ANNDL Homework 1 - Image Classification

Gradient Climbers
Federico Caspani, Andrea Cerasani, Matteo Citterio

November 28, 2022

1 Introduction

This report describes our work regarding the challenge 1 of the ANNDL course. The goal of this challenge is to model a neural network able to classify images of different species of plants based on a given dataset. The following chapters will reflect our workflow, analyzing and describing our choices.

2 Networks from scratch

Our first approach of modeling this problem was to use non pre-trained networks, made from scratch.

2.1 Data pre-processing

At first the dataset was split into train, validation and test set using a 70-15-15 ratio. The validation was performed using *Hold-out method*. To cope with the scarcity of images we chose to perform the following data-augmentation:

```
train_data_gen = ImageDataGenerator(
                                    rotation_range=15, zoom_range=0.2,
                                    height_shift_range=0.2, width_shift_range=0.2,
                                    horizontal_flip=True, vertical_flip=True,
                                    fill_mode='reflect'
                                )
```

We chose 'reflect' in fill_mode to avoid disrupting the network by filling the images with insignificant information.

2.2 CNN

As shown in figure 1, our first try consisted in a classic CNN architecture, composed by 4 convolutional blocks respectively containing 16, 32, 64, 128 filters of size 3x3. In the classifier part of the network we used 3 dense layers alternated with dropout as a form of regularization. The training was performed using early stopping with patience set at 10.

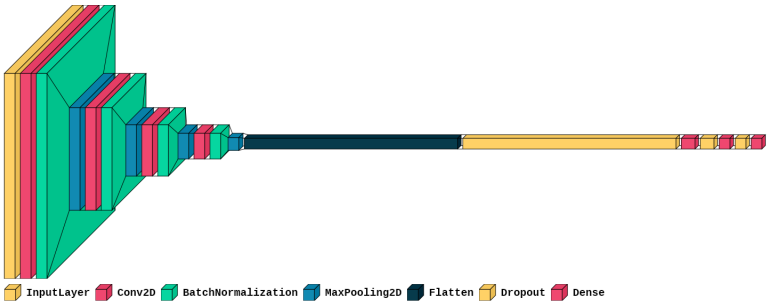

Figure 1: CNN architecture

2.3 ResNet-50

As a second attempt, we implemented a ResNet-50 from scratch taking inspiration from online resources. The core purpose of using this network is to enable us to build deeper networks thanks to the Skip Connection technique. Compared to the original ResNet, the shortcut connections in ResNet-50 skip 3 layers instead of 2. We chose this network because stacking more layers would lead to better results while avoiding the vanishing gradient problem.

2.4 Results

As expected ResNet-50 performed better than CNN (table 1) even though the results were still not satisfactory. With the prospect of better performance we chose to apply some changes to the way we used our data, which will be explained in chapter 3, and to move to the Transfer Learning approach, which will be discussed in chapter 4.

Network	Epochs	Test Accuracy
CNN	183	0.6851
ResNet-50	250	0.7396

Table 1: Networks from scratch results

3 Data scarcity and data quality

We identified the low number of images in our training set as the main factor of our low performances. To deal with this we adopted some countermeasures:

- We increased the size of our **training set** changing the ratio from 70-15-15 to 80-10-10 for train, validation and test sets.
- We tried different configurations of **data augmentation** until we reached our best result by increasing the rotation range up to 360° (since photos were taken from above) and adding contrast with the purpose of having a more varied dataset.

```
img_augmentation = Sequential([
        tfk.layers.RandomRotation(factor=1),
        tfk.layers.RandomZoom(width_factor=0.2, height_factor=0.2),
        tfk.layers.RandomTranslation(height_factor=0.2, width_factor=0.2),
        tfk.layers.RandomFlip(),
        tfk.layers.RandomContrast(factor=0.3),
])
```

- In particular, to deal with the low number of samples in species 1, we changed the **class_weight** parameter in our model.fit() to make the model pay more attention to under-represented classes.

Figure 2: Low quality images

Another measure we adopted to improve the quality was to clean our dataset by deleting images with misleading elements such as unrelated objects or strong and predominant shadows as shown in figure 2.

4 Transfer Learning networks

Beyond the measures adopted above to improve the quality of our training data, the most important technique we adopted to improve the accuracy of our model is **Transfer Learning**. This allows us to exploit the knowledge of networks pretrained on similar problems and considering the low number of images in our training set the effect of this approach is even more important.

This method consists in wrapping a network initialized with pretrained weights and deprived of its top layers to which we substitute our top layers to better adapt the network to our problem.

The input of the pipeline is first resized to match the input size of the pretrained network and then augmented.

The purpose of the pretrained network is to perform feature-extraction out of the augmented images such that our classifier network at the top is able to learn how to correctly classify with respect to the extracted image features. The whole architecture is shown in figure 3.

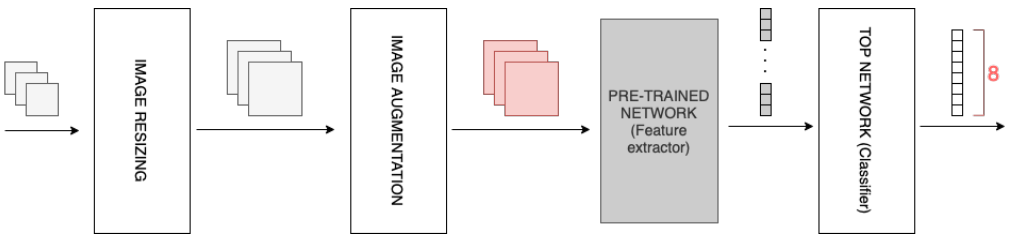

Figure 3: Our Transfer Learning network architecture

The learning process is divided in two phases:

1. We first freeze the layers of the pretrained network training only our classifier.
2. We then perform **fine-tuning** by lowering the learning rate and unfreezing partly or completely the layers of the pretrained network.

				TRANSFER LEARNING		FINE TUNING		
Pre-trained model	**Top network**	**Clean Dataset**	**Class balancing**	**Epochs**	**Val_accuracy**	**Epochs**	**Val_accuracy**	**Test accuracy**
ResNet50	Flatten, Dense (256,128,64)	N	N	70	**0,74**	40	**0,79**	**0,7811**
EfficientNetB7	GAP, Dense (32)	N	N	45	**0,7**	46	**0,82**	**0,8222**
EfficientNetv2B3	GAP, Dense (256)	N	N	100	**0,74**	200	**0,84**	**0,8396**
EfficientNetB7	GAP, Dense(1024)	Y	Y	100	**0,785**	95	**0,86**	**0,865**
ConvNeXtBase	GAP, Dense(64,32,16)	Y	Y	32	**0,81**	22	**0,85**	**0,84**
MobileNetV2	GAP, Dense(128,64,32)	N	Y	100	**0,76**	200	**0,88**	**0,86**
ConvNeXtLarge	GAP, Dense(1024)	N	N	30	**0,78**	30	**0,88**	**0,87**
ConvNeXtBase	GAP, Dense(64,32,16)	N	Y	95	**0,78**	200	**0,91**	**0,9**
EfficientNetV2B0	GAP, Dense (256)	N	N	16	**0,81**	50	**0,91**	**0,9037**
ConvNeXtXLarge	GAP, Dense(512, 256, 128)	Y	Y	60	**0,85**	40	**0,9459**	**0,9475**

Figure 4: Our attempts

We tried this technique using various networks and we experimented several configurations by modifying our top classifier and choosing a different number of layers to unfreeze, different augmentations and by training our model with either the original dataset or the ”cleaned” one. Our main attempts are summarized in figure 4.

4.1 ConvNeXtXLarge

As the schema shows, we obtained our best result using a **ConvNeXtXLarge** for extracting features, a particular network that enhances the traditional ResNet architecture using hierarchical vision Transformers and at the moment it is one of the best networks in terms of classification performance, even though with its 350.1M parameters it results to be heavy to train.

After various attempts with this networks, the configuration that led us to the best result was as follows:

- Our input image is resized to (224, 224) to match the ConvNeXt default input size.
- The ConvNeXt block is preinitialized with *imagenet* weights.
- The ConvNeXt layers are kept frozen during the Transfer Learning phase and all of them are unfrozen during fine-tuning, as this was found out to give the most noticeable accuracy boost in our specific problem.
- We started trying shallower top networks, e.g. one dense layer with 1024 neurons, but we obtained the best accuracy with deeper networks and our final configuration was composed of 3 dense layers respectively with 512, 256 and 128 neurons.
- We used Adam as optimization algorithm with a learning rate of 1e-3 in the Transfer Learning part and lowering to 1e-4 while fine tuning.

4.1.1 Results

The configuration allowed us to improve our accuracy compared to other tries, and we were able to obtain 0.889 at the end of Phase 1.

Given the poor performance in the F1 score of Species 1 (0.7159), as our last attempt in the Final Phase we tried to rebalance the class weights even more in favour of species 1 and by tweaking the augmentation rotation range and contrast and we were able to reach an accuracy of 0.9211 with an F1 score of species 1 of 0.8516 (4th best). The entire learning procedure and the obtained results are shown in figures 5 and 6.

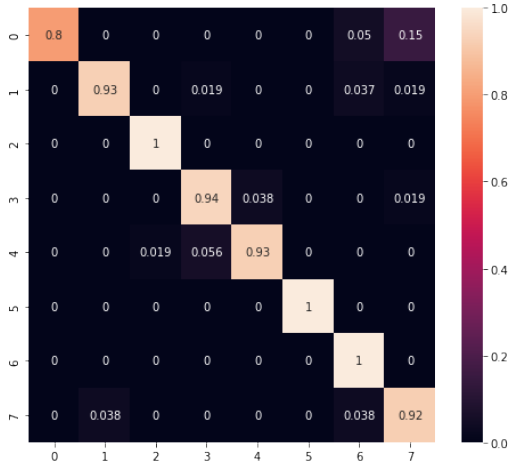

Figure 5: Confusion matrix

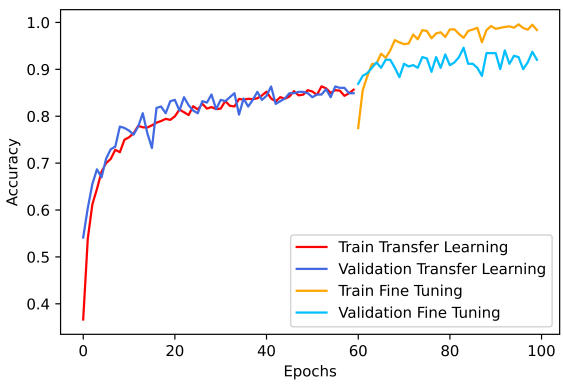

Figure 6: Training of our best model