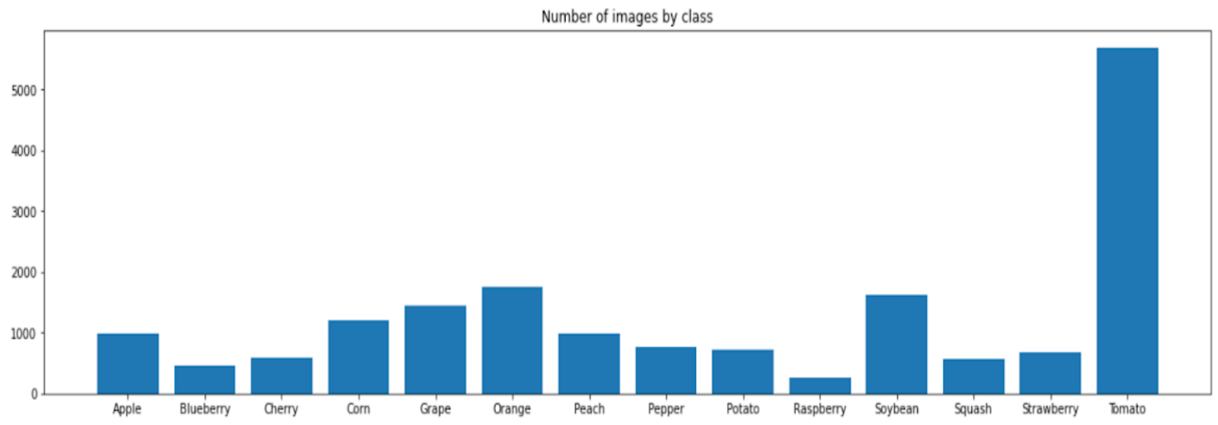
Report Challenge1 ANN&DL

i_Cesaptroni: samuelepartacini, AlessandroMessori, DavideMangano

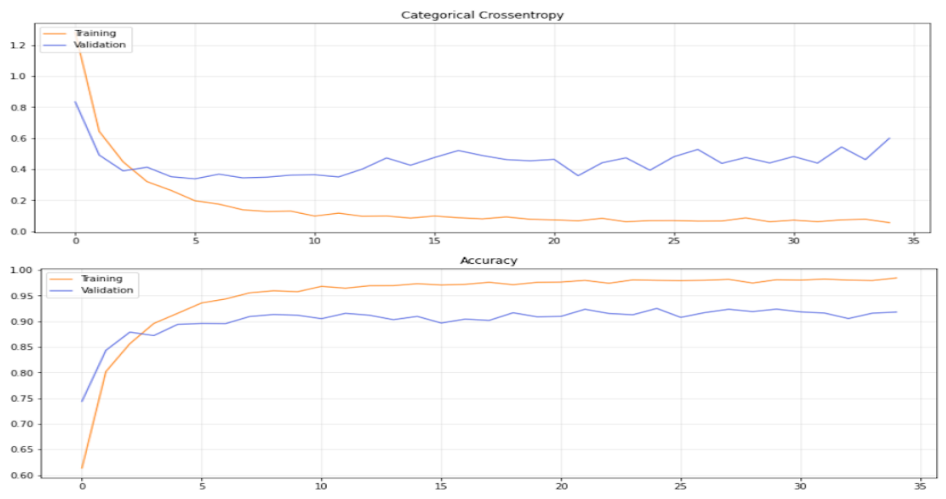
Dataset Exploratory Analysis

Before starting implementing our models, we did some exploratory analysis on the dataset. You can find all what we did in the Preprocessing notebook that we attached. First of all, we inspected the size of each training class. This showed a prominent imbalance among different classes. We ignored this fact in our first attempts, but eventually we handled the thing, as we will discuss afterwards. Then we checked if each image in the dataset was compliant with the declared size (256,256), finding two outliers. After that, we visually inspected some samples from the dataset for each class. We found out that the leaves were in very different states of 'health', which means that many of them were cut, dry or even shattered. Finally, we did some trials of data loading with ImageDataGenerator and of data augmentation, by applying rotation, shifting, flipping and zooming to the images and visually analyzing the results in order to 'calibrate' the augmentation.

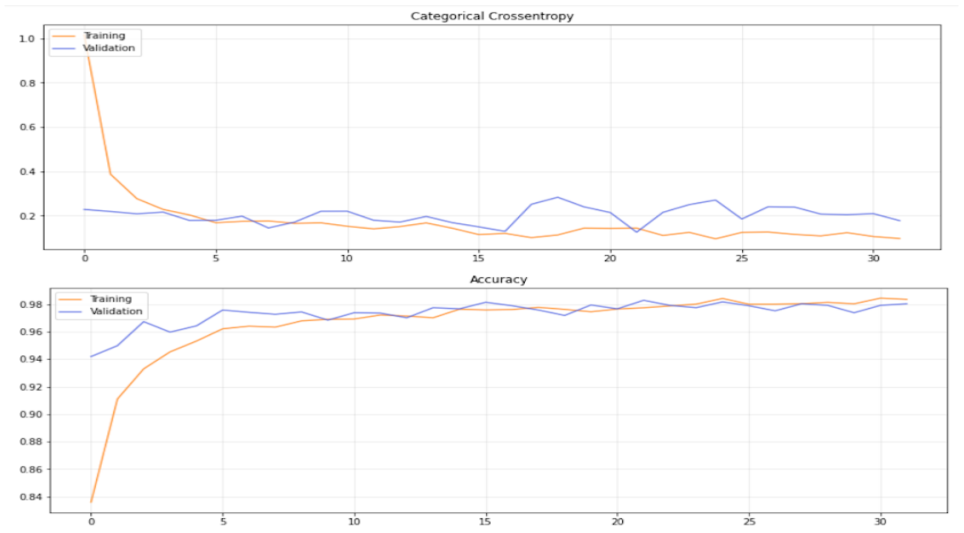


First 'standard' model

For our first trial we developed a very simple convolutional neural network, you can find it in the Standard_Model notebook we attached. It was composed of three convolutional layers, each of which followed by a MaxPooling layer. In the fully connected part we inserted a single dense layer before the output. We used dropout in order to contain possible overfitting. Furthermore, we trained the model using Early Stopping with a patience of ten epochs, this was meant to avoid overfitting but also to shorten training times. From this model we obtained a 0.92 accuracy on a 20% validation set, but the accuracy on the test set was very disappointing; just 0.23.



Transfer learning model
After the poor result of the previous model we decided that we needed a more powerful feature extractor. We then applied transfer learning by using VGG16 as feature extractor, with weights set on the Imagenet dataset. You can see what we did in the Transfer_Learning notebook we attached. After the convolutional part we placed a fully connected network similar to the one we used in the previous model (just one dense hidden layer, but with more units). With this model we obtained better results both on validation set (0.98 accuracy) and test set (0.47), but it still wasn't enough. From this moment we started realizing that the images in the hidden test set could be quite different from the ones at our disposal. Given the very good results on the validation set, we excluded overfitting as the problem.

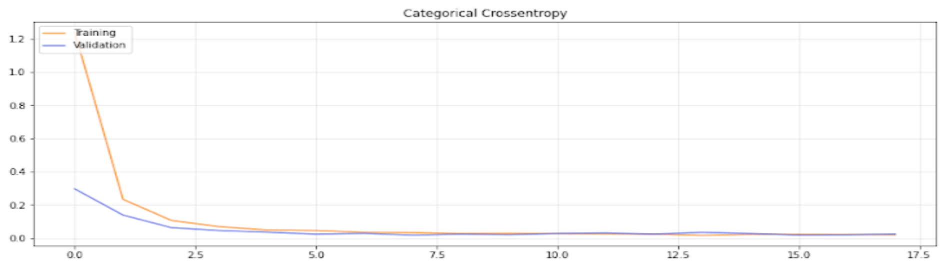


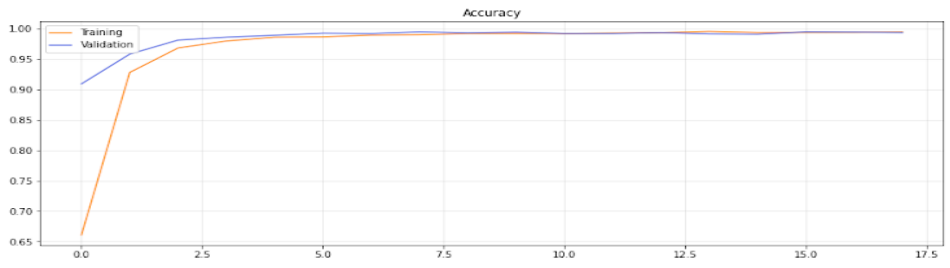
Transfer learning + data augmentation
The first step we took to deal with our poor performance on the test set was applying data augmentation. Our hope was to obtain a wider range of images that could better match the ones in the test set. At the beginning we used VGG16 again as feature extractor, then we moved to VGG19 since it showed a better performance. Nevertheless, our performance on the test set didn't improve so much (0.49). You can find our transfer learning experiment on transfer learning with VGG19 and data augmentation in the TL_aug_VGG19 notebook we attached.

Dataset balancing
The result on the test set showed that our performance differed greatly from class to class. Of course the classes of which we had more images in our dataset were characterized by a greater accuracy. We then developed a script to balance the dataset, by sampling at most X images per class. You can find it in the script balance_dataset.py we attached.
We were aware that with this method we reduce the amount of samples in the dataset, but we thought that the loss of data could be more than compensated by the gained balance in the number of images in each class. However, after creating a balanced dataset with 500 images for class and training our model on that test set, we weren't able to improve our test accuracy.

A bigger dataset
In order to improve the accuracy of our model, we started looking for ways to enlarge our dataset. We found an extended version of the original dataset on Github, and after a bit of analysis we managed to restructure the shape of the directories in the new dataset so that it matched the original structure.
The extended dataset can be found at this address:
https://github.com/spMohanty/PlantVillage-Dataset

Fine tuning
In order to improve the accuracy of our model, we decided to add fine tuning to our transfer learning based one. We decided to freeze 14 layers, as we noticed that this value offered the best tradeoff between training time and accuracy.
We would have wanted to evaluate our models with a cross validation based approach, but that was not possible given the very high training times it would require, so we decided to stick to the standard single validation method.
Initially we started from the model based on VGG16 and, training the model on the extended dataset, we obtained a test accuracy of 0.90, the code for this model can be found in the attached notebook Fine_Tuning.ipynb. From there we were able to further increase the accuracy of our model twice:
The first time by moving from a VGG16 to a VGG19 model and by replacing the relu activation function we had used so far with leaky relu.
The second time by adding another dense layer and tweaking a bit the data augmentation parameters; this improved our accuray from 0.91 to 0.924. The code for this model can be found in the attached notebook Best_Model.ipynb.





Experiments on balanced Dataset

Thanks to the extended dataset, we were able to generate new versions of the balanced dataset, with respectively 800 and 1500 images for each class.
With this method we managed to greatly reduce the unbalance among the classes but none of these models, when trained on the balanced datasets, were able to outperform the ones trained on the complete extended dataset, so we decided to abandon this approach.