M.Sc. Data Science and Machine Learning

National Technical University of Athens (NTUA)

Assignment 7:

Hyperspectral Imaging Classification

Author:
Christos Charisis

Student:
Postgraduate

Student ID:
03400121

Email:
christ.charisis@gmail.com

Module Supervisor: Konstantinos Karantzalos

This exercise was submitted for the following module:

Geospatial Big Data Analysis

November 11, 2021

Purpose of the Work

The aim of this work is to create and train a classifier that can make predictions on unknown data. The problem we are addressing is the classification of hyperspectral data. In summary, the objectives are as follows:

- Design a deep architecture for a real geospatial data classification problem
- Study related literature and experimentation
- Implement and evaluate the actual performance of the model

Data

Specifically, the dataset used in this exercise is the HyRANK which consists of 2 hyperspectral images for training and 3 for testing. Using channels 23,11,7 the visualization of the training images is shown in the following images.

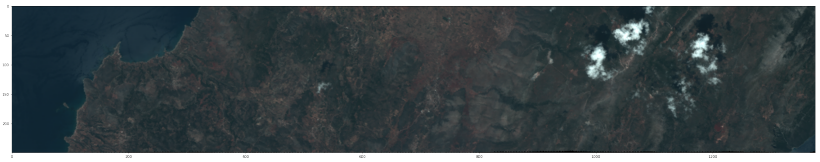

Figure 1: Training image - Dioni.

Figure 2: Training image - Loukia.

For the training of the neural network, the images were utilized using the patch generation method. For each labeled pixel in the ground truth data, appropriate cropping was performed on the hyperspectral image to obtain an area around that pixel. The 2-neighborhood area was chosen, meaning that for each pixel we had an area of 5x5 pixels where the central pixel was the one being examined, as shown in the following image. Due to hardware limitations, it was not possible to choose a neighborhood with a larger number of pixels (e.g., 7x7). The choice of the neighborhood size is an

open issue that also depends on the data, that is, how far from the examined pixel it makes sense to look for land cover.

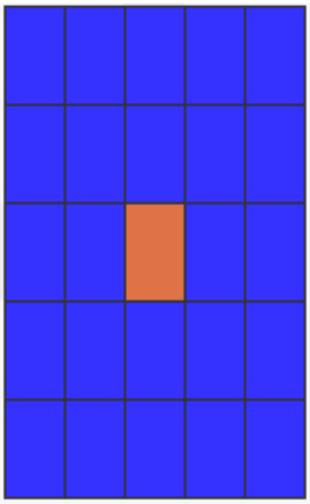

Figure 3: Pixel neighborhood-patch.

The orange color corresponds to the pixel under examination, for which we had its label, while the blue color corresponds to its neighborhood. It should be noted that for pixels on the edge of a labeled area, some pixels in their neighborhood did not have labels, which did not affect us as the only label necessary was that of the central pixel.

The above image clearly shows the 2D information we extract from the data. However, the patch obtained for each labeled pixel was not only a 2D image. In essence, the patch was 3D as in the third dimension we had all the spectral channels. Therefore, each patch corresponding to a labeled pixel was a cube with dimensions 176x5x5, where 5x5 was the neighborhood explained earlier and 176 were the spectral channels available from the data.

During data processing before model training, the mean value and the standard deviation of all pixels per spectral channel were extracted. Then, normalization of the data per spectral channel was performed so that for their distribution we had a mean value of 0 and a standard deviation of 1. The data, maintaining their overall proportions, were split into train-validation-test sets with proportions of 70-15-15% of the total samples.

It should be noted that the total dataset was obtained by combining the patches extracted from both images.

Table 1: Data patches of dataset

	Dioni	**Loukia**	**Total Dataset**
Patches	20024	13503	33527

ML Classifier Training

To have a comparison measure for the percentage that our model will produce, a Random Forest classifier with default hyperparameter values was

used. The Random Forest used spectral information for each pixel and not spatial information. That is, for each pixel, we had 176 features, as many as the spectral channels.

Also, the possibility of applying a dimensionality reduction technique to the data before being input into the classifier was examined. Specifically, the PCA technique (with dimensionality reduction to 20 from 176) was applied, which is quite common in the literature. Training and testing it on the labeled data from both images, we get the following results.

Table 2: Random Forest accuracy

	Random Forest	**Random Forest w/ PCA (20)**
Accuracy (%)	90.6	91.9

Seeing the Random Forest results, we consider that our model should have at least this level of success since it will incorporate both spectral and spatial information.

Architecture Design

After studying the recommended literature, it was chosen to create a model based on 3D convolutions. These networks offer state-of-the-art results as they combine the spectral information and the spatial information that hyperspectral images can provide. Following the literature, the model was chosen to be relatively small in size with a

Architecture Design

After studying the proposed literature, it was decided to create a model based on 3D convolutions. These networks offer state-of-the-art results as they combine spectral information and spatial information that hyperspectral images can provide. Following the literature, the model was chosen to be relatively small in size with simple logic in its layers. This happened because the hyperspectral data we have at our disposal are not many enough to support the training of large networks with many parameters, resulting in overfitting.

In the next image, the architecture of the network and how the dimension of each patch we input changes is shown. The input dimensions are 16x1x176x5x5, where 16 is the batch size, 1 is an auxiliary dimension to be compatible with the layers of pytorch (refers to the number of image channels - different from depth), 176 is the depth of the image which are our spectral channels and 5x5 is the neighborhood of the pixel under examination.

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
my_3D_net                                --                        --
├─Conv3d: 1-1                            [16, 16, 145, 5, 5]       4,624
├─MaxPool3d: 1-2                         [16, 16, 145, 2, 2]       --
├─Conv3d: 1-3                            [16, 32, 114, 2, 2]       147,488
├─MaxPool3d: 1-4                         [16, 32, 114, 1, 1]       --
├─Dropout: 1-5                           [16, 3648]                --
├─Linear: 1-6                            [16, 14]                  51,086
==========================================================================================
Total params: 203,198
Trainable params: 203,198
Non-trainable params: 0
Total mult-adds (G): 1.34
==========================================================================================
Input size (MB): 0.28
Forward/backward pass size (MB): 9.29
Params size (MB): 0.81
Estimated Total Size (MB): 10.39
==========================================================================================
```

Figure 4: Architecture of 3D convolution model.

It should be noted that after the conv layers and before the maxpool there are ReLU activation functions. The conv layers apply kernels with dimensions (32,3,3), with padding (0,1,1), while in the linear layer at the end of the network a dropout layer is applied with a neuron deactivation probability of 0.2. The total number of parameters is approximately 200k which is relatively small compared to other neural network implementations.

Classifier Training

Since we have prepared our data and built the network architecture, the training process begins. The Adam optimizer has been selected with a learning rate=0.001 and weight_decay=0.001 (L2 normalization). The model was trained for 100 epochs with the constant fear of overfitting. However, its small size, dropout, and L2 normalization did their job and such phenomena did not appear. In the next images, the diagrams for loss and accuracy for the training and validation sets are shown.

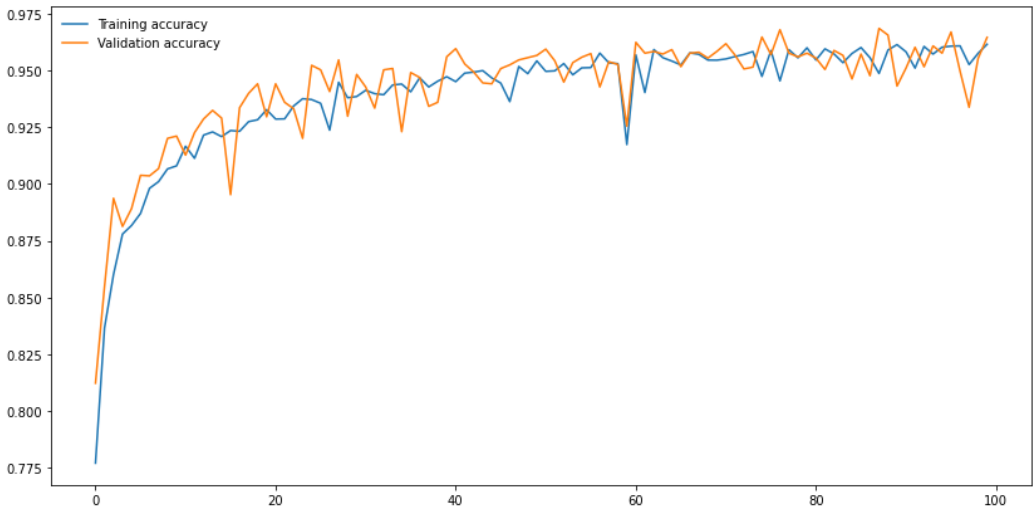

Figure 5: Accuracy training/validation.

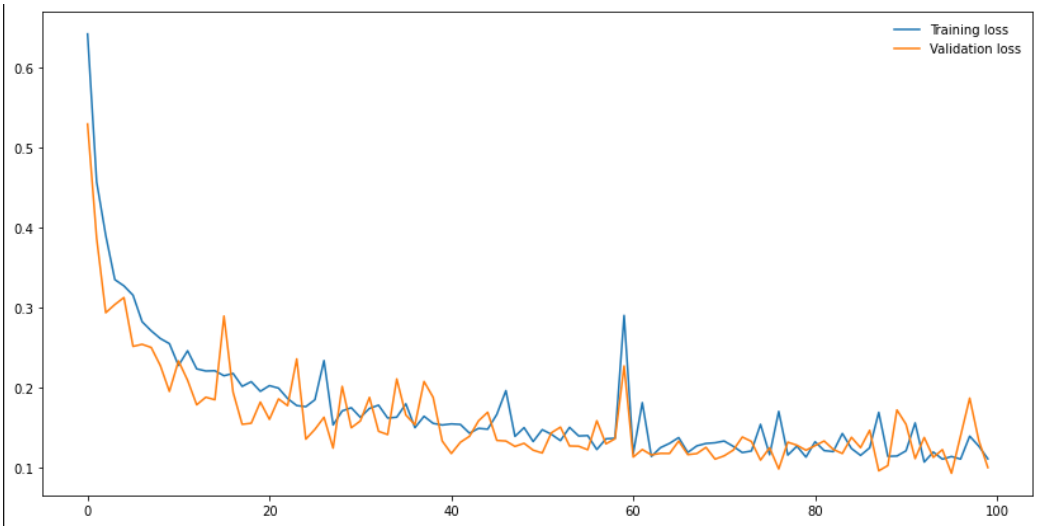

Figure 6: Loss training/validation.

We can observe in the images that both curves behave the same in both cases. If we had a typical overfitting scenario, we would expect the difference between training and validation to be significant. The stabilization of the curves at a value and their small oscillation around it is evident. Applying the model to the test set data we get the following results.

```
Test Loss: 0.006..  Test Accuracy: 0.969
Precision : [0.96412556 1.         0.9760479  0.92307692 0.94845361 0.92592593
 0.92913386 0.91025641 0.96008869 0.98029197 0.98165138 0.97333333
 0.99339207 1.        ]
Recall : [0.92274678 1.         0.94219653 0.70588235 0.96638655 0.75757576
 0.91472868 0.88198758 0.98037736 0.97530864 0.99074074 0.99319728
 1.         0.98425197]
F1 Score : [0.94298246 1.         0.95882353 0.8        0.95733611 0.83333333
 0.921875   0.89589905 0.97012696 0.97779396 0.98617512 0.98316498
 0.99668508 0.99206349]
Precision weighted: 0.9684782527534255
Recall weighted: 0.9685884691848906
F1 Score weighted: 0.9682959284572583
```

Figure 7: Statistic metrics on test set (per class and weighted).

We see that the model performs very well on the test set, even better than the Random Forest we used at the beginning of the analysis. On the one hand, this is expected as we exploit both spectral and spatial information.

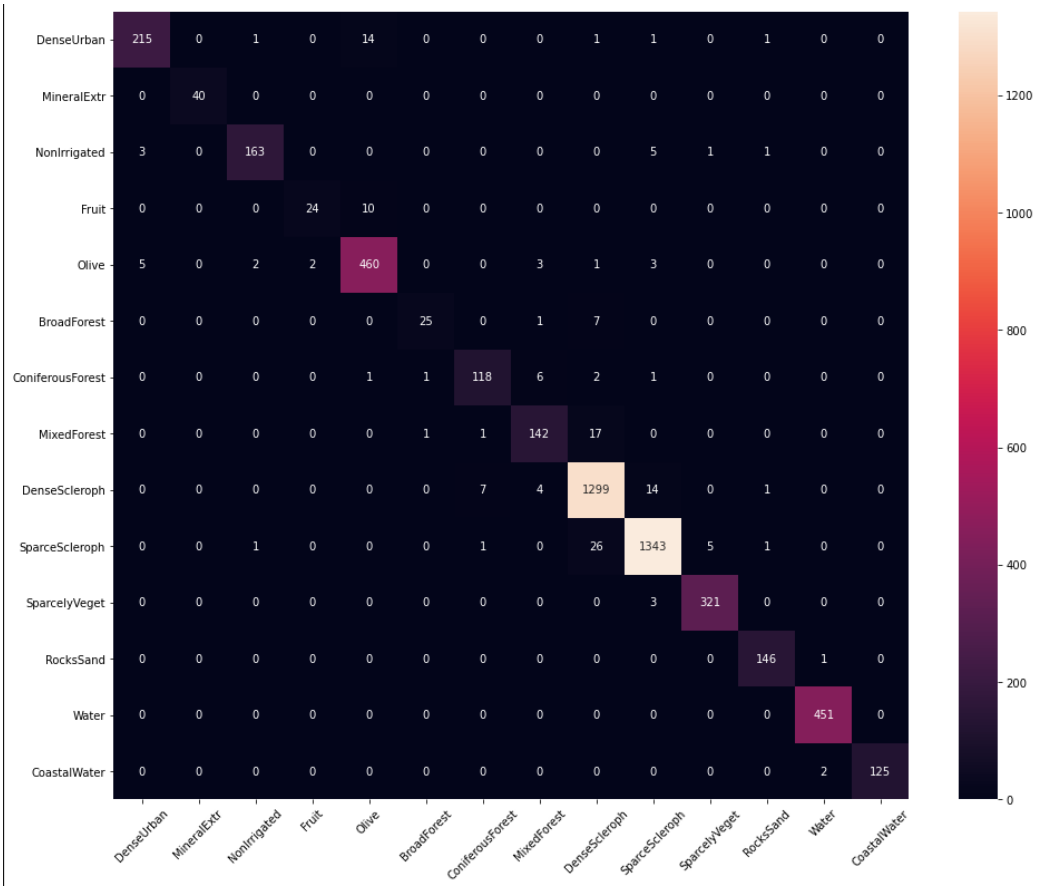

Figure 8: Confusion matrix on test set.

On the other hand, issues arise concerning information leakage. That is, as the data have been randomly divided into train/val/test sets, it is common for neighboring pixels to belong to different sets. This results in the model being able to see part of the val/test data during its training, leading to very high scores in the evaluation metrics. Surely, in completely unseen data of the 3 images given, the success rate will be significantly lower.

Generated Test Set Images

Next, the results given by the model for the 3 images of the test set are visualized. These images are also provided as .tif files corresponding to the files used for model training.

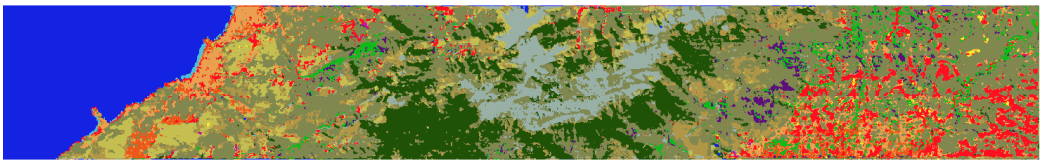

Figure 9: Erato results.

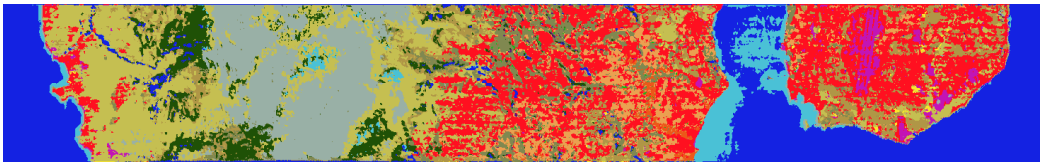

Figure 10: Kirki results.

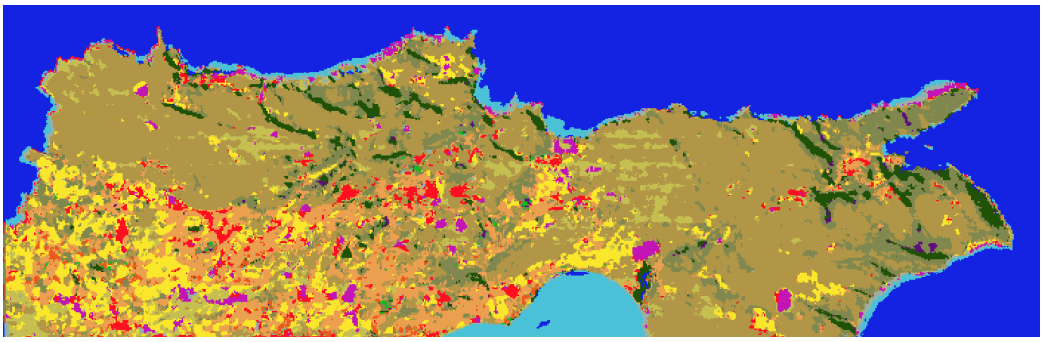

Figure 11: Nefeli results.

The color palette used is approximately the same as the one followed by HyRANK on its page.

Note

The deliverable also includes the .ipynb file used to generate the results of the report. The code does not have detailed comments. If further clarifications are required for the whole process, please contact me.