AN2DL First Homework
Leaves Image Classification

Team Gamma

Rivi Gabriele 994131 10663569
Redaelli Mattia 995873 10622823
Ratzonel Ariel 995067 10631746

November 29th, 2021

1 Introduction

1.1 About the project

The goal of the project is to classify images of leaves, which are divided into categories according to the species of the plant to which they belong.

1.2 Dataset details

- Image details: 256x256 size, color space: RGB
- Number of classes: 14
- In total, the dataset consists of 17728 images.
  We divided the dataset in this way:
  - `Training data`: 70% of the dataset.
  - `Validation data`: 20% of the dataset.
  - `Testing data`: 10% of the dataset.

  Images have been read from the disk using the ImageDataGenerator object.

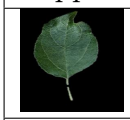
Apple	Blueberry	Cherry	Corn	Grape	Orange	Peach
						
Pepper	**Potato**	**Raspberry**	**Soybean**	**Squash**	**Strawberry**	**Tomato**
						

Table 1: Example of images for each class.

2 Design choices

2.1 First approach

We approached the problem starting from scratch, without performing any data augmentation. Unfortunately, our model scored badly on the private test set (approximately 0.23 of mean average precision). We upgraded our model and performed data augmentation, obtaining an accuracy on the private test set of 0.63.
A brief recap of how this first model was composed: 6 blocks of `Convolution` layer (starting from 16 filters arriving to 512 on the sixth block, kernel size=(3,3), strides=(1,1), paired with `MaxPooling` layers with size of (2,2). Finally, a `Flatten` layer and a `Dense` layer with 512 units (both coupled with `Dropout` at a 0.3 rate). The activation function used is `'ReLu'`. Along with data augmentation parameters, in this phase we set the parameter `rescale = 1/255`.
The next step was taking the previous model, and slightly upgrade it converting the

Flattening layer to a `GlobalAveragePooling2D` layer, since the latter highly reduces the number of trainable parameters and in general helps avoid overfitting, still keeping the `Dropout` with a rate of 0.3. This model reached an accuracy of 0.73.

2.2 What led to Transfer Learning & Fine Tuning

What was presented in the previous section was the best model we obtained starting from scratch, having implemented various version trying to tune the hyperparameters as best as we could, changing number of filters, kernel size on the feature-extractor part of the network, and also finding out which was the best number of convolution layers for the problem. We reached a bottleneck at approximately 0.70-0.73 of mean average precision, that's why we decided to tackle the problem by using the `Transfer Learning` technique. We trained different models starting from different supernets, but in the end we reached the best accuracy both on Codalab and in local testing using `ResNet152V2`.

2.3 Ultimate Model

The ultimate model we developed was created using `Transfer Learning`, starting from a `ResNet152V2` architecture and adding a `GlobalAveragePooling2D` layer followed by a `Dense` layer composed of 256 units and the typical output layer. Both the `GAP` and the `Dense` layers are paired with `Dropout` layers at 0.5 rate. The model was trained using a batch size of 32 and the Data Augmentation [1] performed on the images consists of these transformations:

- rotation_range=30
- zoom_range=0.15
- fill_mode='nearest'
- height_shift_range=0.2
- shear_range=0.15
- width_shift_range=0.2
- horizontal_flip=True
- preprocess_function [2]

We performed the training using an `EarlyStopping` technique (patience of 10) paired with `ReduceLROnPlateau` (introduced later in the developing phase), which is a technique that progressively adjust the learning rate when reaching a "saturated" level. Specifically, we monitored the `val_loss` imposing a factor of adjustment on the learning rate of 0.8, with a patience of 10 epochs (number of epochs with no improvement after which the learning rate will be adjusted), with a minimum reachable learning rate in general of 0.0001.
After training, we also `fine tuned` the model, freezing the first 550 layers and setting a `learning rate` of $1e^{-4}$ for the first time, then again freezing the first 527 layers and setting the `learning rate` to $1e^{-5}$.

[1] To note, in this phase using transfer learning the `rescale` parameter was cut out since the use of preprocessing_function

[2] We imported the specific preprocess function of the ResNet152V2 network to prepare the input in the required way.

3 Test Results

We used a google sheet file to keep track of the results of each model, in the image below we can see the improvement model by model.

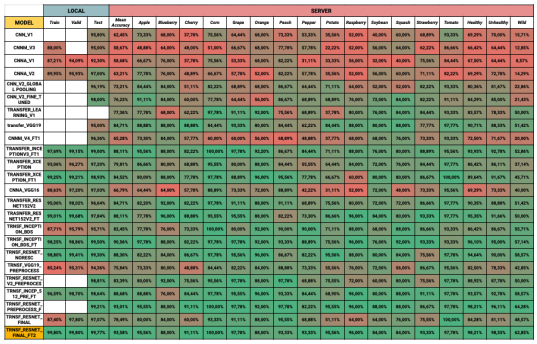


Fig. 1: Google sheet for the models evaluation.

3.1 Training plots

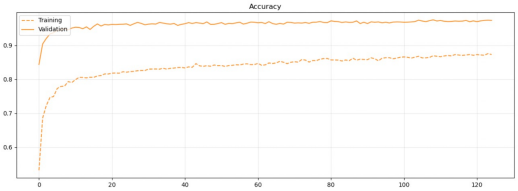


Fig. 2: Accuracy from the history of the training of the final model

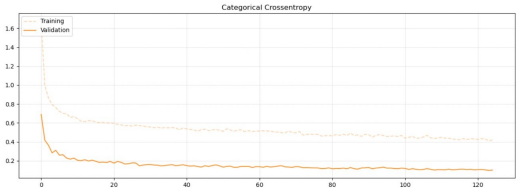


Fig. 3: Loss Function (Categorical Crossentropy) from the history of the training of the final model

3.2 Confusion Matrix

The test set used to obtain these scores has been generated from the complete dataset PlantVillage found on the internet, taking the ten percent of images from each class, to better simulate and analyze the behaviour of the model on a large number of data.

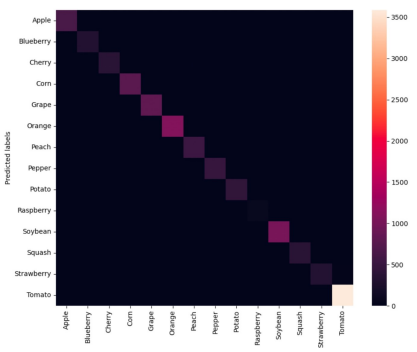


Fig. 4: Confusion Matrix

Classification Report

	precision	recall	f1-score	support
Apple	0.98	1.00	0.99	635
Blueberry	1.00	1.00	1.00	302
Cherry	1.00	0.97	0.99	382
Corn	1.00	1.00	1.00	772
Grape	1.00	1.00	1.00	813
Orange	1.00	1.00	1.00	1103
Peach	1.00	0.99	1.00	533
Pepper	0.99	0.98	0.99	495
Potato	0.98	0.99	0.98	432
Raspberry	1.00	0.95	0.97	75
Soybean	0.99	1.00	0.99	1018
Squash	0.99	1.00	0.99	367
Strawberry	0.92	1.00	0.96	313
Tomato	1.00	0.99	0.99	3632
accuracy			0.99	10872
macro avg	0.99	0.99	0.99	10872
weighted avg	0.99	0.99	0.99	10872

Fig. 5: Classification Report