First Challenge

Report

Artificial Neural Networks and Deep Learning

29/11/2021
A.Y. 2021/2022

Submitted by

Person Code	Names of Students
10582405	Castellano Sebastian
10578330	Ciapparelli Federica
10608654	Colapenna Francesca

POLITECNICO
MILANO 1863

1 Approach

The first thing we tried to do was to start from the practical reference notebooks from the professor and to begin to build a simple convolutional network composed by 5 Conv2d layers with padding 'same' and activation 'ReLU' plus Maxpooling, plus a Flattening layer with dropout and finally add a dense layer with 512 neurons and dropout (plus the output layer with a softmax).
This few numbers of layers proved not to be sufficient to do data extraction and classification on such a large dataset. As a consequence, our next approach was to exploit transfer learning techniques, importing some known Structures such as VGG_16, $IncpetionV3$, $Resnet$ and $Xception$, all with Resnet pre-trained weights, to take care of the data extraction part.

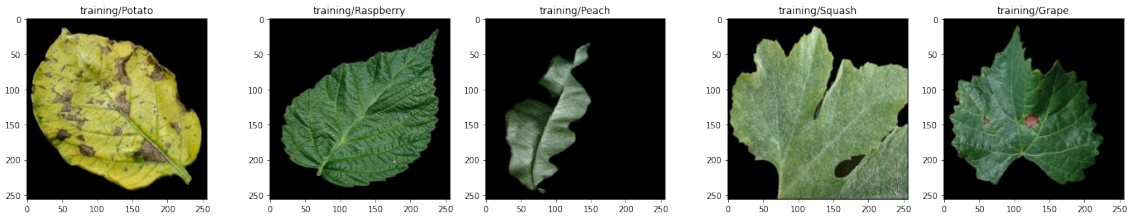
The first structure we took into consideration was $InceptionV3$, after which we added just a flattening layer and the output layer. After training the model we tested the accuracy through the submission on CodaLab; the result obtained was not satisfactory, we got about 54%.
The reasons of this low accuracy may have been the absence of sufficient image pre-processing, the absence of a dense layer between the flattening and the output, and not enough training epochs. (indeed in this first case we took into account only 10 epochs).

In the meanwhile we also tried to create models with the structures mentioned above: the $xception$ and VGG_16. At the same time we added a more complex dense model after the convolutional part, and we reached an interesting level of accuracy. Therefore we decided to focus on this type of structure.
In the next section we will explain in details our most accurate model.

2 Final Model: VGG_16

2.a Pre-processing

- **Data Augmentation**:
  Before importing our datasets, we took a look at some images representative of each class, as to better understand how our dataset images looked. We noticed that often the leaves had different orientation and zoom levels, while the background was uniform which was good.



(a) (b) (c) (d) (e)

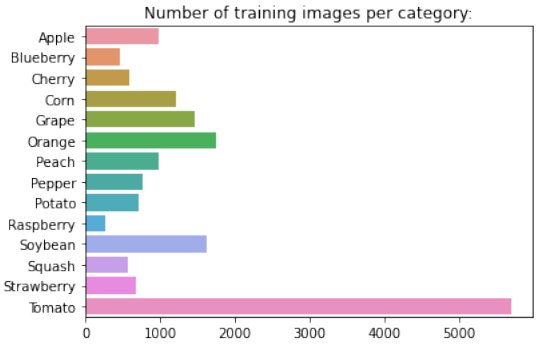
Figure 1: Some instances of leaf images belonging to different classes

Therefore in our ImageDataGenerator objects we decide to act on the following parameters:

Parameters	Meaning
rotation_range=30,	allows the rotation of the structure around the coordinated axes
width_shift_range=40,	apply displacement transformations on images horizontally, vertically, left or right
height_shift_range=40	apply displacement transformations on images horizontally, vertically, left or right
zoom_range=0.3,	apply image transformation to get enlarged / reduced image versions
fill_mode='reflect'	points outside the boundaries of the input are filled according to a given mode (reflect in this case)
horizontal_flip=True	applies an inversion transformation allows us to change the direction of our image horizontally
vertical_flip=True	applies an inversion transformation allows us to change the direction of our image vertically
rescale=1./255.	rescaling factor, we multiply the data by the value provided (after applying all other transformations)

In particular notice that out of all these data-augmentation choices, the rescale of our image is necessary to be applied also to the test set in order fro the model to give correct predictions.

- **Unbalanced dataset**:
  Another important thing to consider is the number of images belonging to each class, indeed we have some classes that greatly outnumber others in terms of number of images. In particular the Tomato class is very over-represented.



To tackle this issue we resorted to creating a dictionary of $weight\_classes$, in which we assigned to each key (going from 0 to 14 corresponding to our classes) a value which was inversely proportional to the number of images of the relative class over the total. In this way, by later passing it as a parameter in the $model.fit$ step, it will teach the model to give more importance to the misclassification of images belonging to under-represented classes.

2.b Structure of the Network

As we said we used the VGG_16 structure, composed by 13 convolutional layers plus many pooling layers and a final $GlobalMaxPooling2D$ layer. Moreover t this, we added the final dense layer. In our case we decided to add two hidden layers of 200 Neurons each with ReLU activation and the final output layer.

- **Fine-tuning**:
  In the VGG_16 structure we chose to set as Trainable only the last three layers (meaning only the last Convolutional layers). In this way we found out that we could benefit both from the transfer learning

pre-trained weigths of the network while at the same time performing some fine-tuning on the feature extraction done which better fit our purposes.

- **Compiling/Fit of the model**:
  Seeing the purpose of our model, we chose to use as loss Categorical Crossentropy, as metric Accuracy, and monitored the validation loss through Early Stopping callbacks.

- **Results**:
  After training for 15 Epochs, the validation Accuracy reached a satisfying 95%.

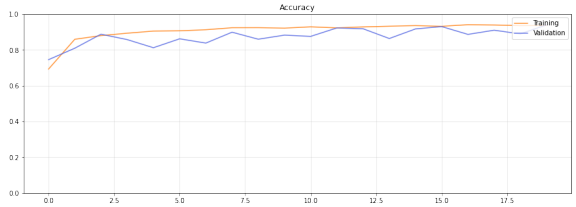


Figure 2: Accuracy given by the VGG_16 Structure

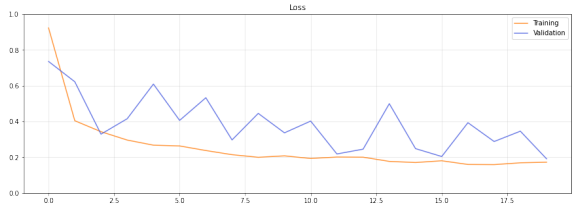


Figure 3: Training vs val loss of the VGG_16 Structure

After submitting, the test set accuracy was 80%, but this decrease is to be expected when fitting on a different test set.
Our network classified best the Classes Apple and Tomato (95% Accuracy) and worst the Orange and Pepper classes (64% Accuracy). These results confirm that passing the weight of the classes to our $model.fit$ was effective, indeed we don't see any meaningful correlation in the number of images of a class and its Accuracy (indeed the less represented class, Raspberry, achieved 84% Accuracy). Moreover by looking at the sub-classes of Healthy/Unhealthy/Wild leaves, we can say that our network doesn't struggle to classify unhealthy leaves with respect to healthy ones (bot around 85% Accuracy).
Regarding Wild Leaves, we can see that these are the most difficult to classify.
If we were to give an explanation on why this happens, this is probably due to the nature of our training data-set which may have been mostly composed of non-wild leaves. Indeed some different features of wild leaves may differ form the non-wild ones and may confuse the classification.