Artificial Neural Networks and Deep Learning - Academic year 2022-2023
Professor Matteo Matteucci

Authors:
Antonio Sgaramella (10665339)
Dario Vernola (10629354)
Francesco Vittorini (10864252)

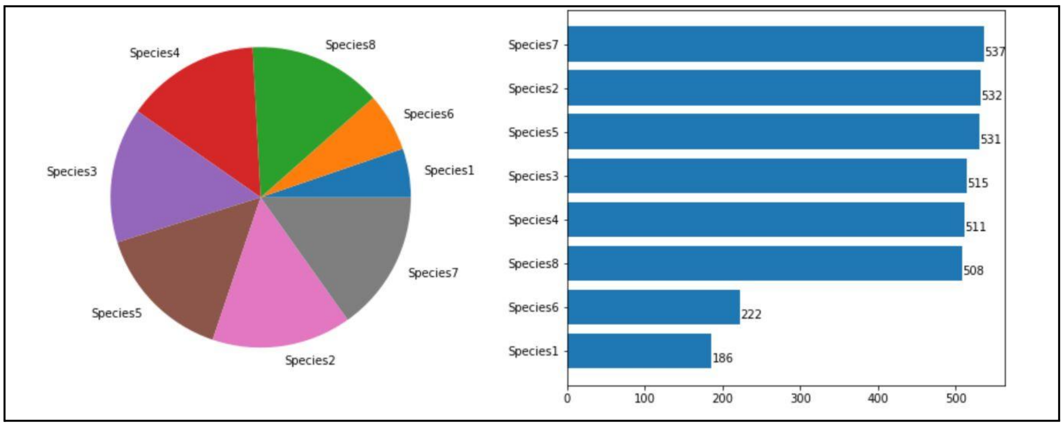
HOMEWORK 1 REPORT

1- Introduction

The goal of the first Homework was to design a Convolutional Neural Network capable of classifying images of plants belonging to eight different species starting from a given dataset of labeled images.
Scope of this document is to show the experiments and the process that lead to the final solution, analyzing the choices that have been made during the development phase.
Firstly, we will start by describing the provided dataset and the subsequent preprocessing phase.
Then we will delve into the various steps we followed to find the best suited model for this specific classification problem.
Finally, the obtained performance results will be displayed.

2- Dataset analysis and preprocessing

The provided dataset was a collection of 3542 96x96 RGB images split into 8 folders, one per plant species.
Such data is distributed in the following way:



By analyzing the latter graphs a strong class imbalance can be clearly noticed.
In particular, the "*species 1*" and "*species 6*" are undersampled with respect to the remaining species.
To solve this problem, in the training phase we decided to assign different weights to different species according to the available number of samples for each species, using the following formula:

$$w_i = \frac{n\_samples}{(n\_classes * n\_samples_i)}$$

By using these weights misclassification errors performed over undersampled species are penalized more than others.

The dataset has been split into two different subsets: training set and validation set, the former used to train the network to compute the weights that minimize the loss function, while the latter represents images never seen by the network during training and used to assess the generalization capabilities of the different models and perform model selection and hyperparameters tuning.
The split was performed using the splitfolders class, choosing as split rate 0.2 (80% training and 20% validation) in a stratified manner, that is, by maintaining the classes distribution among the two sets.
In order to train the model with a greater amount of data than the one provided, data augmentation techniques were used on the training set using the ImageDataGenerator class from the Keras library. This class provides multiple choices for image transformation (e.g. horizontal and vertical flip, brightness etc…) . We performed multiple trainings by selecting different combinations of the provided transformations and came up with interesting results.
In particular, the best performance was obtained by using the following mix of transformations: *height_shift_range*, *width_shift_range*, *horizontal_flip*, *vertical_flip*, *fill_mode='reflect'*.

3- Earlier experiments and simple models trained from scratch

The design of the model was approached gradually, starting from a very simple one, presenting very few layers, and making it gradually more complex in order to increase the capabilities of the model to extract the relevant features but being careful not to make it too complex to avoid overfitting.
This process led us to realize a CNN composed of 5 convolutional layers with 3x3 filters interleaved by batch normalization, reLU and 2x2 average pooling layers (which proved to be more efficient with respect to max pooling ones) on top of a flattening layer, a 512-neuron classifier layer followed by an 8-neuron output layer. This particular layer stack was inspired by an architecture used to solve a problem similar to ours [1].
This model alone, trained from scratch, reached an accuracy of about 85% on the hidden test set.
During the training of this model we noticed that some image transformations introduced in the preprocessing such as *zoom_range* and *rotation_range* were underperforming and for this reason we removed them, obtaining far better results also in the following models. We also noticed that by removing the dropout layer initially introduced the model was still not overfitting and performed better on the validation set. Further modifications to this model did not provide any significant improvement and for this reason we decided to exploit pre-trained models using transfer learning and fine tuning techniques.

4- Transfer learning and fine tuning

We did some research to discover which pre-trained network was used for classification problems similar to ours and noticed that the *DenseNet121* network was used in some plant-based classification problems so we decided to use it as our reference network for transfer learning.
This was later confirmed by our own experiments which showed that other networks performed worse than *DenseNet121*. The only network with comparable results was VGG16.

Preprocessing was done on the input data using the same preprocessing function used during the training of *DenseNet121* which is provided by the Keras application library.
The network was then imported excluding the top classifier which we substituted with our own implementation, using a GlobalAveragePooling2D layer and BatchNormalization layer followed by a 256-neuron Dense layer as classifier connected to a 8-neuron output Dense layer.

[1] https://doi.org/10.1016/j.biosystemseng.2016.08.024

To exploit the feature extraction capability of the DenseNet network we firstly trained only the top part of the network, freezing all the remaining layers (the DenseNet ones). This part of the training was performed using a batch size equal to 8 and a number of epochs equal to 300 using early stopping with patience equal to 20 epochs and default learning rate of the Adam algorithm.

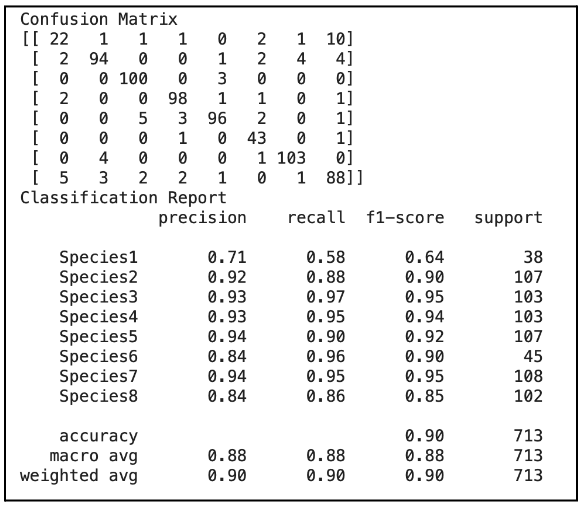
This phase brought us to a validation accuracy of about 82%, we then decided to use fine tuning to further improve such result by setting the weights of DenseNet to trainable to adapt them to our specific problem. In order to maintain the feature extraction capabilities of DenseNet and not to modify excessively the pre-trained weights, though, we decided to greatly lower the learning rate to 1e-5, a value which we later modified even further to 1e-6 in the final phases of the training.

With this procedure we managed to reach a 90% accuracy on the validation set and 87% on the development phase hidden test set. This performance was never topped by any other experiment made with other types of keras applications.
Many experiments were carried out by modifying the top part of the network (e.g. changing number of neurons, using flattening instead of GlobalAveragePooling2D etc..) but without better results.

This model was the one considered as the best one by our group.
Further improvements were reached by changing the split rate of the dataset between training and validation (in order to increase the number of training samples). In particular, we switched to a ratio of 0.1, hence 90% data for training and 10% for validation.
This approach led to a score of 88.57% on the hidden test set.

Finally, to maximize the score on the test set, we trained our network on the entire dataset provided, which brought us to a score of 89.84% on the hidden test set and 88.17% on the final phase hidden test set.

<u>5- Model evaluation</u>



```
Confusion Matrix
[[ 22   1   1   1   0   2   1  10]
 [  2  94   0   0   1   2   4   4]
 [  0   0 100   0   3   0   0   0]
 [  2   0   0  98   1   1   0   1]
 [  0   0   5   3  96   2   0   1]
 [  0   0   0   1   0  43   0   1]
 [  0   4   0   0   0   1 103   0]
 [  5   3   2   2   1   0   1  88]]
Classification Report
              precision    recall  f1-score   support

    Species1       0.71      0.58      0.64        38
    Species2       0.92      0.88      0.90       107
    Species3       0.93      0.97      0.95       103
    Species4       0.93      0.95      0.94       103
    Species5       0.94      0.90      0.92       107
    Species6       0.84      0.96      0.90        45
    Species7       0.94      0.95      0.95       108
    Species8       0.84      0.86      0.85       102

    accuracy                           0.90       713
   macro avg       0.88      0.88      0.88       713
weighted avg       0.90      0.90      0.90       713
```

Model evaluation was carried out through the use of confusion matrix and classification metrics (precision, recall and F1 score).
The figure on the left represents the performance of our best model trained over a training set characterized by 80% of the dataset and after performing fine tuning.
The metrics are evaluated over the performance of the model with respect to our validation set (the remaining 20% of the given dataset).

[1] https://doi.org/10.1016/j.biosystemseng.2016.08.024