Challenge 1: Image Classification

Dario Del Gaizo
Valeria Detomas
Chiara Fumelli

Group Epoch404

1 Overview

The goal of the A.Y. 2021/2022 first competition is to solve an image classification problem. We performed classification of 14 different classes given a dataset containing images of leaves. The framework used during the competition is tensorflow.

2 Data

The uploaded dataset is composed of around 17K images of different types of leaves (e.g. Tomato, Apple, Raspberry etc.) on black uniform background, all centered. The images' dimension is 256x256 on a RGB scale. We decided to keep it this way for a good performance. Something that has to be addressed is the severe image uniformity of the dataset: to avoid this problem we decided to implement Data Augmentation with the following parameters:

```
ImageDataGenerator(rotation_range=30,
                   width_shift_range=25,
                   height_shift_range=25,
                   zoom_range=0.3,
                   horizontal_flip=True,
                   vertical_flip=False,
                   fill_mode='reflect',
                   rescale=1./255)
```

Figure 1: Data augmentation parameters.

3 First Approaches

The very first approach we felt like trying was based on the models seen during the practice lectures: a basic CNN with data augmentation performed on the training set, followed by training parameters adjustion. Our CNN reached average local results and awful leaderboard positioning, because of a mix of poor number of layers and bad preprocessing.

4 Early Stopping

With a high number of epochs, early stopping prevented overfitting. We set a patience of 10 (with epochs=100). The model performed even better with the final test set, backing our hypothesis of insignificant overfitting.

5 Transfer Learning

After our first CNN models we decided to advance with a transfer learning approach. Taking a model trained on a large dataset (Imagenet) to our small dataset improved our accuracy. With vgg16 we reached an accuracy value of 0.66, then improved by adding fine tuning and better preprocessing

(0.87). However, we noticed that the model was very large and training was quite slow. For these reasons we decided to explore new pretrained models. The one architecture that reached the best accuracy result is Xception. The advantage of using this model wasn't just that: the number of parameters was much smaller and the file size was reduced. This made us able to try and explore more models.

6 Weight balance

After we applied transfer learning to our model, it seemed to obtain a very good accuracy on some classes, but not so good in other ones. We noticed that the dataset was quite imbalanced. There were different solutions for coping with this problem: undersampling, oversampling and reweighing the class weights. We thought that our dataset was too small to be undersampled and oversampling our dataset would have led to overfitting. We came to the conclusion that reweighting class weights would have been the best solution. By doing this, classes with less images were given more "importance" in respect to classes with more pictures. This led to a slight improvement in our accuracy.

7 Retrain with fine tuning

Techniques as fine-tuning are implemented to overcome problems such as poor datasets and to speed up the training. In this particular case we started from the pre-trained model, freezing the first convolutional layers and just leaving the last five convolutional layers. In this way the latter layers identified more specific features of our images dataset.

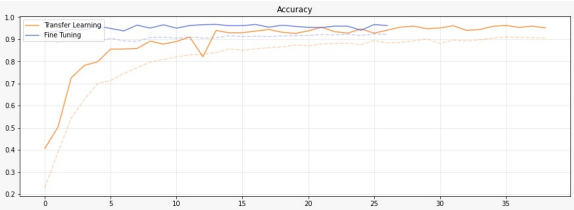

Figure 2: Final vgg16 model.

In Figure 2 it can be noticed that with transfer learning the accuracy increases dramatically in the first ten epochs. Both transfer learning and fine tuning reach very high accuracy values but with the test set they were not as much accurate as seen with the validation set:

- 0.83 vgg16 without fine tuning
- 0.87 vgg16 with fine tuning

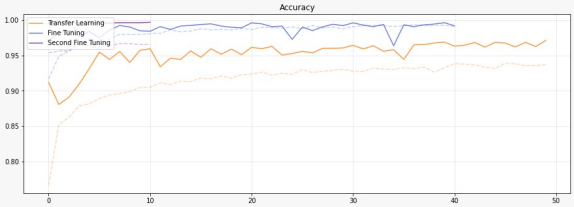

Figure 3: Final Xception model.

Figure 3 shows accuracy values of our final model. We reached the best result starting from the Xception pre-trained model, applying a first fine tuning by freezing the first 126 layers. After training this first model we reached an accuracy of 0.89. We decided to retrain it by freezing extra three layers to identify more specific features of a new augmented training set(added brightness_range and shear_range).

- 0.89 Xception with fine tuning
- 0.91 Xception with fine tuning and retraing with new data augmentation