Stefano Fumagalli 10628587
Lara Ferro 10622035
Alberto Noris 10634510

Artificial Neural Networks and Deep Learning

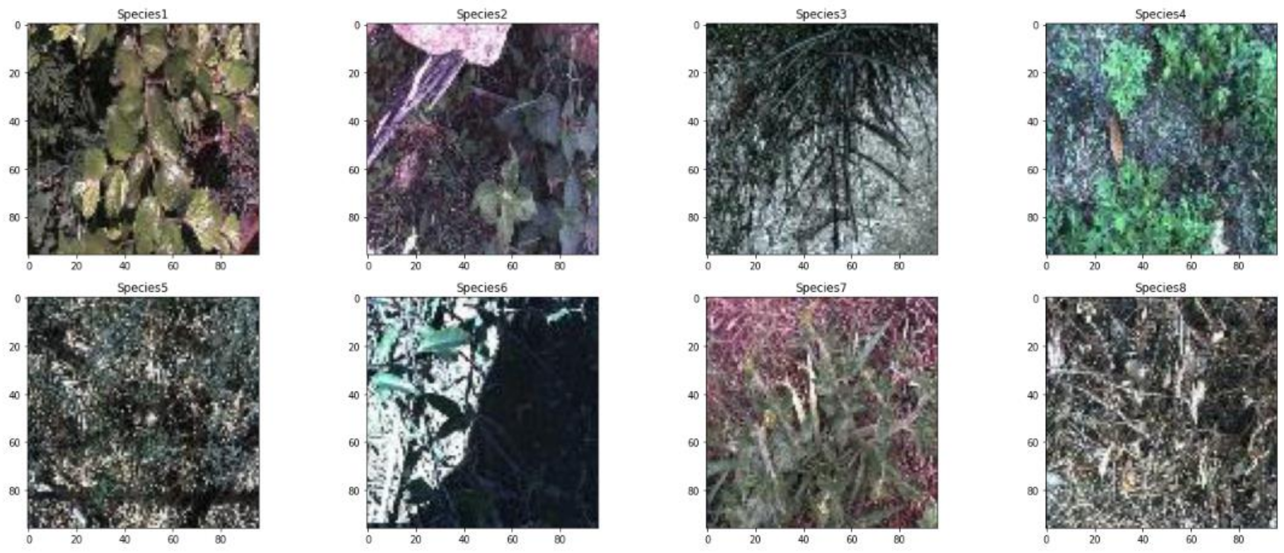
Image Classification Problem

For this challenge we had to correctly classify images belonging to eight different species of plants, provided us in a zip folder already divided in subdirectories, one for each kind of plant. We had at disposal 3542 images with size 96x96. Moreover, the training images per class were not balanced.

Facing this problem, in order to predict the correct class label, our team decided to organize the flow of work in this manner: to develop a CNN from scratch starting from the very basics and gradually proceeding towards more advanced models, applying Transfer Learning and Fine-Tuning techniques.

In the end, we had three different solutions for the problem:

- Convolutional Neural Network from scratch
- Model exploiting Transfer Learning technique
- Model exploiting Fine-Tuning technique



Samples of images from the 8 classes

Custom Convolutional Neural Network

To start with a custom CNN, we split the given dataset in two parts: 90% for the training set and 10% for the validation set to perform cross validation, early stopping and prevent overfitting [source]. Then, we created a simple CNN with few layers, 3 convolution and 3 max pooling followed by a flattening and a dense layer; since the dataset had few images and we wanted to avoid overfitting we also decided to add dropout layers [source]. Last but not least we preprocessed our training set dividing each value by 255 thus normalizing the images, and in the end trained the model obtaining a result of 26% on the test set.

Starting from this initial net, in order to improve the performance, we decided to add few more blocks composed by a convolutional and a max pooling layer, till we obtained a result of 60% on the validation set.

To increase the number of images we took inspiration from the laboratory notebook and added data augmentation. We started adding one transformation at a time, such as shift, rotation, zoom and brightness, finding out that some techniques worked well (like shift and brightness) while others not so much (like rotation and zoom). Moreover, we also tried to gradually combine them to find the best combination to use in our model.

In addition, we inserted a Global Average Pooling layer and tried regularization with L1, L2 and L1L2 with different values of lambda (0.02, 0.002, 0.00002), reaching our best result yet with L1L2 and with lambda equals to 0.002.

We thought that one of the possible reasons for which the model struggled to learn was the small dimension of the images, so in order to fix this we tried to add a resizing layer after the input. This layer doubled the size of the images and increased the performance on our validation set but unfortunately didn't work as well for the test set [source].

As stated before, one of the other problems we identified was the unbalanced size in the classes of the dataset, for class1 and class6. We believed that using class weights would have solved this issue, but even though, it increased the performance on the classes with few images it decreased it for the classes with more elements, not improving the total accuracy with respect to the model without class weights.

In the pursuit of optimizing the model, we tried to change the activation function and obtained our best new result with the ReLu function instead of the Leaky ReLu.

To verify that the small number of images were no longer a problem we applied K-fold to estimate the skill of our model on new data [source]. We were satisfied with our results, and we submitted our final model obtaining an accuracy of 66%.

Model with Transfer Learning

Following our path in the evolution of deep learning, as seen in the laboratories, we moved on to Transfer Learning to exploit pretrained nets. We started with a simple test using Vgg16 and its preprocessing function, excluding its top-net and inserting ours defined in the previous section. After that, by playing with hyper parameters like learning rate and number of neurons in the dense layer, we reached our best new result with an accuracy of 72%.

Since we noticed that after several attempts the accuracy didn't improve as expected, we started asking ourselves about the possible causes and after searching for answers we discovered that one possible reason was related to the high number of parameters of Vgg16 with respect to the few numbers of images of in training dataset [source]. For this, we decided to use a pretrained model with less parameters with respect to Vgg16, like Xception, DenseNet, EfficientNetB0, Inception and ResNet50: we compared the accuracy of these models, and after several attempts of hyperparameters tuning, we selected ResNet50 as the model to be used as it raised our accuracy to 76%.

To further increase the performance, we tried different rates for the dropout layers (which we increased to 0.35 to prevent overfitting), lowered the learning rate [source] added a scheduler exponential decay to reduce the learning rate every few epochs [source] and changing the optimizer from Adam to SGD, without obtaining the desired improvements. In the end, after all these attempts, our accuracy on the test set was 80%.

Model with Fine-Tuning

Using the same model as before, we decided to apply fine tuning by freezing some layers and trained the others with our training set. We found out that our model couldn't reach an accuracy higher than 83% and by analyzing the training data we thought it could have been due to overfitting during fine-tuning.

To solve this problem, we decided to decrease the number of parameters, so we switched to EfficientNetB2 and we repeated the steps made in the CNN to avoid overfitting: we increased the dropout rate, we added class weights to have a better performance on classes 1 and 6 and we inserted data augmentation, with few techniques like shifts and fill (the reflected option gave our best outcome). We also discovered that the highest accuracy was reached by unfreezing all the layers and training them with a much lower learning rate. With this new model we were able to achieve an accuracy of 89% on the test set.

Looking at the confusion matrix, we found out that one of the main causes of error was the class8 identified as class1. So, in order to solve this issue we tried to increase the weight of these classes , but it didn't work well and so we came back to the original configuration.

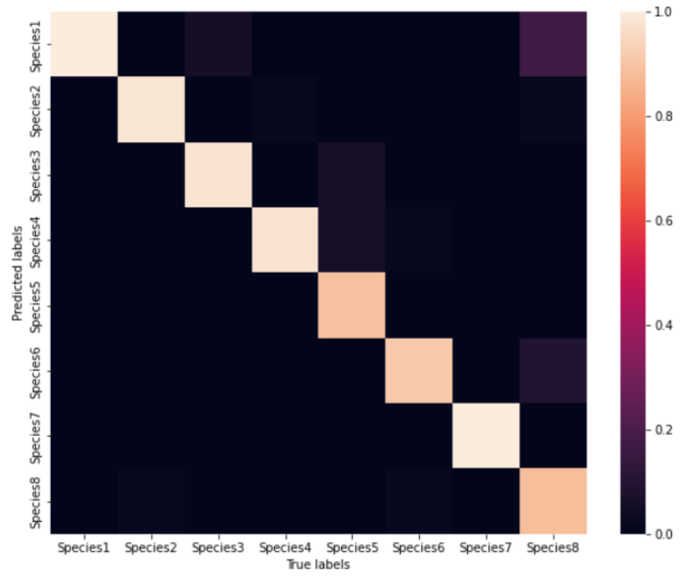
In the end, we tried other augmentation techniques like rotation or zoom but they didn't work well. Instead, we obtained an increase of 2% by implementing cutout on the training set and another increase of 1% by changing the brightness of the images.

To do that, we changed a little bit the shape of our model: we implemented a function that applies cutout to the single image, and we passed it as a preprocessing function to the ImageDataGenerator object. Then, in order to preprocess the data, we added a new layer between the input and the resizing ones, defined as a Lambda function that calls the preprocessing function of EfficientNetB2.

We also tried to implement cutmix function, but we discovered that this transformation didn't bring any improvements, making the prediction less accurate, so we removed it.

Finally, we ended up with a model giving us an accuracy of 90,56% on the test set of the first phase and an accuracy of 90,14% in the second phase, with the following confusion matrix referred to the validation set:

```
Accuracy: 0.9516
Precision: 0.9395
Recall: 0.9522
F1: 0.944
```



Confusion matrix of the best model

```
              precision    recall  f1-score   support

           0       0.82      1.00      0.90        18
           1       0.96      0.98      0.97        53
           2       0.94      0.98      0.96        51
           3       0.93      0.98      0.95        51
           4       1.00      0.89      0.94        53
           5       0.91      0.91      0.91        22
           6       1.00      1.00      1.00        53
           7       0.96      0.88      0.92        50
```

Performances of the best model