Artificial Neural Networks and Deep Learning - Challenge 1

Samuele Tagliabue - 10563460
Marco Tagliafierro - 10602548

How the problem has been approached

The problem to be solved is a classification problem where the classifier has to assign an input image to the class corresponding to the largest score (weighted sum of all the image pixels).
Firstly, we analyzed the distribution on the dataset, consisting of 17728 256x256 colors images divided in 14 classes: we noticed that the images are not equally distributed (i.e. the Tomato's class is composed of 5693 images, while Blueberry's class is less than 500) - more on this will be discussed in the later sections.
Then we proceeded with the definition of the model and an iterative improvement on its performances.

Considered models and final model structure

In every model we trained there are some common features:

- To prevent overfitting, we have used early stopping in order to monitor the loss on the validation set and stop the training when the network doesn't improve for 10 consecutive epochs. Another technique we have adopted is the addition of dropout layers: this way each hidden unit has a probability of 0.3 of being set to 0 during training.
- The fully connected layers are composed by:
    - GlobalAveragePooling: we decide to use it because it is helpful to minimize overfitting by reducing the total number of parameters in the model
    - Dense layer: the first composed of 256 neurons and the output one composed of 14 neurons with a softmax activation function since the nature of the considered problem

For the features extraction we started from a convolution section composed by convolution2d layer followed by a maxpooling2D repeated five times to use as a sort of starting model to improve. With this solution we obtained an accuracy of 0.69 on the hidden test set (the one used on the evaluation server).

To improve our model we decided to use transfer learning where only the classifier layer of the network is trained: as starting model we have tested both VGG-16 and VGG-19 but we did not find big differences in accuracy between the two models, so we chose VGG-16

because it was faster in training; at this point our model obtained an accuracy of 0,81 on the hidden test set.
To further increase the performances of our model we decided to perform fine tuning on it: we made the last two convolutional layers of VGG trainable to improve the feature extraction.The results start to get good with an accuracy of 0.87.

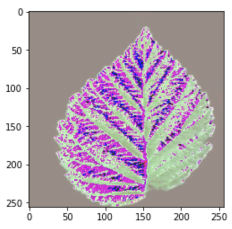
At the end we added a gaussian noise layer before the convolutional part of the model in order to mitigate overfitting (you could see it as a form of random data augmentation - more on this in the next section). We obtained a final accuracy of 0.91 on the hidden test set.

Pre-processing and oversampling

We noticed that the provided dataset has two main problems, which needed to be addressed before using it to train our models.

- The distribution of samples among different classes is not homogenous: while there is the *Tomato* class with more than 5500 samples, there are several ones with less than 1000. If not addressed, this clearly leads to models that are very precise in classifying leafs of some specific classes while other types of plants are penalized. In order to tackle this problem we used an external library, called *split-folders* (https://github.com/jfilter/split-folders), that allowed us to implement *oversampling* by copying multiple times images of less populated classes in our test set. Using this tool we were also able to have validation sets with the same size for each class.
- The samples contained in the dataset are noise-free and the position, dimension and rotation of the pictured leafs are quite regular. In order to build a more robust model we implemented different *data-augmentation* strategies, applied randomically:
    - rotation of the image, up to ±50 degrees
    - shift in both x and y directions, up to ±30 pixels
    - horizontal and vertical flips
    - zoom in the [0.75, 1.25] range
    - shear, up to ±10 degrees

  The *preprocess_input* function of VGG-16 model has also been used in order to adequate the images from the dataset to the format required by the used model; citing the official Tensorflow documentation, using this function as preprocessing *images are converted from RGB to BGR, then each color channel is zero-centered with respect to the ImageNet dataset, without scaling* (on the right is visible a sample of image used for training).

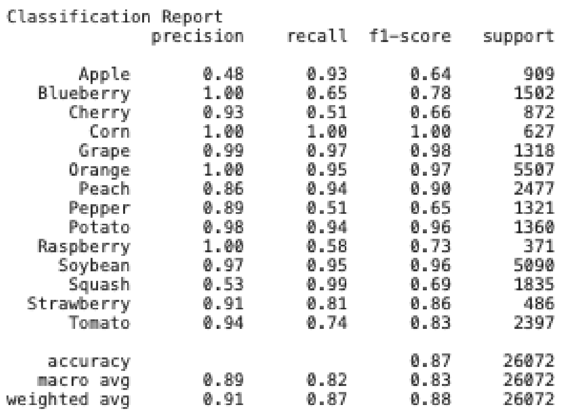
  Another important step taken to build a more robust model was including a GaussianNoise layer following the input one: while this step is separate from the data augmentation process, it allowed to train our model on images with a random noise applied to them leading to a better overall response to real images not taken from the training dataset.

Hyperparameters

Once we choose what we found to be the best performing model, in order to evaluate the hyperparameters different actions have been performed.

- The dataset has been divided into two different sets, one for training and the other for validation, using random-based functions with a fixed seed in order to obtain repeatable results. This allowed us to evaluate each epoch during training (see the next point).
- Every time a version of the model has been trained, only the best epoch has been considered using an *early stopping* callback; the patience parameter has been set to 10 since we noticed that with the considered models higher values didn't lead to any significant advantage.
  Another relevant parameter in this stage is the batch size: we found that a size of 16 has the best regularization effect on the training process without slowing it down excessively. Also a steps per epoch parameter has been added and tuned in order to speed up the training process without losing too much performance.
- We slightly modified each hyperparameter and evaluated the impact that the specific change had on the overall model performances using a test set obtained from a separate dataset from the one used for training and validation. This allowed us to use as many samples as possible to train our model and, at the same time, to test it on a more realistic set of images.

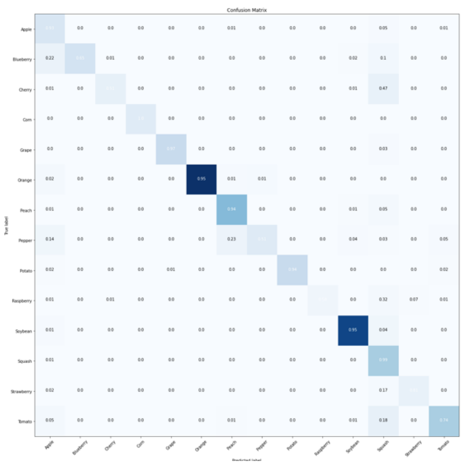
Several metrics have been used in order to evaluate different iterations of our model, like accuracy, precision and F1 - all visible in the example below.
We have also plotted the confusion matrix at each step in order to have a spot-on idea of the performances of our models for each class and better compare them: more than an absolute measure of the goodness of our implementations we were interested in having a comparison metric between them, in order to understand if the modifications made improved the performances or not. This consideration is particularly important since our test set is quite different from the training/validation ones and the observed performances are generally worse than the results from the submission server.



```
Classification Report
              precision    recall  f1-score   support

       Apple       0.48      0.93      0.64       909
   Blueberry       1.00      0.65      0.78      1502
      Cherry       0.93      0.51      0.66       872
        Corn       1.00      1.00      1.00       627
       Grape       0.99      0.97      0.98      1318
      Orange       1.00      0.95      0.97      5507
       Peach       0.86      0.94      0.90      2477
      Pepper       0.89      0.51      0.65      1321
      Potato       0.98      0.94      0.96      1360
   Raspberry       1.00      0.58      0.73       371
     Soybean       0.97      0.95      0.96      5090
      Squash       0.53      0.99      0.69      1835
  Strawberry       0.91      0.81      0.86       486
      Tomato       0.94      0.74      0.83      2397

    accuracy                           0.87     26072
   macro avg       0.89      0.82      0.83     26072
weighted avg       0.91      0.87      0.88     26072
```


Confusion Matrix
True label
Predicted label
Apple
Blueberry
Cherry
Corn
Grape
Orange
Peach
Pepper
Potato
Raspberry
Soybean
Squash
Strawberry
Tomato