Pytorch Project
Land cover classification using Sentinel-2 satellite images

Deep Learning UniPD

Alessandro Viespoli 2120824
Francesco Colla 2122543
Kabir Bertan 2122545

June 2024

Abstract

In this project we have analyzed different deep learning approaches on the same dataset (27000 images captured by Sentinel-2 satellites with 13 channels), using information extracted from multiple bands to maximize classification performance. Starting from the base RGB analysis, we have done further testing on a ensemble of multiple classificator, trained on different bands and PCA dimensionality reduction to collapse 13 bands into 3 to be then fed into the network. The tests have been done on ResNet50 network; in addition deeper networks like DenseNet121 and 161 have been taken into consideration.

Dataset informations

The dataset used in this project, available via Zenodo, is made of 10 different classes composed of 2000 to 3000 images per class with a total of 27000 images took by the satellites of the mission Sentinel-2. The images have resolution of 64x64 pixels and every pixel has a spatial resolution of 10 meters. The taken dataset was also manually checked and corrected by the authors, removing images with no information such as only snow or ice. For our target analysis we have considered the dataset with the multi band images, that were available in the ".tiff" format. These images have 13 channels:

0. Aereosols
1. Blue
2. Green
3. Red
4. Red edge 1
5. Red edge 2
6. Red edge 3
7. Near infrared(NIR)
8. Red edge 4
9. Water vapor
10. Cirrus

11. Short wave infrared (SWIR)1
12. Short wave infrared (SWIR)2

With all this data available it was possible to select specific channels to feed the network, getting more information than the usual RGB values to help the prediction accuracy. The runs have been executed on three different machines all running Windows and with RTX series GPUs using CUDA 12.4.1 and PyThorch 2.2.2+cu121

Data preprocessing

ResNet50 can only accept as input images composed of three channels and with a specific resolution(244x244) so obviously the full images with all the 13 channels cannot be fed directly to the network, in this case preprocessing was mandatory. To begin with, the images are loaded from the storage and some transformations had to be applied to make them compatible as ResNet's input. The transforms involved are a transformation to a Pillow image, a resize to 256x256 and finally a center crop to reduce the image at 224x224. At this point it is sufficient to transform the result into a tensor where standardization is automatically applied. The dataset was split into two main partitions: 80% train 20% test, taking the validation from the training set. A second split was also tested with 50/50 train and test to verify how the system would behave with less training data.

Training

Fine tuning strategy has been applied using ResNet50 as pretrained base model(more complex model like DenseNet121 and DenseNet161 are been considered, with results in Observation paragraph), the last layer has been modified to adapt the network to our specific use-case by changing the number of reported classes to 10. A softmax layer is then applied. Training was done multiple times, After multiple training iterations the best possible combination of hyperparameters with optimizers Adam and SGD with momentum were chosen. The loss function that has been used is Cross Entropy Loss.

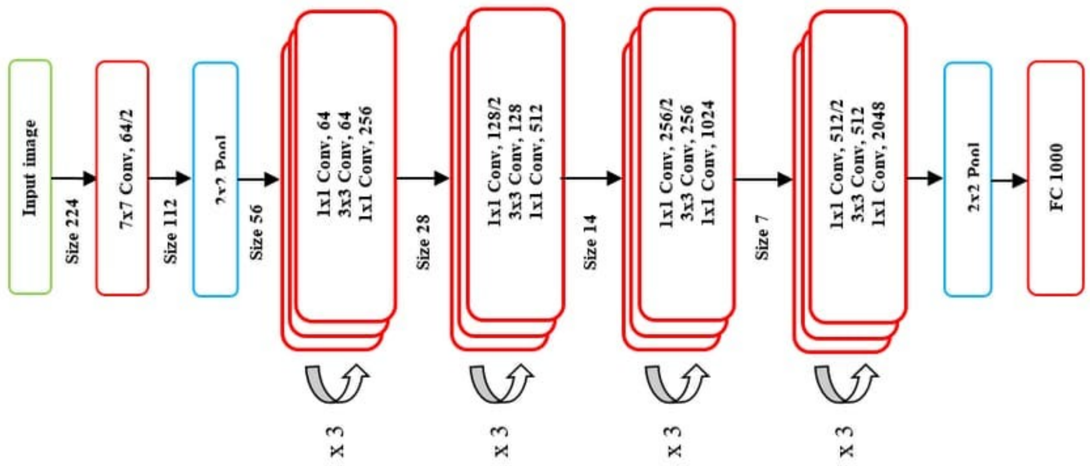


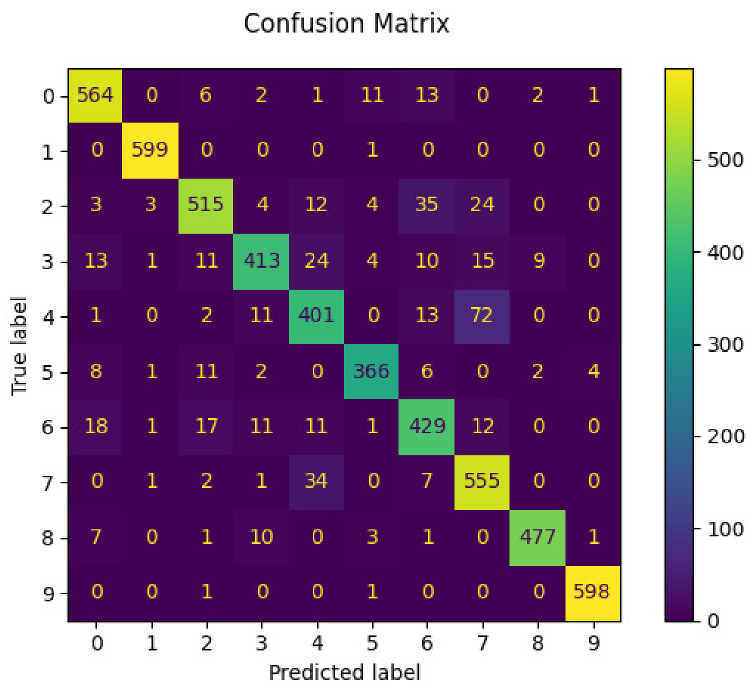
Figure 1: Resnet50 architecture

RGB

To have a baseline we executed the analysis on the RGB bands. The RGB analysis is executed without further actions, feeding the preprocessed data to the network(with normalization, mean=[0.485,0.456,0.406] and std=[0.229,0.224,0.225], values used for RGB with ResNet50). Executing tests with both partitioning methods we obtain acceptable results but with room for improvement.

- 80% train: Accuracy 91,05%, Precision 91,18%, Recall 91,05%, F1 91,04% Using Learning rate 1.00E-4, Batch size 32, 30 epochs, Optimizer Adam, Scheduler STEPLR with stepsize 15 and gamma 0.1
- 50% train: Accuracy 88.26% Precision 88.35%, Recall 88.26%, F1 88.24% Using Learning rate 1.00E-4, Batch size 32, 100 epochs, Optimizer Adam, Scheduler STEPLR with stepsize 20 and gamma 0.1

Results using only RGB bands



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	564	0	6	2	1	11	13	0	2	1
1	0	599	0	0	0	1	0	0	0	0
2	3	3	515	4	12	4	35	24	0	0
3	13	1	11	413	24	4	10	15	9	0
4	1	0	2	11	401	0	13	72	0	0
5	8	1	11	2	0	366	6	0	2	4
6	18	1	17	11	11	1	429	12	0	0
7	0	1	2	1	34	0	7	555	0	0
8	7	0	1	10	0	3	1	0	477	1
9	0	0	1	0	0	1	0	0	0	598

Figure 2: Confusion Matrix of RGB result

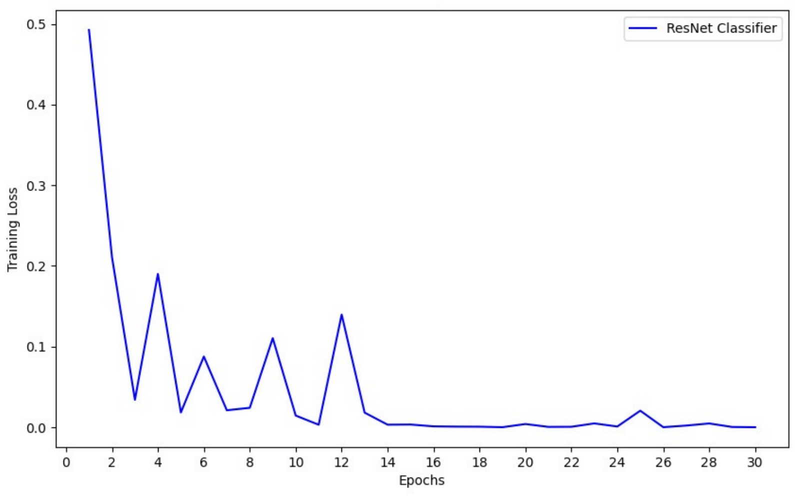


Figure 3: Plot of RGB result

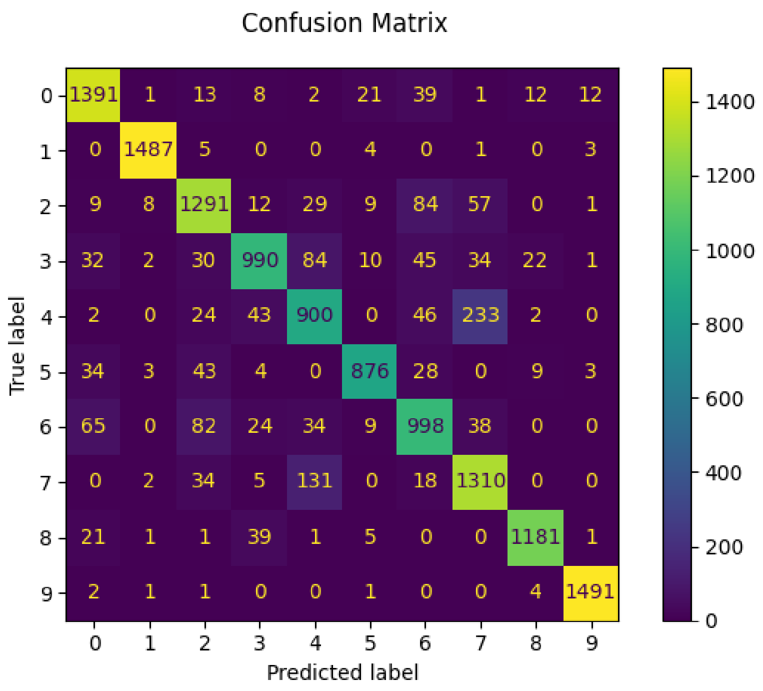


Figure 4: Confusion Matrix of RGB result with training 0,5 and test 0,5

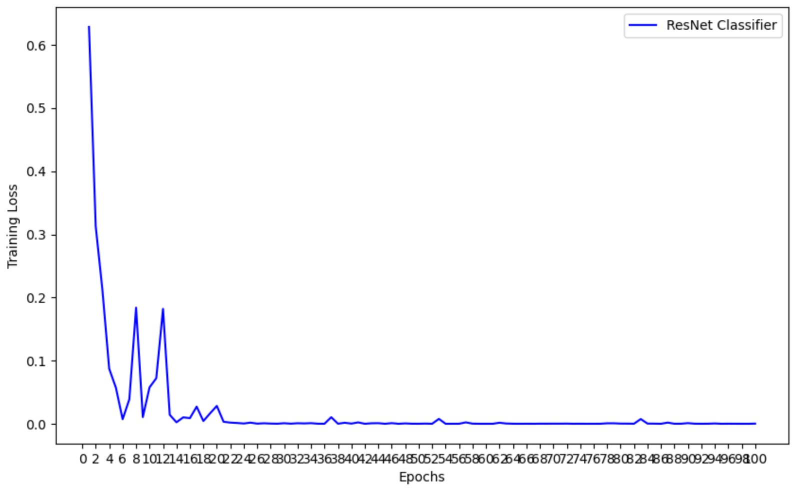


Figure 5: Plot of RGB result with training 0,5 and test 0,5

Ensemble

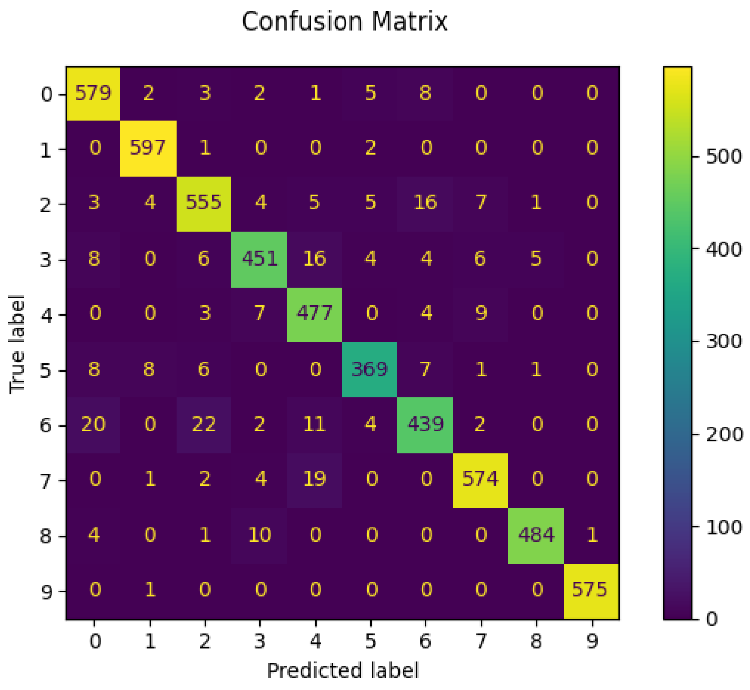
To pursue the task of extracting more information from the dataset we executed ensemble of networks using the weighted sum rule. The set of bands chosen for the ensemble networks are:

- Network 1 RGB: bands [3,2,1]
- Network 2 Atmospheric Factors: bands [0,9,10]
- Network 3 Red edge: bands [4,5,6]
- Network 4 SWIR: bands [7,11,12]

In order to get the weights for the final gathering of the single nets predictions, we further split the train dataset to extract another 20% of data for validation. We used the validation accuracy of the classifier as its weight. The final predicted label chosen will be the class with the highest summed score (the predictions of the subnetworks are given in one hot encoding). While running the tests for the ensemble procedure we spotted that the RGB bands were being corrected by the other bands, increasing the overall performance of the classifier.

- 80/20 rule: Accuracy 94.86%, Precision 94.88%, Recall 94.86%, F1 94.85% Using Learning rate 1.00E-4, Batch size 32, 20 epochs, Optimizer Adam, Scheduler STEPLR with stepsize 5 and gamma 0.1
- 50/50 rule: Accuracy 91.21% Precision 91.22%, Recall 91.21%, F1 91.14% Using Learning rate 1.00E-4, Batch size 32, 15 epochs, Optimizer Adam, Scheduler STEPLR with stepsize 5 and gamma 0.1

Results using Ensemble Method



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	579	2	3	2	1	5	8	0	0	0
1	0	597	1	0	0	2	0	0	0	0
2	3	4	555	4	5	5	16	7	1	0
3	8	0	6	451	16	4	4	6	5	0
4	0	0	3	7	477	0	4	9	0	0
5	8	8	6	0	0	369	7	1	1	0
6	20	0	22	2	11	4	439	2	0	0
7	0	1	2	4	19	0	0	574	0	0
8	4	0	1	10	0	0	0	0	484	1
9	0	1	0	0	0	0	0	0	0	575

Figure 6: Confusion Matrix of Ensemble results

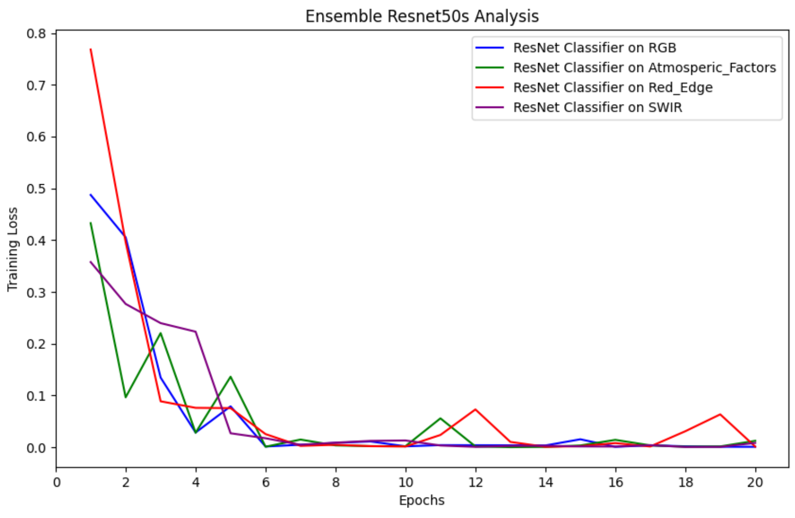
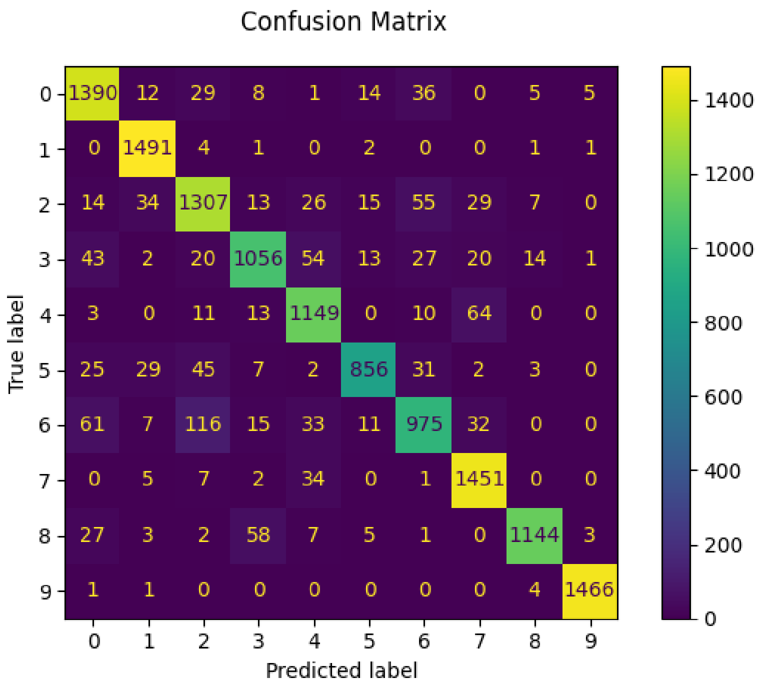


Figure 7: Plot of Ensemble results



	0	1	2	3	4	5	6	7	8	9
0	1390	12	29	8	1	14	36	0	5	5
1	0	1491	4	1	0	2	0	0	1	1
2	14	34	1307	13	26	15	55	29	7	0
3	43	2	20	1056	54	13	27	20	14	1
4	3	0	11	13	1149	0	10	64	0	0
5	25	29	45	7	2	856	31	2	3	0
6	61	7	116	15	33	11	975	32	0	0
7	0	5	7	2	34	0	1	1451	0	0
8	27	3	2	58	7	5	1	0	1144	3
9	1	1	0	0	0	0	0	0	4	1466

Figure 8: Confusion Matrix of Ensemble results with training 0,5 and test 0,5

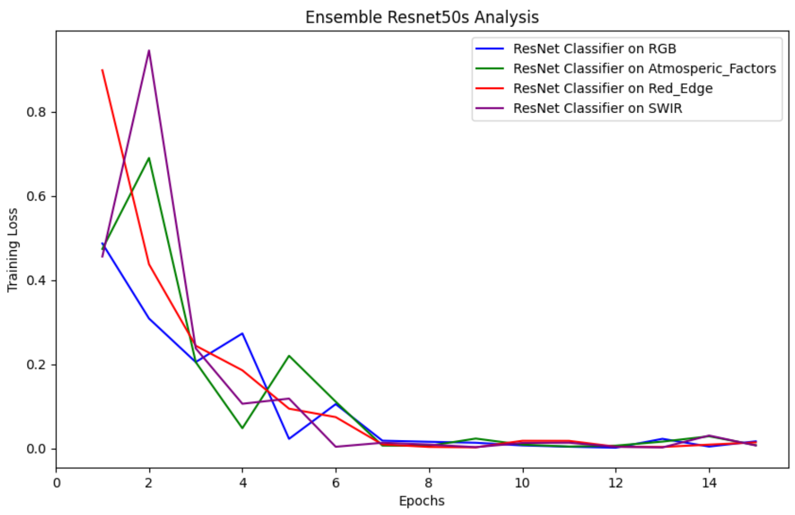


Figure 9: Plot of Ensemble results with training 0,5 and test 0,5

PCA analysis

The principal component analysis makes possible to collect the most important information of the available channels and discard what's not considered as useful. This mathematical procedure allows us to reduce the number of channels to 3 so that the result can be ran into the ResNet50 network. To execute the PCA the dataset instances have to be normalized and flattened first. The PCA function is then applied to each image with 3 as number of components to keep. The function will maintain the best features while losing the least important ones. The results were not ideal, giving in most cases performance worse than the RGB base network.

- 80/20 rule: Accuracy 88.80%, Precision 88.88%, Recall 88.80%, F1 88.77% Using Learning rate 1.00E-4, Batch size 32, 100 epochs, Optimizer Adam, Scheduler STEPLR with stepsize 20 and gamma 0.1

- 50/50 rule: Accuracy 85.55% Precision 85.55%, Recall 85.55%, F1 85.54% Using Learning rate 1.00E-4, Batch size 32, 100 epochs, Optimizer Adam, Scheduler STEPLR with stepsize 20 and gamma 0.1

Results using PCA

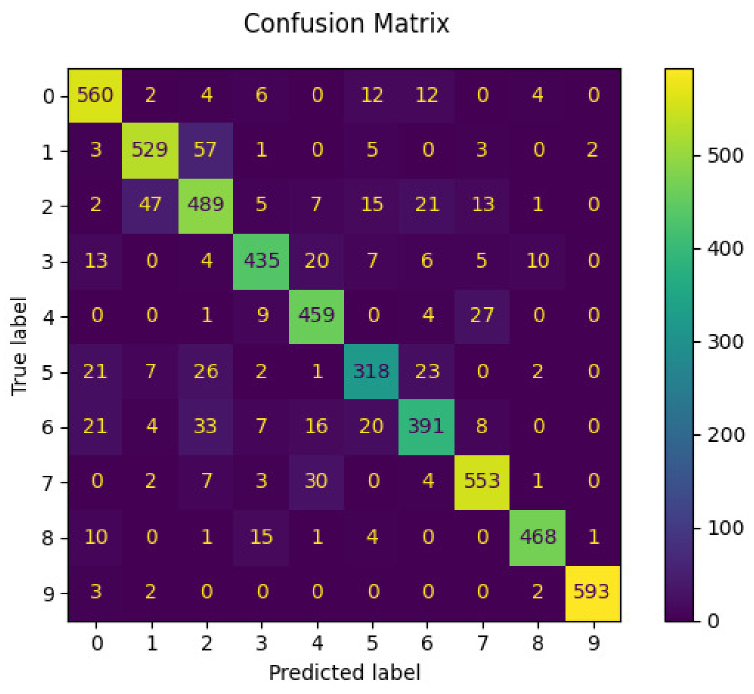


Figure 10: Confusion Matrix of PCA result

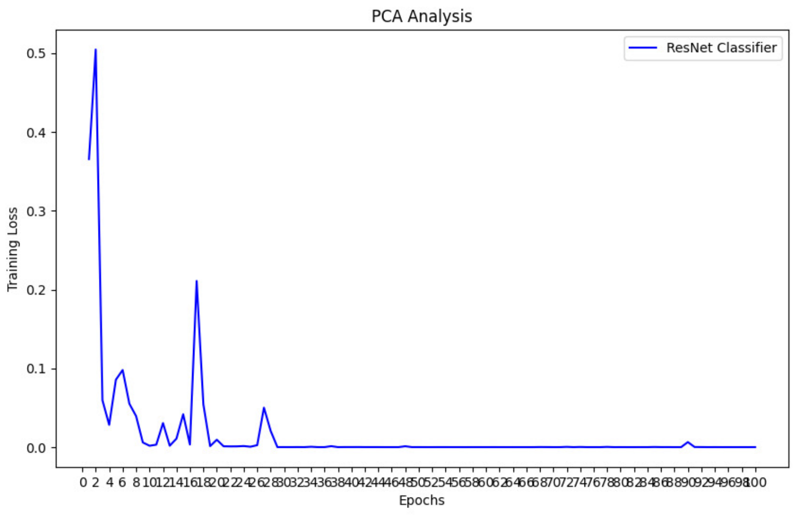
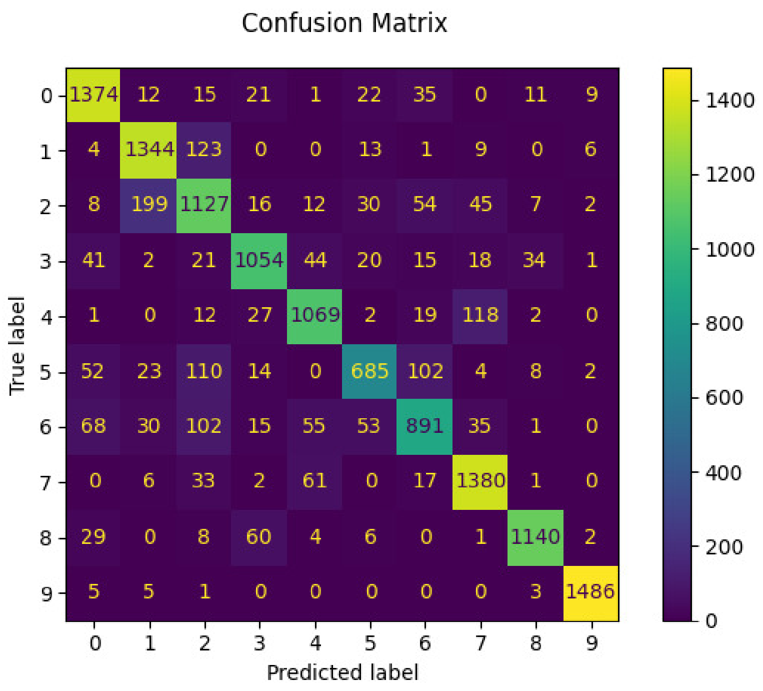


Figure 11: Plot of PCA result



	0	1	2	3	4	5	6	7	8	9
0	1374	12	15	21	1	22	35	0	11	9
1	4	1344	123	0	0	13	1	9	0	6
2	8	199	1127	16	12	30	54	45	7	2
3	41	2	21	1054	44	20	15	18	34	1
4	1	0	12	27	1069	2	19	118	2	0
5	52	23	110	14	0	685	102	4	8	2
6	68	30	102	15	55	53	891	35	1	0
7	0	6	33	2	61	0	17	1380	1	0
8	29	0	8	60	4	6	0	1	1140	2
9	5	5	1	0	0	0	0	0	3	1486

Figure 12: Confusion Matrix of PCA result with training 0,5 and test 0,5

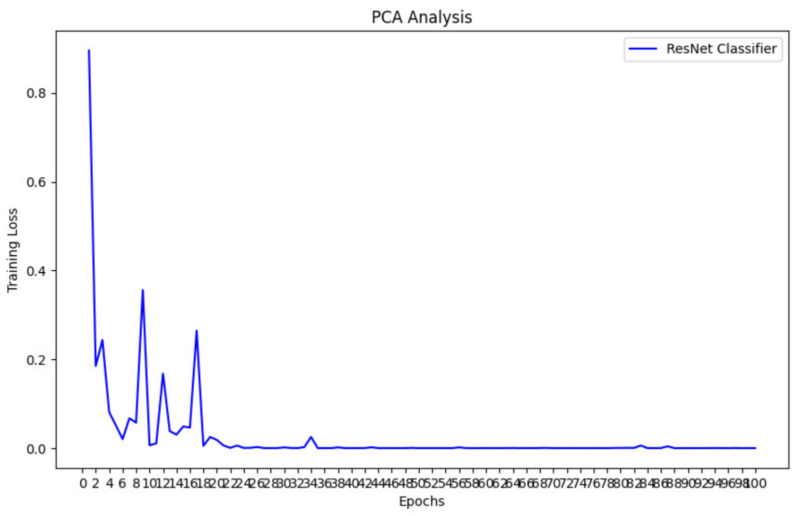
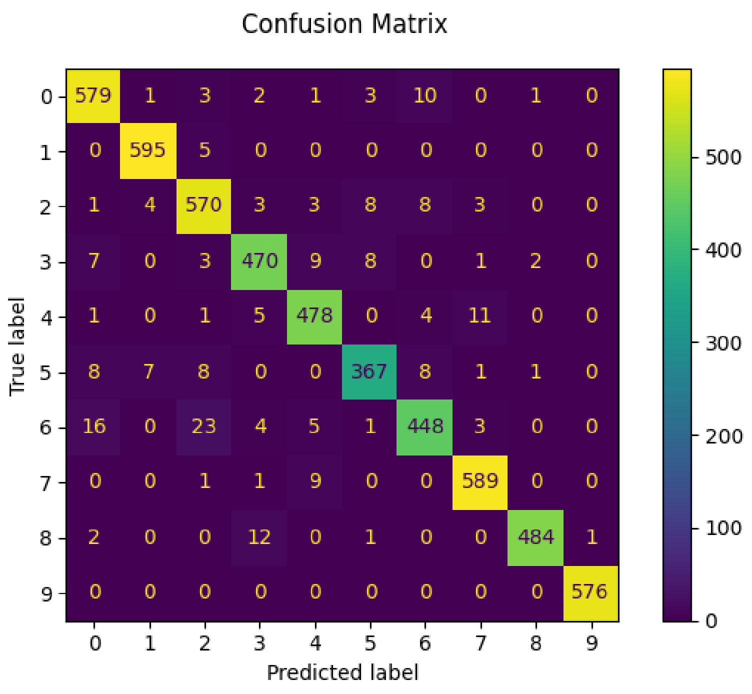


Figure 13: Plot of PCA result with training 0,5 and test 0,5

Results using more complex DNN

Using deeper networks such as DenseNet121 and 161 we have obtained better results without any change in the code (except import of densenet121/161 and model assegnation). We were able to get the accuracy up to 96,9%, however the training time spiked exponentially, arriving at about 16 hours on consumer level hardware (GPU Nvidia RTX 3070). This suggest to us that by using deeper networks instead of ResNet50 we can improve the performance, learning more complex information about the data.

Our best ResNet50 result is obtained with Ensemble strategy with an accuracy of 94.9%, however with DenseNet161 we obtain (with hyperparameters: Learning rate 1.00E-4, Batch size 32, 10 epochs, Optimizer Adam, Scheduler STEPLR with stepsize 5 and gamma 0.1) an accuracy of 96.9%.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	579	1	3	2	1	3	10	0	1	0
1	0	595	5	0	0	0	0	0	0	0
2	1	4	570	3	3	8	8	3	0	0
3	7	0	3	470	9	8	0	1	2	0
4	1	0	1	5	478	0	4	11	0	0
5	8	7	8	0	0	367	8	1	1	0
6	16	0	23	4	5	1	448	3	0	0
7	0	0	1	1	9	0	0	589	0	0
8	2	0	0	12	0	1	0	0	484	1
9	0	0	0	0	0	0	0	0	0	576

Figure 14: Confusion Matrix of Ensemble result with DenseNet161

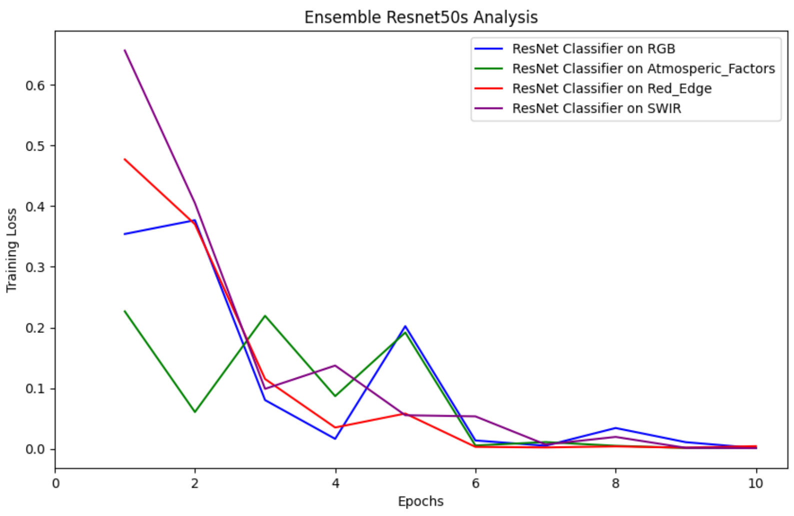


Figure 15: Plot of Ensemble result with DenseNet161

Conclusion

Doing fine tuning (using the pretrained ResNet50) on the RGB values we built a baseline that we wanted to optimize. After applying the PCA even on an large number of epochs there was not an improvement, even lowering the performance. On the other side the ensemble of networks, even if it makes the training process longer, allows us to have better results than RGB in both partitions. Using the information coming from more channels allowed the improvement of the classification procedure even if the computational cost drastically improves. A possible future improvement that might increase accuracy and stability of our analysis could be to generate more data through data augmentation.