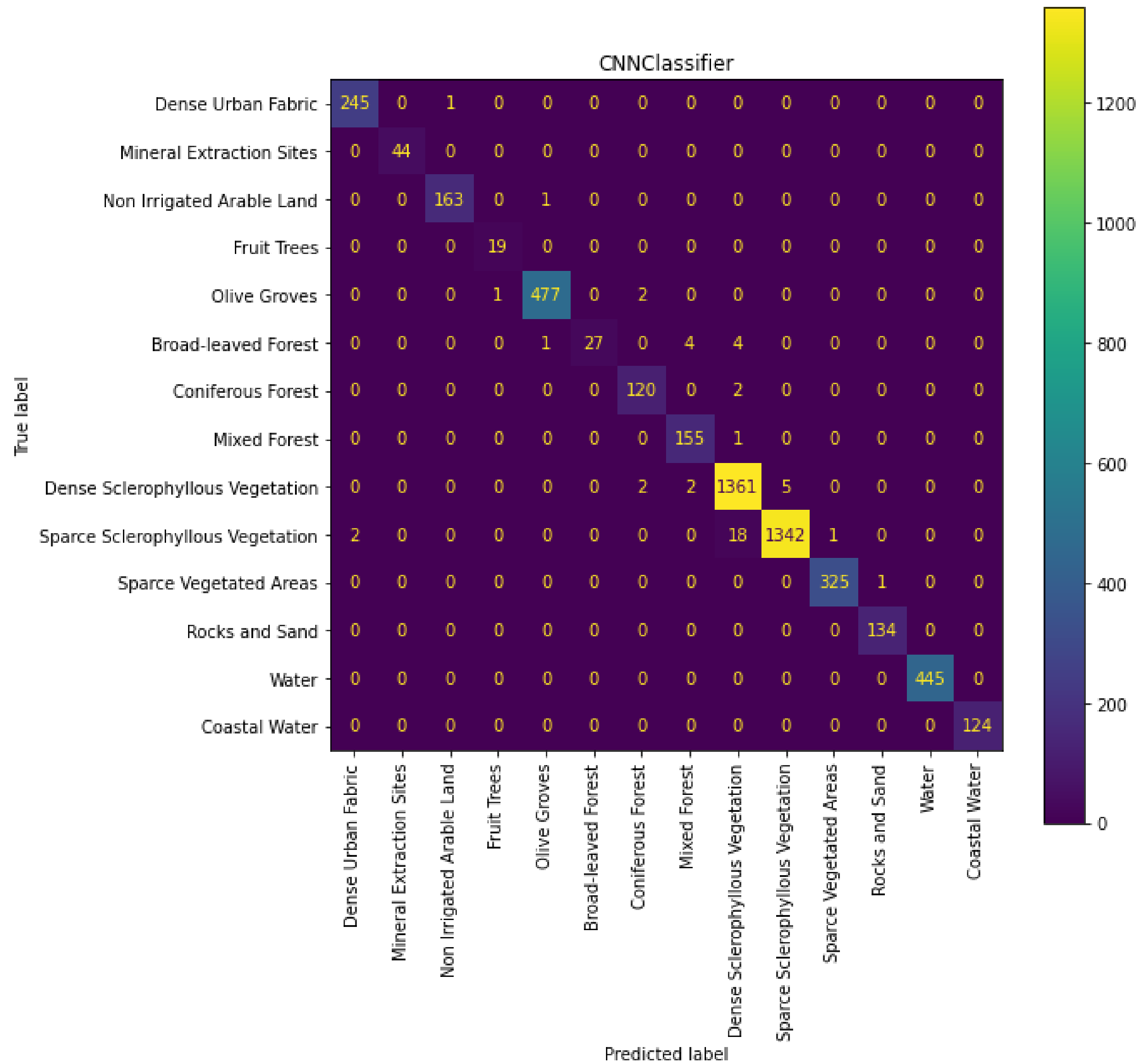
Patch Based Approach

In this approach, for each pixel, we slice from the original image the 15 by 15 surrounding patch that is centered at the pixel. If a patch is out of the image boundary, we zero-pad it equally on all sides. While training, we randomly flip the patch vertically and rotate it by 0, 1, 2, or 3 right angles, as a form of data augmentation. See PatchDatasetPostPad and flip_and_rotate in lab3.datasets. The models are trained and evaluated using a 70-15-15 train-val-test split. **This split produces very biased results (considerably more biased than even the split in pixel-based classification) that should only be used to compare between patch-based approaches**. They are biased because neighboring pixels produce essentially the same patch, which means that our training and test sets are far from independent - they are essentially the same dataset.

CNN

We train a small residual convolutional network (see CNNClassifier in lab3.models) composed of 4 residual blocks, with each block containing 2 convolutional layers with batch normalization. This evaluation of the model produced seemingly “extremely good” results, but as discussed earlier, it is most probable that the true results are in reality much lower. The network was trained for 200 epochs, with the best validation loss achieved on epoch 186.

	precision	recall	f1-score	support
Dense Urban Fabric	0.99	1.00	0.99	246
Mineral Extraction Sites	1.00	1.00	1.00	44
Non Irrigated Arable Land	0.99	0.99	0.99	164
Fruit Trees	0.95	1.00	0.97	19
Olive Groves	1.00	0.99	0.99	480
Broad-leaved Forest	1.00	0.75	0.86	36
Coniferous Forest	0.97	0.98	0.98	122
Mixed Forest	0.96	0.99	0.98	156
Dense Sclerophyllous Vegetation	0.98	0.99	0.99	1,370
Sparce Sclerophyllous Vegetation	1.00	0.98	0.99	1,363
Sparce Vegetated Areas	1.00	1.00	1.00	326
Rocks and Sand	0.99	1.00	1.00	134
Water	1.00	1.00	1.00	445
Coastal Water	1.00	1.00	1.00	124
accuracy			0.99	5,029
macro avg	0.99	0.98	0.98	5,029
weighted avg	0.99	0.99	0.99	5,029

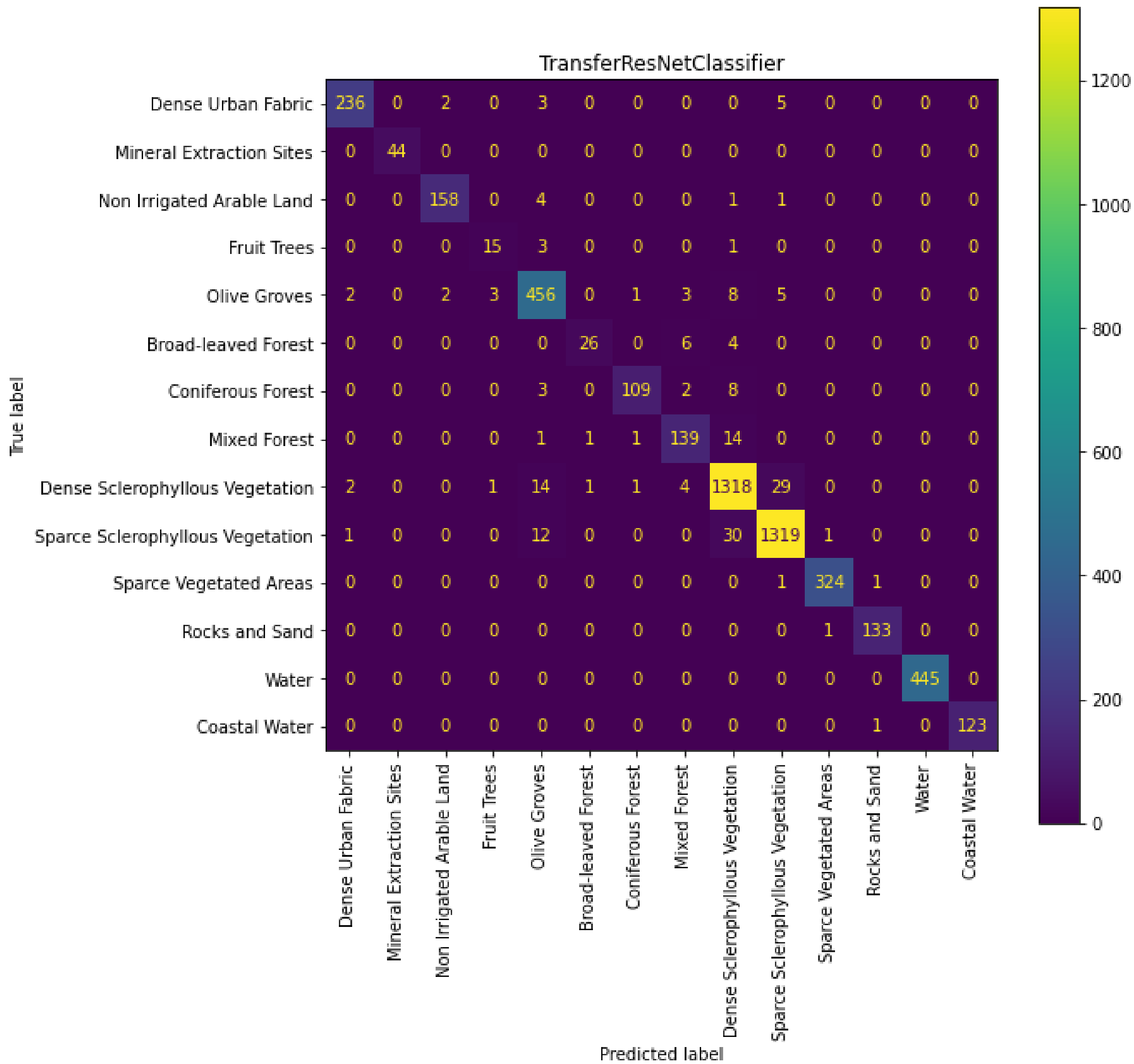


Transferred ResNet18

As an alternative approach, we attempt transfer-learning from a Resnet18 pre-trained on ImageNet, with the model's head unfrozen. To achieve this we select only the RGB channels from our original image. Even though ImageNet doesn't contain any images similar to satellite images and the channels were only 3 instead of 176, the model achieved decent results when compared to the other resnet. The network was trained for 200 epochs, with the best validation loss achieved on epoch 189.

	precision	recall	f1-score	support
Dense Urban Fabric	0.98	0.96	0.97	246
Mineral Extraction Sites	1.00	1.00	1.00	44

Non Irrigated Arable Land	0.98	0.96	0.97	164
Fruit Trees	0.79	0.79	0.79	19
Olive Groves	0.92	0.95	0.93	480
Broad-leaved Forest	0.93	0.72	0.81	36
Coniferous Forest	0.97	0.89	0.93	122
Mixed Forest	0.90	0.89	0.90	156
Dense Sclerophyllous Vegetation	0.95	0.96	0.96	1,370
Sparce Sclerophyllous Vegetation	0.97	0.97	0.97	1,363
Sparce Vegetated Areas	0.99	0.99	0.99	326
Rocks and Sand	0.99	0.99	0.99	134
Water	1.00	1.00	1.00	445
Coastal Water	1.00	0.99	1.00	124
accuracy			0.96	5,029
macro avg	0.95	0.93	0.94	5,029
weighted avg	0.96	0.96	0.96	5,029



Semantic Segmentation Approach

In this approach we use an encoder-decoder fully convolutional architecture in the form of UNet (see UNetClassifier in lab3.datasets).

The dataset for this approach is constructed by running a *sliding window* of a specified *size*, and with a specified *stride* over the image. If the sliding window can't cover the entire image, we can optionally *pad* the original image equally on all sides with zeros, so that the window can cover all the contents of the original image. The implementation creates a sliding window "lazily" by calculating *only the window coordinates and then slicing* on the original image, which means that no extra memory is required. The dataset is represented by CroppedDataset in lab3.datasets. During training we also flip and rotate by 0, 1, 2, or 3 right angles both the image and the target, using the same transformation on both, as a form of data augmentation.

To evaluate our model fairly, we *partition* the image by using equal window and stride sizes, specifically 64. This allows us to reduce the statistical dependency between train and test samples, as there is no pixel overlap, and pixels near the center of each 64 by 64 window are not very dependent on pixels from other windows. This of course significantly reduces the number of samples that can be used to train the model, which means that the results are most likely going to be an underestimate of what can be achieved. The split that was used was a 70-15-15 train-val-test one.

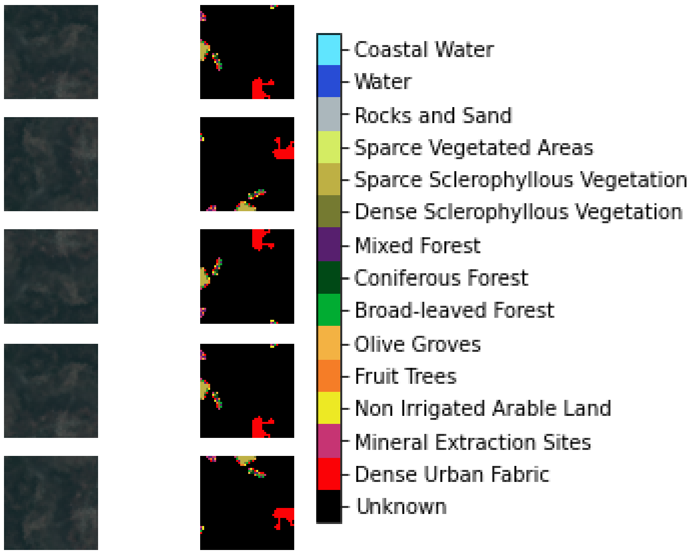
A different strategy is used when we need to train but not evaluate the model. In this case, we allow overlap between windows by selecting a window size of 64 and stride 16. This way we present to the model many more ways that pixels can be related to each other and the “value” of our dataset essentially increases linearly by the size of the dataset, which is $(64/16)^2 = 16$ times bigger compared to the dataset where no overlap is allowed.

We achieve inference by using soft-voting. More specifically, we slide a window of size 64 with stride 8 over the image and accumulate the softmax probabilities. Then, for each pixel we predict by selecting the category with the highest mean probability.

The semantic-segmentation approach is much faster than the pixel and patch based approaches since we predict many pixels in parallel. Also, with semantic segmentation our models can recognize relations between longer distances, since the image crops/patches used tend to be significantly larger than the ones used in patch-based approaches.

Below we can see a reconstruction of Dioni using a window of size 64 with stride 52 and padding enabled, while skipping every 5th window position. The window traverses the image rowise.

We can also see the results of the random flip_and_rotate transform when applied multiple times on the same window sample.

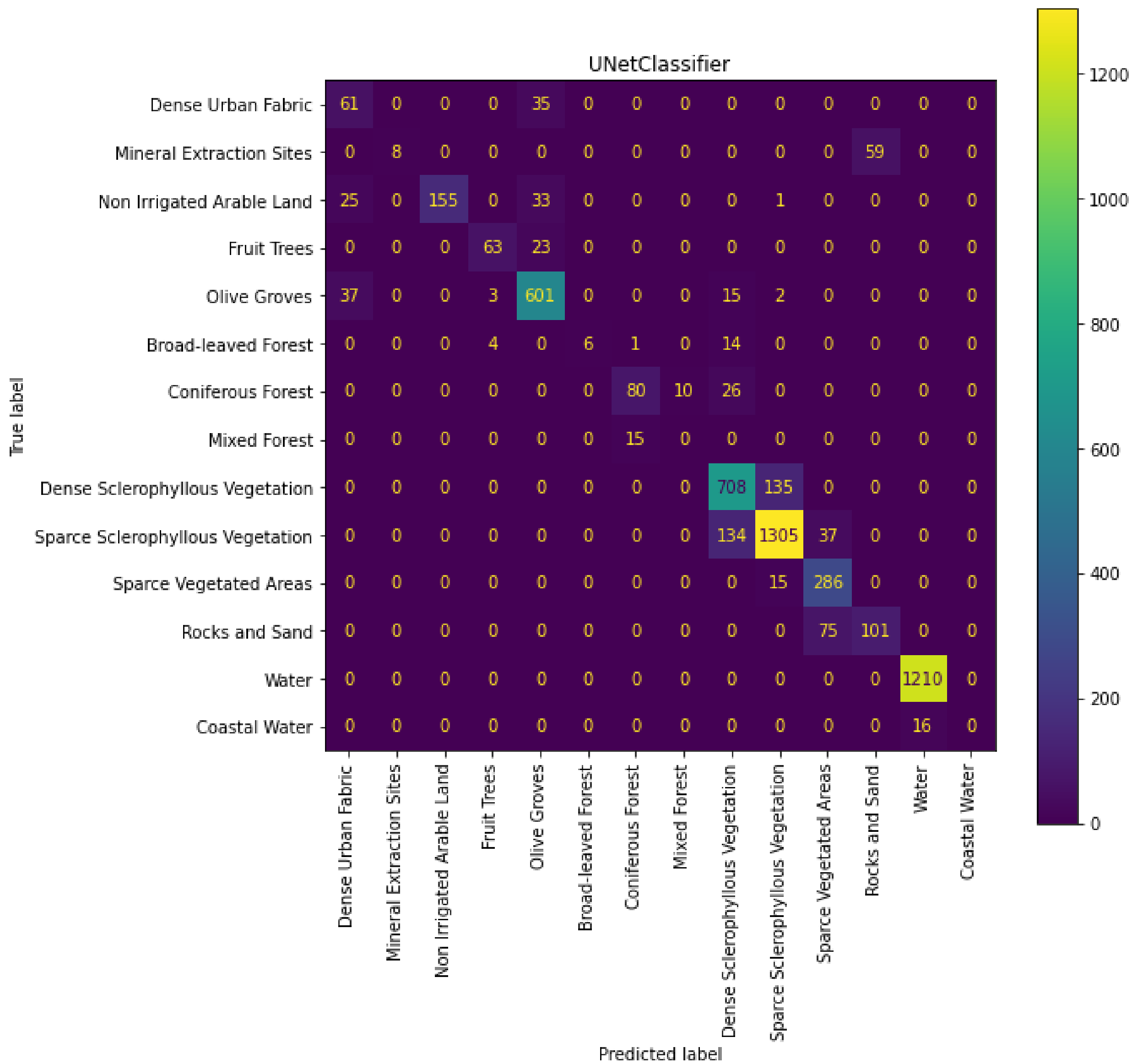


U-Net

We train a U-Net of depth 4 on a dataset created as discussed in the previous section. The results are seemingly “bad” compared to the other approaches, but this is mainly because our train and test sets are now relatively independent. The network was trained for 500 epochs, with the best validation loss achieved on epoch 222.

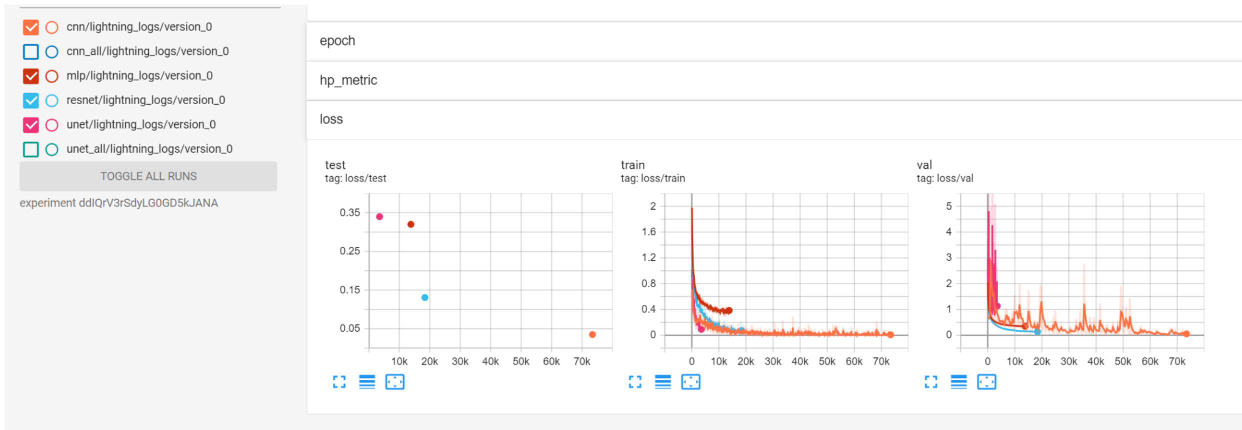
	precision	recall	f1-score	support
Dense Urban Fabric	0.50	0.64	0.56	96
Mineral Extraction Sites	1.00	0.12	0.21	67
Non Irrigated Arable Land	1.00	0.72	0.84	214
Fruit Trees	0.90	0.73	0.81	86
Olive Groves	0.87	0.91	0.89	658
Broad-leaved Forest	1.00	0.24	0.39	25
Coniferous Forest	0.83	0.69	0.75	116
Mixed Forest	0.00	0.00	0.00	15
Dense Sclerophyllous Vegetation	0.79	0.84	0.81	843
Sparce Sclerophyllous Vegetation	0.90	0.88	0.89	1,476
Sparce Vegetated Areas	0.72	0.95	0.82	301
Rocks and Sand	0.63	0.57	0.60	176
Water	0.99	1.00	0.99	1,210

Coastal Water	0.00	0.00	0.00	16
accuracy			0.87	5,299
macro avg	0.72	0.59	0.61	5,299
weighted avg	0.87	0.87	0.86	5,299



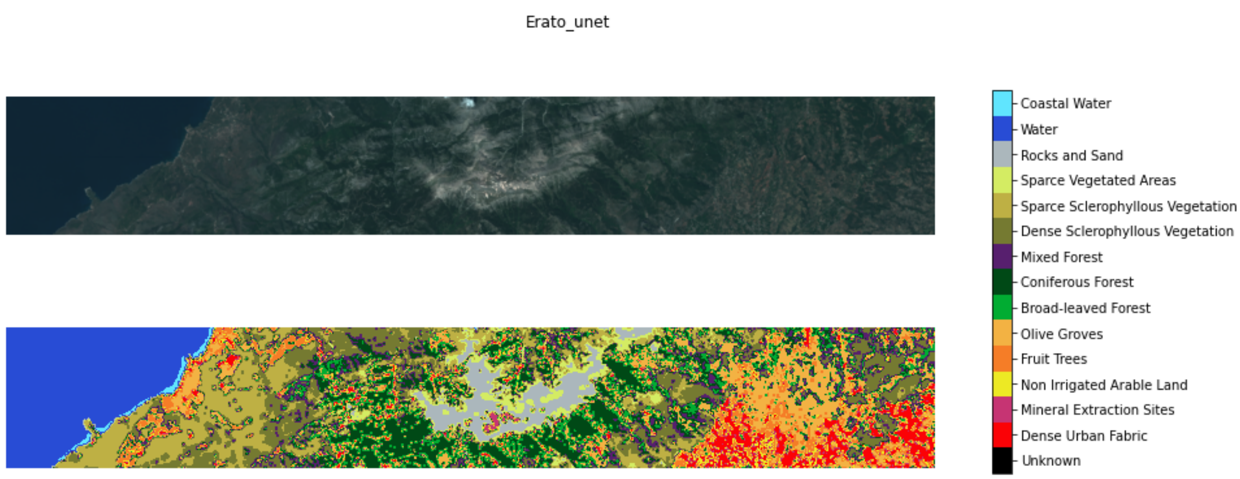
Learning curves

The training logs and interactive plots of learning curves can be found here:
https://tensorboard.dev/experiment/ddIQrV3rSdyLG0GD5kJANA
Note that the accuracy curves for unet do not seem to be correct, probably due to incorrect ignore_index and mdmc_average parameters in torchmetrics.Accuracy or an implementation bug.

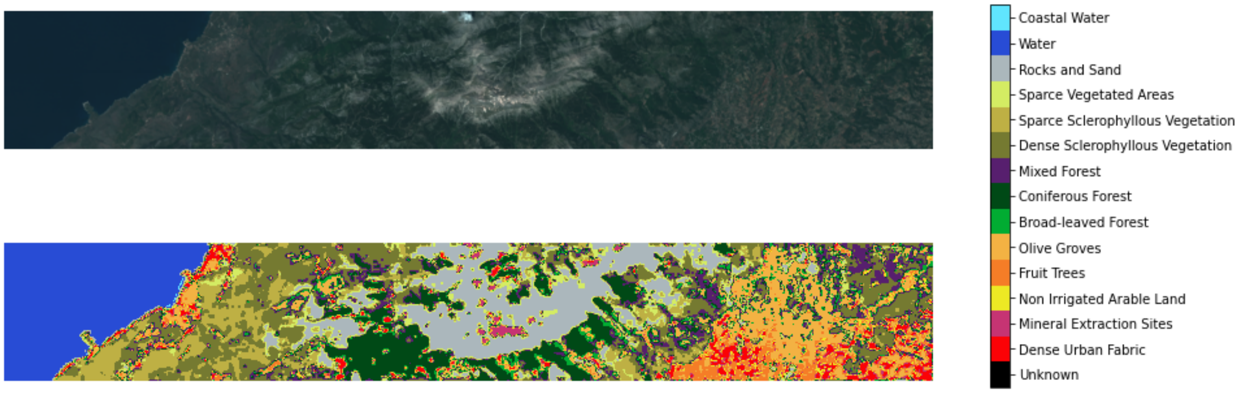


Inference

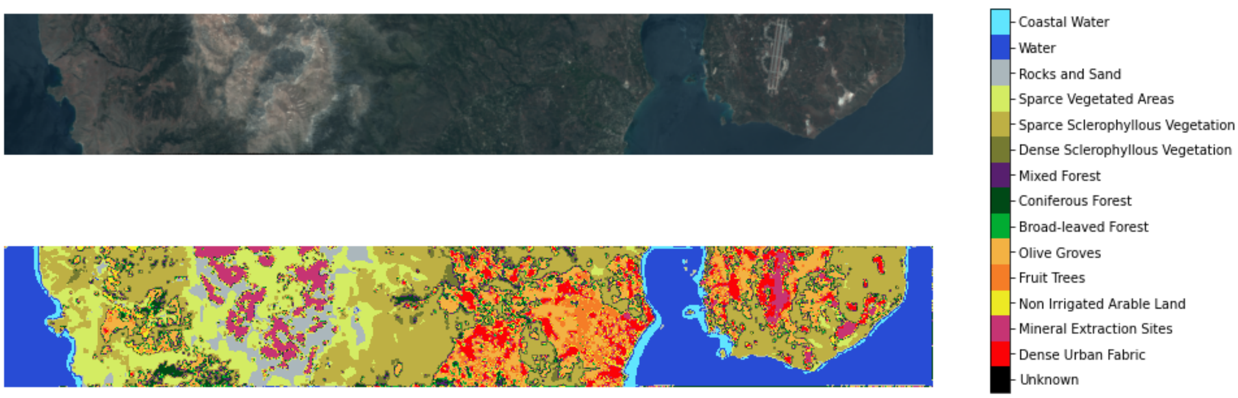
We select U-Net and CNN as candidates for best models and train them on the entire labeled images, and allow window overlap in the U-Net as previously discussed. The U-Net was trained for 400 epochs, while the CNN was trained for 200 epochs. We ran inference on the unlabelled dataset, using the soft voting method for the U-Net that we described earlier. The results are shown in the plots below. It seems that U-Net has made better predictions, with the main reason possibly being the fact that it can utilize relations between more distant pixels, as we mentioned previously.



Erato_cnn

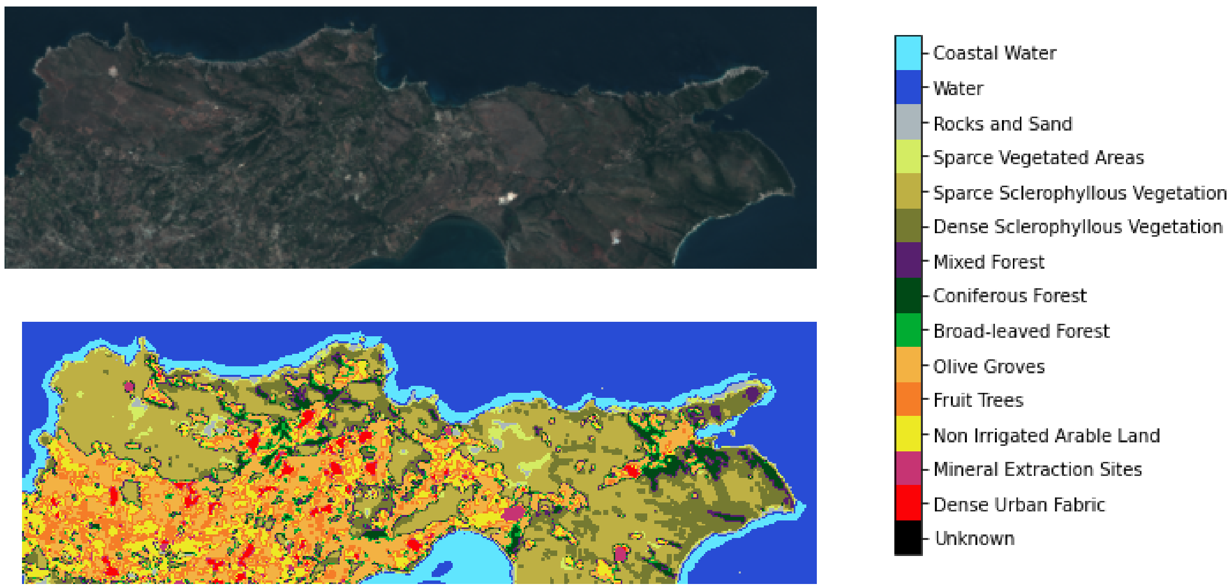


Kirki_unet



Kirki_cnn

Nefeli_unet



Nefeli_cnn