Leaves Classification

Artificial Neural Networks & Deep Learning - a.y. 2022/2023
Challenge 1 Report

Beatrice Insalata
Codalab Name: B34
Team Name: Personal Trainers
10708628 - 222966

Lorenzo Mauri
Codalab Name: Lorma200
Team Name: Personal Trainers
10696228- 225949

Matteo Pancini
Codalab Name: matt4pa4n
Team Name: Personal Trainers
10656944 - 223085

INTRODUCTION

The objective of this challenge was to correctly classify 8 kinds of different plants through a dataset of their leaves images as input. In this report we discuss, explore and compare our development and design process.

1. PRELIMINARY DATASET ANALYSIS

1.1 DATA DISTRIBUTION AMONG CLASSES

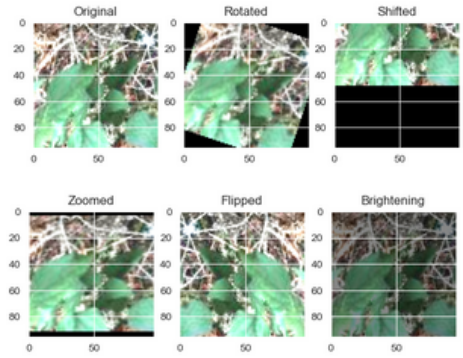
After a quick dataset observation we plotted the distribution of images among classes to find out if they were equally distributed, obtaining the following result:

Stratified sampling procedure was used to correct the unbalancement in classes and maintain the same distribution in both training and validation set.

To balance the bias that this unequal distribution would have caused during the training phase, we calculated a weight for each class to be introduced. This allowed us to give higher importance to classes with fewer samples. Every weight was calculated based on the overall number of samples for each folder.
Since data was extremely limited, we decided to split the dataset into training (80%) and validation (20%). Only later we changed to a 60-20-20 split to be able to test the models on fresh data.

1.2 DATA AUGMENTATION

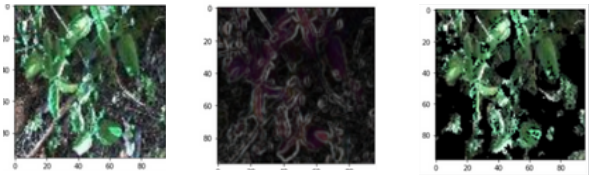
We used Data Augmentation to better handle the reduced dimension of the dataset. With Keras ImageDataGenerator several transformations of the pictures were created for better generalization.
We tried to input in the function different parameters until we obtained a sufficiently good outcome. A rescale factor of 1/255. was preferred to the keras preprocessing function after result comparison.



Side by side view of non-augmented image and a dataset element after some random augmentation

1.3 CUSTOM PREPROCESSING FUNCTION

Few different preprocessing techniques were applied. Although those experiments didn't bring much improvement on the performance, it has been interesting exploring the possibility of different preprocessing approaches, such as the Skimage Sobel Filters, custom Background Removal and Color Background Removal for better image recognition.



Standard image compared with Sobel filter application and Color Background Removal

Sobel filters and background removal emphasized the shape of the plants, but the images ended up being too low quality to successfully apply those techniques. We also applied a color mask to focus on the parts of the images in a specific range of shade, taking the image to hsv format and putting it on. The result of this approach looked much more promising, even though overall outcomes did not rise in accuracy.

2. HAND-CRAFTED CNN MODEL

We initially manipulated the CNN shown during the labs. 5 Convolutions with an increasing number of filters (32, 64, 128, 256, 512) gave us the best outcomes.

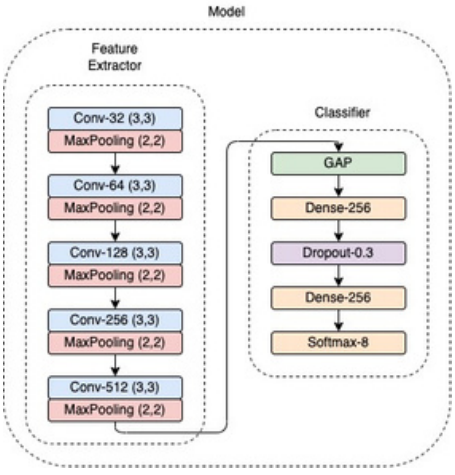
The Flatten Layer was switched with a Global Average Pooling Layer in order to lighten the structure of the network.

We chose to regularize and generalize the model to avoid overfitting, adding dropout and batch normalization.

Changing the classificator from 64 to 256 units and adding another classifier layer of 256 units produced better results, while trying different activation functoin did not give significant improvements.

We finally tried to add weight decay regularization with a manual hyperparameter tuning between λ=0.01, λ=0.001 and λ=0.0001, preferring the second one.

EVOLUTION	VAL_ACCURACY
Base Model	0.47
Dropout + Batch Normalization	0.53
Second Classifier Layer	0.63
WeightDecay	0.65



Structure of custom built CNN

3. TRANSFER LEARNING MODELS

We started comparing the most famous CNN architectures examined during the lectures through Transfer Learning, followed by a Global Average Pooling + Dense Softmax Layer as the simplest classifier.

NET	VAL_ACCURACY
VGG16	0.6374
DenseNet201	0.7178
Xception	0.5439

Each group member was assigned a different architecture in order to optimize and then compare them. We manage to get satisfying results from everyone.

3.1 VGG16

VGG was our most tested model and the baseline comparison for other nets and techniques.

We experimented with different numbers of layers, neurons and dropout rates. The accuracies obtained varied according to the specific configuration. We proceeded by trial and error to familiarize with the structure and also started experimenting on fine tuning. The best accuracies were found on completely trainable or only half freezed networks. After finding some stable densenet architecture and preprocessing parameters we proceeded to test our final draft which instantly gave us better results.

EVOLUTION	VAL_ACCURACY
Base Model	0.63
Fine Tuning (8 frozen) + Dense 512	0.81
Dropout 0.3	0.87
Fine Tuning (16 frozen)	0.93

3.2 DENSENET

At the beginning DenseNet appeared to be the most promising model.

Using fine tuning we found our final model by freezing 95 layers, then we edited our classifier a bit in order to add some complexity.

We started by putting a Dense Layer before the softmax and we found out that choosing 2 layers of 128 units was the best choice.

To regularize the model, we put after each Dense Layer a Batch Normalization Layer.

CLASSIFIER UNITS	VAL_ACCURACY
64	0.9130
128	0.9149
256	0.9006
512	0.9018
128 + 128	0.9481

Since the model was prone to overfitting, we tried to insert a Dropout Layer between the two Dense layers, and the testing accuracy increased with 0.2 dropout:

DROP RATE	VAL_ACCURACY
0.1	0.9193
0.2	0.9481
0.3	0.9105

Our final try was to change the activations of the classifier and we reached our highest score with ELU:

ACTIVATION	VAL_ACCURACY
ReLu	0.9481
LeakyReLu	0.9067
ELU	0.9204

3.3 XCEPTION

Xception is a powerful model capable of reaching great accuracy on the training set in few epochs.
Its growth rate makes overfitting an issue we managed to contain through a number of different attempts and structural combinations.
We initially added another 128-unit Dense layer and a Dropout with 0.3 rate, significantly improving the performance.
Another Dropout layer with 0.1 rate and the reduction of units to 256 resulted in an even better outcome, combined with the fine tuning of most of the layers.
The final Xception model was built using a Flatten layer combined with a 256-unit Dense and two 0.1 Dropout layers. After fine tuning, this net managed to surpass a val_accuracy of 0.96.

EVOLUTION	VAL_ACCURACY
Base Model	0.5439
Dense 128 + First Dropout 0.3	0.8191
Dense 256 + Second Dropout 0.1	0.8573
First Dropout 0.1	0.9102
Fine Tuning	0.9645

4. FINAL SUBMISSION MODEL

The model we decided to submit is based on the VGG16 network: even if in the local testing we have been able to obtain better performances from other models, this one has outperformed the others considering the test set of the challenge.

Last step, we used keras tuning with bayesian optimization to explore different hyperparameter configurations and find the optimal values.
The final model submitted is composed of:

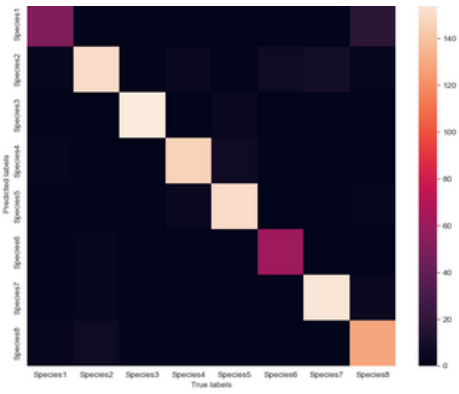
- batch size = 32
- fine tuned layers = 16
- loss = Categorical Crossentropy
- optimizer = Adam(5e-5)

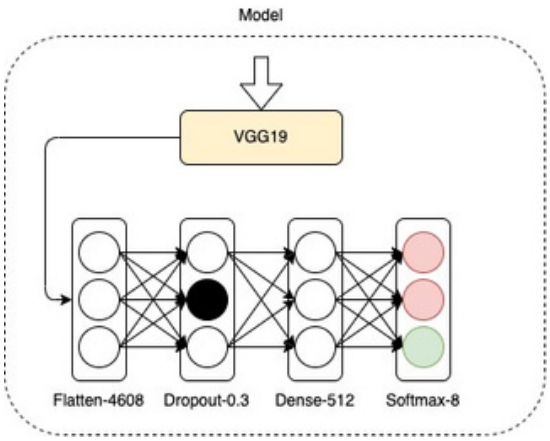
Layer (type)	Output Shape	Param #
Input	[(None, 96, 96, 3)]	0
VGG16	(None, 3, 3, 512)	14714688
Flatten	(None, 4608)	0
Dropout	(None, 4608)	0
Dense	(None, 512)	2359808
Softmax	(None, 8)	4104

Total params: 17,078,600
Trainable params: 17, 078, 600
Non-trainable params: 0

With a 80-20 dataset split, the model, evaluated with Early Stopping with patience 10 on validation loss,, reached the best weight at epoch 45.
Then we computed the confusion matrix, using a 60-20-20 dataset split, with the following testing performance:

Accuracy: 0.9308
Precision: 0.9206
Recall: 0.9262
F1: 0.9217





Development Phase Accuracy: 84.84%
Final Phase Accuracy: 85,21%