# Artificial Neural Network and Deep Learning

Homework 1

Bita Rahmat Zadeh-10758773
Héloïse Viossat-10847709
Gabriele Bruni-10804805

November 28, 2021

## 1 Introduction

Aim of the homework is to train a neural network classifier to predict 14 categories of leaves. We briefly introduce the dataset and discuss some statistics about it. Next we present the models used to accomplish the homework and the experiments conducted. Finally, we show and discuss the results obtained with the final model selected and relative conclusion.

## 2 Dataset

Our dataset is composed of 17.728 RGB images belonging to 14 classes with dimension of $[256, 256, 3]$. A quick inspection to the target distribution show that data might suffer of class imbalances problem. This problem will be addressed later in section 4.

## 3 Model

### 3.1 Baseline model

Our baseline model is a Convolutional Neural Network composed of 3 convolutional layer, very similar to the one introduced in the introductory laboratory lesson of the course. Between each convolutional layer we place a max pooling layer, except for the last one where we place an average global pooling operation, for a total of 440,206 trainable parameters.

### 3.2 Xception model

To improve over the baseline model we choose Xception, an architecure introduced by Google in 2016 based on Inception V3, see reference [1] for more details. We opted for Xception because of the superior accuracy performance with respect to VGG16 (introduced in laboratory class) on the ImageNet dataset [1]. The Keras implementation is composed of 131 layers.

## 4 Experiments

For every test, we reserve the 20% of the training data for the validation set. To prevent overfitting we place a Dropout layer on top of the Dense layer for each tested model and we use Early Stopping on validation accuracy. All the model was trained optimizing Categorical Cross-Entropy.

First, we train our baseline model. The result are discussed in section 5. Then we decide to directly training the more complex Xception model to empirically evaluate the difference and the possible improvements. During this step, classical Transfer Learning procedure was performed: we load a pre-trained Xception model on ImageNet dataset, extract the convolutional part and used as features extractor, training a dense net on top of it. Subsequently, because of the superior result obtained, we decide to improve on top of this result and discard the baseline one. We fine-tuned this model freezing the first 100 convolutional layer and make the last 31 layer trainable. The test was performed loading the model trained in previous step with the relative weights. In the next step, we performe some data augmentation to the dataset using the classical Keras api. As previously, we take the model trained in previous phase with the relative weights. Then, we manage the class imbalances problem using the class weight parameter of Keras api, making training procedures

"pay more attention" to examples from an under-represented class. As final trial, we try to fine tuned more layer (51, 20 layer more) of the model obtained in the last step. Data for all the training phase done with Xception model was preprocessed as inidicated in the relative Keras implementation documentation. Batch size was keeped fixed to 256 for all experiments.

# 5 Results

## 5.1 Baseline model

While obtaining good training performance, on validation set our baseline model perform poorly, see the score below:

```
Accuracy: 0.3736
Precision: 0.3935
Recall: 0.391
F1: 0.2951.
```

## 5.2 Xception - base

Our first attempt with Xception model show encouraging result on validation set:

```
Accuracy: 0.9675
Precision: 0.9602
Recall: 0.9649
F1: 0.962 ,
```

but the result wasn't replicated on test set. Even if we do not have training curve to show (we forget to plot it and history training of the model was lost), we conjecture that the model was badly overfitting.

## 5.3 Xception - Fine Tuning

Fine Tuning still improved the score on validation set, but the gap with respect to the test result was still high, even if noticeable better:

```
Accuracy: 0.9819
Precision: 0.9776
Recall: 0.9788
F1: 0.9777
```

## 5.4 Xception - Fine Tuning + Data Augmentation

Performing data augmentation on the previous model only still slightly increase the validation score, but more importantly, fill the gap with respect to test set:

```
Accuracy: 0.9895
Precision: 0.987
Recall: 0.9895
F1: 0.9881
```

## 5.5 Xception - Fine Tuning + Data Augmentation + Class Imbalance

Managing the class imbalance gives a slightly ulterior boost to validation performance, reflected also in test set score:

```
Accuracy: 0.9918
Precision: 0.9899
Recall: 0.9943
F1: 0.992
```

### 5.6 Xception - Fine Tuning(more fine tuned layer) + Data Augmentation + Class Imbalance

Fine tuning more layer doesn't produced noticeable improvements:

```
Accuracy: 0.9929
Precision: 0.991
Recall: 0.9915
F1: 0.9912
```

## 6 Conclusion

The final model was selected based on validation score: even though fine tuning more layer doesn't seems to produce noticeable result, the slightly better validation score lead us to choose the model proposed in section 5.6. Before to submit for final evaluation, the selected model was trained again on the entire training data, including the one in the validation set. With this settings (Xception - Fine Tuning(more fine tuned layer) + Data Augmentation + Class Imbalance), we obtain a test score of 0.9283018868.
As final note, we can probably obtain more higher result performing a cross-validation to better tune the hyperparameters and learning rate of such model. More-over, we don't have investigate clearly the impact of the weighted loss function doing a full retraining, without applying them to an already trained model: perhaps we could have run into in a more promising local minima of the loss function. Would have been interesting also explore how further augment the data using Generative-Aversarial-Network impact the training phase.

## References

[1] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *CoRR* abs/1610.02357 (2016). arXiv: 1610.02357. URL: http://arxiv.org/abs/1610.02357.