ANNDL 2021 Homework 1 - GengisGAN

Marco Ferrè, Stefano Iachini, Gioele Mombelli

November 28, 2021

1 Introduction

The first Artificial Neural Networks and Deep Learning Challenge is based on classifying images of leafs divided into categories according to the species of the plant to which they belong. Our models needs to be submitted and tested on a *blind set* and the given result gives us a position in the general Leaderboard.

2 The Development

In order to reach the goal we decided to follow a progressive path from the basics to more advanced solutions improving our results.

2.1 Our First CNN

- `CH1.ipynb`

This is the starting point for the whole challenge. In order to experiment and to take confidence with the environment, we tried to adapt the simplest CNN we have seen during the exercise sessions. The performance on our validation was great but the submit phase break our dreams, result: **19,24%**. This Model needs to be improved to be more general to perform better on the blind set.

2.2 Data Augmentation (Preprocessing and Parameters Tuning)

- `CH1 DataAug.ipynb`

The first technique we decide to adopt was Data Augmentation. We started with the basic settings provided in Labs and then we tune it.

Driven by the fact that leaves have, in general, a vertical symmetry and after a visual examination of the dataset where we noticed that the leaves are captured with different angles (according to the position on the table) we adopted an augmentation with a $\pm 180°$ rotation. We also noticed that in general all the photos has black solid background so we set a constant-filling policy.

Moreover we decided to perform *Rescaling and Normalization* in every model we trained.

This evolution in data augmentation is described step-by-step in the next sections. In *Figure 1* the final result.

2.3 Xception

- `CH1 Xception.ipynb`
- `CH1 Xception Aug180.ipynb`
- `CH1 Xception FancyAugmentation.ipynb`

The exercise session's architecture gave us a good insight on the task, but we couldn't manage to push it over 57% of accuracy on the hidden online test. After some research, we decided to drop this implementation and try a new architecture, in order to experiment also with CNNs that were not explained during the lectures.

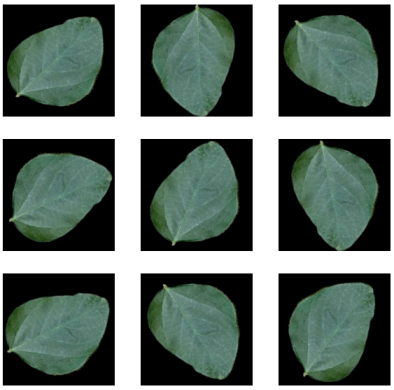

Figure 1: Data Augmentation on Leaves

Our goal was to implement a network that could obtain good performances on image classification, but we also wanted it to be fairly simple and relatively new, in order to try one of the modern solutions. We chose to implement the Xception Neural Network, and we found some good implementation of it in the Google Colab tutorials. As can be seen from the notebook `CH1 Xception`, we kept the same skeleton seen during the exercise sessions but we changed the network, implementing *Xception* with the same data augmentation of the previous solution. We also keept the Early Stopping active on the validation loss.

This first version of *Xception* gave us a **62.6%** of accuracy on the hidden test, a better result than the previous architectures, given the fact that the network basically trained on the same data of the previous ones. We believed that there was still room for improvement, so we tried to push further the data augmentation part. This led to two different solution: one with the data augmentation acting on the rotation degree in order to have the rotation in the range $\pm 180^\circ$ (thus covering every orientation possible), and the other one also acting on different brightness of the images. The code can be found in the notebooks `CH1 Xception Aug 180` and `CH1 Xception FancyAug`.

By manually inspecting the dataset, we noticed that some leaves presented imperfections such as illinesses, missing parts and different shades of colour. We thought that a variation of the brightness of the leaves would make the network more able to distinguish ill leaves. Actually, the version with only the rotation of 180° obtained **76%** of accuracy, while the augmentation on the brightness led to worse performances, with a score of **74%**.

Given the fact that we could not perform further augmentation, both because of the scarcity of time and the complexity of the augmentations needed to match the ill leaves, we decided to stop our experiments with the *Xception* architecture and we moved on to another approach based on Transfer Learning.

2.4 Our First Transfer Learning

- `CH TransLear.ipynb`

2.5 Transfer Learning from InceptionV3

- `CH1 InceptionV3 TLFT.ipynb`

After various attempts with the Transfer Learning example from the exercise sessions, we decided to perform Transfer Learning by using a different architecture as a supernet.

We read some papers and articles about transfer learning from image recognition, and we found that there were some pre-trained networks available, which used weights learned on the Imagenet dataset. Among all the architectures, we decided to implement Transfer Learning from the *InceptionV3* model. The structure of the network is quite complex and the size of the model produced is large, but we decided to try it anyway because of the known good performance of the architecture on image recognition.

As can be seen from the notebook, we started by importing the pre-trained network without the top layer in order to be able to built it manually, adapting the network to solve our classification problem. In this notebook we decided to perform two different trainings: one that built a network with pure transfer learning, and another one with also the fine tuning part.

The pure transfer learning part needed the same pre-processing of *InceptionV3* to be applied to our dataset, which we did by exploiting the built in functions of Keras. We also relied on early stopping on the validation loss, because we wanted to be fairly sure of our predictions, even if this could lead to lower values of accuracy. Moreover, we also performed some data augmentation, applying random rotations to the images on the training set.

Regarding the Fine Tuning part, we decided to start by leaving 80 layers free and freezing the remaining 700, for a total count of 780 layers. After training the two models, we decided to submit the fine tuned one, because it obtained better results in validation. We obtained the highest score up to that moment, with an accuracy online of **83%**.

2.6 InceptionV3 TLFT: hyperparameters tuning

- `CH1 InceptionV3 TLFT 2.ipynb`
- `CH1 InceptionV3 TLFT ExtremeTrainingExtended.ipynb`

The performance of the *InceptionV3* model were really good, and we decided to improve it by performing some hyperparameter tuning. Since we wanted to exploit the capabilities of fine tuning, we trained different networks, each with a different number of free layers. We found that more free layers led to better performances (and longer training times, even with early stopping). We tried to freeze only 500 layers, then 100 and finally we just left one layer frozen, the first one.

Each version outperformed the previous one, so we decided to send the network trained with just 1 layer frozen, obtaining **90,94%** accuracy on the online testbench.

This implementation can probably be improved even more, by letting it train on a larger training set or by tuning other parameters in the network, but we decided to keep it as is in order to avoid a possible overfitting on the data available on the online platform.

3 Conclusion and Final Phase

Since the start of the challenge we knew that Transfer Learning would have brought the greatest improvement. However, for didactical purposes, we decided to try an incremental approach where started from simpler architectures moving towards more complex ones, and only in the end applying Transfer Learning.

Our final submissions were made using three Transfer Learning networks that imported the *InceptionV3* trained on imagenet dataset, and they obtained the following results:

- **92,26%** - Transfer Learning from InceptionV3, 1 layer frozen and training performed on extended dataset
- **91,69%** - Transfer Learning from InceptionV3 and freezing only 100 layers
- **90,75%** - Transfer Learning fom InceptionV3 and freezing only 1 layer