Politecnico di Milano

Artificial Neural Networks and Deep Learning
First Homework

Assignment: Image Classification

Anas Bahtaoui, Théo Saulus
Centrale International Team

November 28, 2021

Description of the homework

In this first homework, we aim to classify images of leaves that are divided into classes. We have 14 classes distinguishing different species of leaves.

Our approach was to work first using a handcrafted neural network using data augmentation. And then we tried building a model using transfer learning in which we tried both the VGG16 and InceptionV3.

Data preparation

The dataset provided showed big discrepancy regarding the number of pictures accessible (more than 5000 samples for tomatoes vs. about 250 for Raspberry), which can cause the network to overlook some classes compared to others. That is why we decided to implement several methods to take the most out of our data.

First, we decided to weight our class, which modifies the computation of the loss to increase the weight of underrepresented classes.

Then, we decided to perform oversampling and undersampling on our data, in particular in the case of the extremal classes. Due to time constraint, we did not have time to implement a "clean" oversampling/undersampling algorithm, and therefore modified the dataset manually. Thus, we cut by half the number of images of tomatoes, and almost doubled the number of images of the raspberry set.

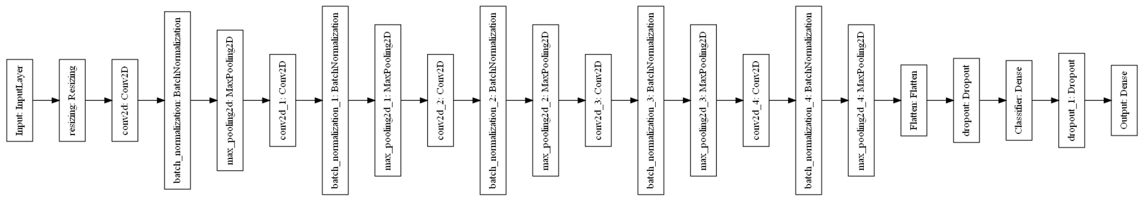
To train our network to a more diverse set of images, and to avoid to compute several times the same identical image after oversampling, we used data augmentation. We used slightly conservative values (only 15° of rotation, 10 of translation) because we assumed that the final test set would be photographs of leaves approximately corresponding to our training set (that is: cantered on the leaf, not rotated too much).

Finally, in order to be able to test our networks before submitting it to Codalab, we created a separated test set, containing 6 images from each class, completely separated from the training data. We were therefore able to plot a confusion matrix and estimate a test accuracy, to avoid any kind of bias on the validation set (in the sense that we may tune the parameters to overfit indirectly the validation set). Among the remaining images in the training set, we chose to use 0.20 of them in validation.

In-house Neural Network

Even if we were aware that transfer learning methods would provide better results, we experimented with a classical CNN. The architecture we came up with is inspired by VGG16, but with only one convolution before each max-pooling layer, and batch normalisation before pooling.

The first layer after the input is used to resize the images to 64×64, to accelerate the training time (either to avoid disconnection on Colab, or to make the training faster on a local computer).


Input: InputLayer
resizing: Resizing
conv2d: Conv2D
batch_normalization: BatchNormalization
max_pooling2d: MaxPooling2D
conv2d_1: Conv2D
batch_normalization_1: BatchNormalization
max_pooling2d_1: MaxPooling2D
conv2d_2: Conv2D
batch_normalization_2: BatchNormalization
max_pooling2d_2: MaxPooling2D
conv2d_3: Conv2D
batch_normalization_3: BatchNormalization
max_pooling2d_3: MaxPooling2D
conv2d_4: Conv2D
batch_normalization_4: BatchNormalization
max_pooling2d_4: MaxPooling2D
Flatten: Flatten
dropout: Dropout
Classifier: Dense
dropout_1: Dropout
Output: Dense

Using our in-house CNN, we managed to obtain 0.58 precision on the test set, after tuning the use of L2 regularisation, the dropout placement and intensity, and number of convolutions.

To be more precise, we finally decided to remove the L2 regularisation from our convolutions and dense layers, after we noted that it had little effect on the prediction (it only seemed to slow down the learning rate). These qualitative remarks seem to be partially confirmed by articles we read [1, 2], even though we did not have time to read and understand in detail why and how the behaviour is actually modified when using both batch normalisation and L2 regularisation.

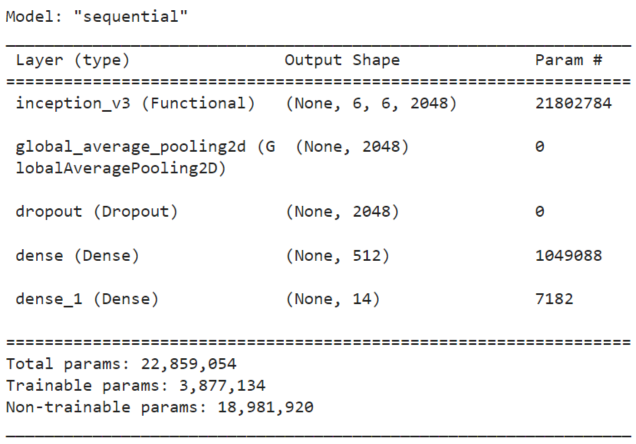
Transfer Learning

For the transfer learning approach, we have used a class of pre-trained models. We namely tried VGG16 and InceptionV3.

Same as before, we took into account the fact the classes are unbalanced by using the class weights. However this time we split our training data into training and validation sets only.

Using VGG16 followed by a global average pooling layer, a dropout layer, a dense layer with a relu activation function and then an output layer with a softmax activation function, we obtained an accuracy of 0.84 on the test set.

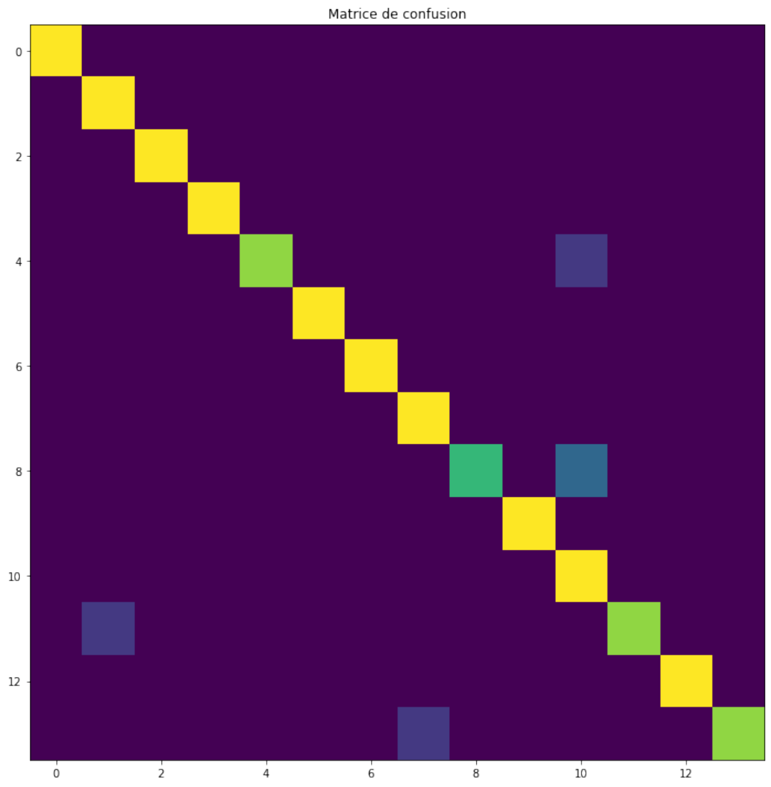
Using InceptionV3 followed by the same structure we trained the model. And we did some fine tuning but freezing the first 285 layers and unfreezing the rest of the layers. This is our final model that yielded an accuracy of 0.9 on the hidden test set.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 inception_v3 (Functional)   (None, 6, 6, 2048)        21802784  
                                                                 
 global_average_pooling2d (G  (None, 2048)             0         
 lobalAveragePooling2D)                                          
                                                                 
 dropout (Dropout)           (None, 2048)              0         
                                                                 
 dense (Dense)               (None, 512)               1049088   
                                                                 
 dense_1 (Dense)             (None, 14)                7182      
                                                                 
=================================================================
Total params: 22,859,054
Trainable params: 3,877,134
Non-trainable params: 18,981,920
_________________________________________________________________
```

Plots and analysis

Confusion matrix, example on our in-house network, tested on our test set (6 images per classes):



References

https://arxiv.org/abs/1706.05350v1

https://blog.janestreet.com/l2-regularization-and-batch-norm/