AN2DL - First Challenge

Andrea Deretti
Francesco Rettore
Michele Russo

POLITECNICO
MILANO 1863

Politecnico di Milano
Italy
27 November 2021

1 Introduction

1.1 Goal of the Competition and Available Data

This competition consists in an image classification task. In particular, we are required to classify images of leaves according to the species of the plant to which they belong by building a Convolutional Neural Network (CNN). Moreover, the data at our disposal to train the CNN consist in 17728 images of size 256x256 belonging to one of the following 14 classes:

- Apple (988 images).
- Blueberry (467 images).
- Cherry (583 images).
- Corn (1206 images).
- Grape (1458 images).
- Orange (1748 images).
- Peach (977 images).
- Pepper (765 images).
- Potato (716 images).
- Raspberry (264 images).
- Soybean (1616 images).
- Squash (574 images).
- Strawberry (673 images).
- Tomato (5693 images).

Observe that the dataset is unbalanced, meaning that the number of samples per class is not uniform among the classes.

1.2 Data Pre-Processing

The majority of images in the dataset are similar to the one shown in Figure 1. In particular, the leaf is always depicted from the same angle (*i.e.* from a frontal perpsective) and with the same orientation (*i.e.* with the tip at the top of the picture). Moreover, we can also observe that the leaf is always at the center of the picture and that the background has been removed. In order to teach the CNN to classify images out of this "controlled" environment we resorted to the technique of data augmentation by introducing (at random) the following transformations to the images of the dataset:

Figure 1: Image from the class *Peach*

- *Rotations* (up to an angle of 90 degrees), *horizontal flips and vertical flips*. Such transformations are useful to classify images in which the leaf has a different orientation from the one characteristic of the training set.
- *Shear transformations*. Such transformations are useful to classify images in which the leaf is depicted from a different perspective from the one characteristic of the training set (*i.e.* frontal perspective).
- *Horizontal and vertical shifts* (up to 100 pixels in both directions). Such transformations are useful to classify images in which the leaf is not centered.
- *Brightness shifts*.
- *Zoom* of the image (from 0.5x to 1.5x).

Moreover, since we performed *Transfer Learning* from *InceptionResNetV2* (see sections Architecture of the CNN and Choice of the Supernet and Hyperparameters), we also rescaled the pixels between -1 and 1 as required by such supernet.

Another important pre-processing step consisted in deleting corrupted samples from the dataset. Indeed, while removing the background some images were corrupted in the sense that the leaf was partially or totally removed from the picture, as shown in Figure 2. Therefore, we developed a Python script (*remove_corrupted.py*) in order to identify and remove a total of 21 corrupted images from the dataset.

All in all, we split the dataset into a training set (used to train the CNN) and a validation set (used to avoid overfitting the CNN). In particular, the validation set has been created by sampling at random (without replacement) 110 images from each of the 14 classes. This was done in order to obtain a balanced validation set: indeed, if we decided to sample a fixed proportion of the images then the performance of the CNN on the validation set would not be representative of the performance on the least populous classes (such as *Blueberry* and *Raspberry*).

Figure 2: Examples of corrupted images

N.B. We performed data augmentation only on the training images.

2 Model for the CNN

2.1 Architecture of the CNN

In order to achieve good performances we decided to apply *Transfer Learning* from a pre-trained supernet. On top of that, we developed our own network consisting in a *Flatten* layer, three *Dense* layers and four *Dropout* layers (introduced to prevent overfitting) as shown in Figure 3.

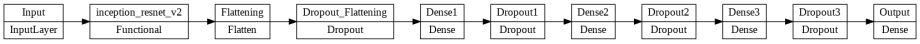

Figure 3: Architecture of the CNN

2.2 Choice of the Supernet and Hyperparameters

In order to choose which supernet to use, we fixed the number of neurons of the *Dense* layers (respectively 512, 256, 128) and the frequency for the *Dropout* layers (0.3). We then tried different architectures and observed that *InceptionResNetV2* was the best performing in our case: this has been our choice.

We then performed hyperparameter tuning in order to find more suitable values for the following:

- The number of neurons for each *Dense* layer;
- The frequency for each *Dropout* layer;
- The number of non-trainable layers of the *InceptionResNetV2* (*Fine Tuning* of the supernet).

The search of the hyperparameters space has been implemented in the notebook *hyperparameters.ipynb* via the library *KerasTuner*. The values obtained for the hyperparameters are the following:

- Number of neurons for the three *Dense* layers: 512, 480 and 192 respectively;
- Frequency for the four *Dropout* layers: 0.2, 0.3, 0.2 and 0.15 respectively;
- Number of non-trainable layers: 460.

3 Training

During the training phase, we used the Categorical Crossentropy as the loss function and the Adam algorithm as the optimizer.

Given the fact that the training set is imbalanced, the CNN gets biased towards the most populous classes so that it does not perform well on the least populous ones. Therefore, we decided to weight the loss function by assigning larger weights to the least populous classes according to the following formula:

$$w_i = \frac{1}{n_i} \cdot \frac{\sum_j n_j}{14} \tag{1}$$

where n_i is the number of samples of the i-th class. In this way we are essentially asking the CNN to "pay more attention" to samples from under-represented classes.

As for the training of the CNN, we observed that the accuracy on the validation set converges to an high value (above 99%) pretty fast whereas the accuracy on the training set increases slowly. This is probably because it is more difficult to achieve good performances on the training set due to the augmentation of the images. Since we are interested in achieving good performances also on augmented images, we decided to not resort to the *Early Stopping* technique. Instead, we monitored the validation accuracy "by hand" (to make sure we did not overfit the training data) and tried to achieve the highest possible accuracy on the training set. In particular, we trained the model in two phases:

- First, we set the learning rate to 10^{-4} and ran the training for 15 epochs;
- After that, we lowered the learning rate to 10^{-5} and increased the ε parameter of the Adam algorithm to 10^{-4}. We then ran the training for 20 epochs and saved the model achieving the best accuracy on the training set.

Such decision was guided by the fact that we observed that the learning becomes unstable after approximately 15 epochs, meaning that the accuracy on the training set does not increase consistently as epochs pass but oscillates between higher and lower values. Therefore, we modified the parameters of the optimizer in an attempt to stabilize the learning.

4 Assessment of the Results

The learning curve of the model is depicted in Figure 4, where the red vertical line divides the two training phases.

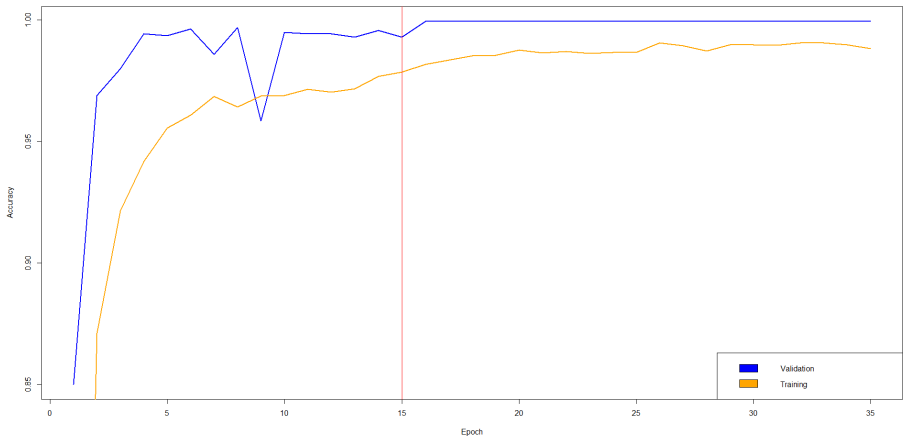

Figure 4: Accuracy on Training and Validation set for each epoch

As we anticipated, the accuracy on the validation set is very high and constant amongst the epochs (so that we can conclude we did not overfit the training data) whereas the accuracy on the training set keeps growing up to approximately 99%.

In order to assess the performances of the final model, we decided to test it on the *PlantVillage* dataset, which consists of 60343 (augmented) images of leaves belonging to the same species we considered during the training. In particular, the accuracy of the CNN on such test set is equal to 98.44% and the confusion matrix is shown in Figure 5. We can observe that the worst performing class is *Strawberries* and that sometimes *Potatoes* are misclassified as *Strawberries*.

All in all, we should mention that a further step to improve the performances of the CNN would be to deal with the problem of images for which the background has not been removed.

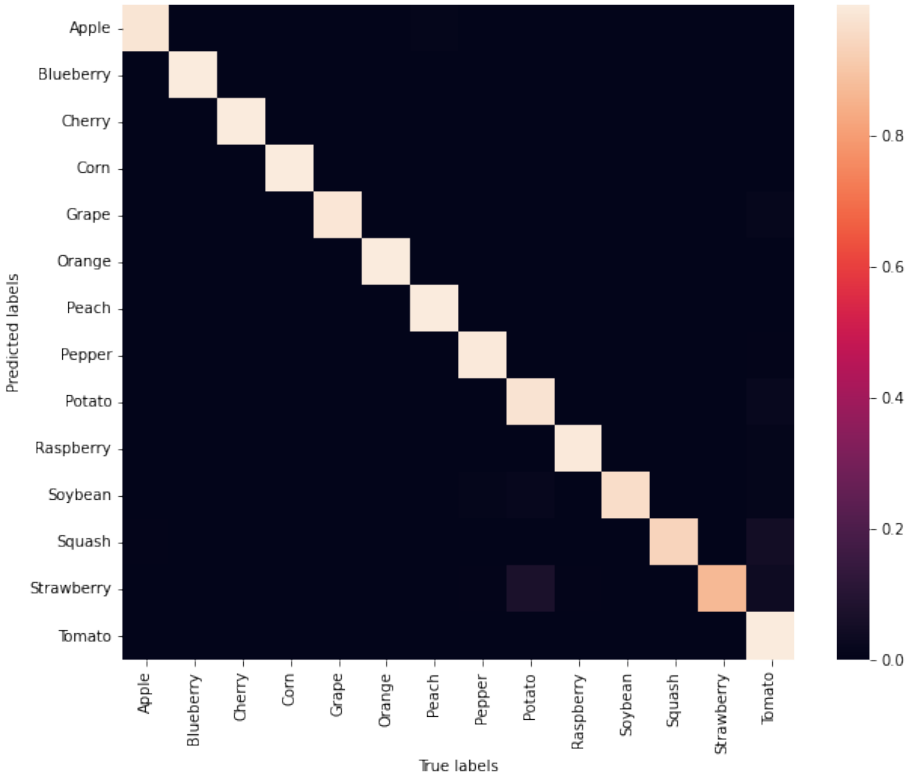

Figure 5: Confusion Matrix