Artificial Neural Network and Deep Learning

Homework 1
Image Classification

Team Members: *Gabriele Marra, Matteo Miceli, Giuseppe Piccirillo*

1 Overview

The homework aims to classify leaves according to the plant they belong to. Given a dataset of 17728 images (256x256 pixels, RGB) already divided into the 14 types of plants, we were supposed to build and train a neural network able to predict the plant type given a leaf.

2 Testing models

We decided to start with a basic convolutional neural network to test the submission platform and the notebook environment.

After the first attempt, we tried to implement and test all the techniques we learned.

2.1 Getting familiar: CNN

As the first model, we decided to implement a simple Convolutional Neural Network with 5 convolutional layers, each one followed by a ReLU activation layer and a 2x2 MaxPooling.

The top of the network is built with a Dense layer of 256 neurons, two Dropout layers, to end with a 14 neurons layer with a Softmax activation.

The dataset has been split into training set (80%) and validation test (20%).

This model has 8,788,910 parameters in total and we obtained an accuracy of 0.19 on CodaLab.

2.2 Data Augmentation

Given the poor result obtained previously, we developed a series of new models applying Data Augmentation techniques like increasing or decreasing the rotation range, the width shift, the height shift, and the zoom.

We tried also to significantly increase the batch_size from 8 to 256 (with intermediate steps, like 32 and 64) and decrease the learning rate to $1 \cdot 10^{-5}$ to reach a better loss computation and a slower correction of the trained weights.

We also split differently the dataset: the training set contains now 90% of the images provided.

The model obtained has a structure similar to the previous one, but accuracy has improved remarkably: we scored 0.66 as maximum.

2.3 Transfer Learning

Looking for state-of-the-art research on plant recognition we noted that Inception was remarkably better than VGG16 so we directly started by implementing a TL model using inception.

The first basic model used only Inception V3 with all the layers set to not trainable, a GAP layer following that, and ending with a 14 neurons layer with a Softmax activation that got a score of 0.80 on CodaLab.

We then tried different configurations varying batch size from 128 up to 512, different learning rates ranging between $1 \cdot 10^{-3}$, and $0.5 \cdot 10^{-5}$ and setting to trainable various layers of the inception architecture. We also added, after the GAP layer, some dense layers of different sizes with a Dropout of diverse probability before the final Softmax layer.

2.4 Subsampling

Given the fact the dataset was unbalanced, we did a subsampling operation to balance it.
We thought in this way the accuracy metric would be better.
By the way, we didn't obtain significant improvements.

2.5 Honourable Mentions

Also for the unbalance of the dataset, we thought about using the f1score instead of the accuracy as a metric. However, the fact that accuracy was the metric evaluated on CodaLab combined with the fact that we noticed a strong correlation between an improvement in accuracy on the validation set and an improvement in score on the submissions, led us to use accuracy instead of f1score.

After noticing on the CodaLab submission the score on ”Wild mean accuracy” was by far the lowest of the three different means, we tried to add GaussianNoise layers to improve the robustness of the network but we didn't get any improvement on the score.

3 Final decision

Tweaking all the hyperparameters of the Transfer Learning model explained in section 2.3 we ended up with this model:

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 6, 6, 2048)	21802784
global_average_pooling2d (Gl	(None, 2048)	0
flatten (Flatten)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 1024)	2098176
dropout_1 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
dropout_3 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
dropout_4 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 14)	1806

Total params: 24,591,790
Trainable params: 24,045,470
Non-trainable params: 546,320

Figure 1: Summary of final implementation

The first 50 layers of the inception network are set to not trainable. All the Dense layers at the end are using a LeakyRelu with a customized coefficient. They all also have a Dropout of 0.3 or 0.35. We used a learning rate of the Adam optimizer of $0.5 \cdot 10^{-5}$ and a batch size of 256.

With this model, we obtained an accuracy of 0.9377 in the first phase and 0.9340 in the second one.