ANNDL Assignment 1 - Alessia Cotroneo, Alessandro Del Vitto, Michele Di Sabato

Introduction

In this report we describe the steps we followed and the issues we encountered while conducting experiments for the first competition of the course Artificial Neural Networks and Deep Learning. At first, we tried to implement our own **CNN architecture**, designing both the feature extractor and the fully connected layers. Then, we used **Fine Tuning** and we observed an increase in the accuracy in both the training and test sets. We **conclude** with the description of our best model.

1. CNN Architecture

1.1. Feature Extraction part

At first, we implemented our own feature extraction layers, as shown during the lab sessions and lectures. The most challenging part was leveraging the right number of convolutional layers and the right number of filters for each one of them. According to our results, if the switch from the feature extraction part and the top layers of the CNN is too quick, then not only the computational cost is higher (since the flattening layer produces longer vectors to give as input to the fully connected layers), but also the performance of the Neural Network is undermined. Our choice was to use a number of convolutional layers, each followed by a MaxPooling layer to gradually decrease the width and height of the input. We kept the first layers with size similar to the one of the input image, to allow the neural network to first learn the specific details of the leaves given as input, and then the high level patterns (as seen during the lectures). In particular, we used: stride = (1,1), filter size = $(3, 3)$, $ReLu$ activation function.

1.2. Fully Connected part

Relying on the universal approximation property for the classification task, we used two dense layers ($ReLu$ activation function) in addition to the final output one ($softmax$ activation function, as suggested for classification purposes). We chose to adopt a decreasing number of neurons, trying not to have in the end too many parameters to train. In the CNN model with 62% of accuracy we used 128 for the first dense layer and 64 for the second one, for a total of 2,695,134 parameters. In order to avoid overfitting each dense layer was followed by a dropout layer with "switch off" parameter equal to 0.4, we implemented the Early Stopping technique and we included a regularization term in the loss function.

Preprocessing

We noticed that most of the leaves in the training set were vertically oriented so we opted for the use of Data Augmentation in order to help the Network learn the actual features of the leaves and not be biased by the dataset images. The results improved by a few percentage points. We then tried rescaling each pixel to the range (0,1): this lead to a relevant increase in the accuracy of our network both in the training and the test sets. This is reasonable since, as seen during the lectures, the training phase works best with small weights.

For more coding details on this model, see the file **OurCNN.ipynb**.

2. Fine Tuning

The following checkpoints represent the main progresses of the second part of our project, in which we explored pure transfer learning and partial or total fine tuning:

1. We firstly attached the VGG16 architecture (freezing all its weights) to two dense layers (each followed by a dropout) and the output layer. This model did not reach a satisfying level of accuracy in the test set, so we moved to fine tuning.

2. We used fine tuning on the last 5 layers of VGG16 and this resulted in a higher accuracy of the model on the training, validation and test set.

3. After plotting the structure of the model we realised that the switch from the feature extractor and the dense layers might have been too quick, so we added after the fine-tuned VGG16 architecture one convolutional layer in between two maxpooling layers and this resulted in the highest accuracy out of all of our attempts (**see FineTuningVGG16-FINAL-MODEL.ipynb for reference**).

4. Then, motivated by the higher accuracy reached when we added additional layers before the fully connected part, we developed a new model, composed of two convolutional layers added after the VGG16 structure, each one followed by a max-pooling layer. In order to improve the accuracy score, we decided to both attempt to partially fine tune the VGG16 layers, and then also to fully tune the entire feature extraction layers of the VGG16 model. The first attempt resulted in an 8% decrease on the accuracy in the test set with respect to the previous attempt of point 3. On the other hand, the second attempt was not satisfying, since the accuracy reached by this configuration was not proportionate to the (much) higher computational cost (**see TransferLearning-FT-vgg16.ipynb for more coding details**).

5. Finally, we tried to tackle the problem using another pre-trained architecture, namely InceptionV3. The accuracy generated by this last attempt was much lower than our best score (**see TransferLearning-InceptionV3.ipynb for reference**).

For each step in our procedure, we performed both rescaling (i.e. we divided each pixel by 255) and preprocessed the data with the preprocessing function appropriate to the architecture used.

3. Conclusions

In conclusion, the highest level of accuracy in the test set has been reached when we used fine tuning on the last 5 layers of the VGG16 architecture and adding to it an additional convolution in between two maxpooling layers (subsection 2.3). We used a low batch size (8), regularisation and dropout techniques, to avoid overfitting, and a low learning rate (10^{-4}), since we found that when working with previously trained architectures it's best to keep this hyperparameter low. Moreover we added a line of code to reduce the leaning rate whenever a plateu occured.
The following figures show the accuracy, loss functions and confusion matrix of some of our models.

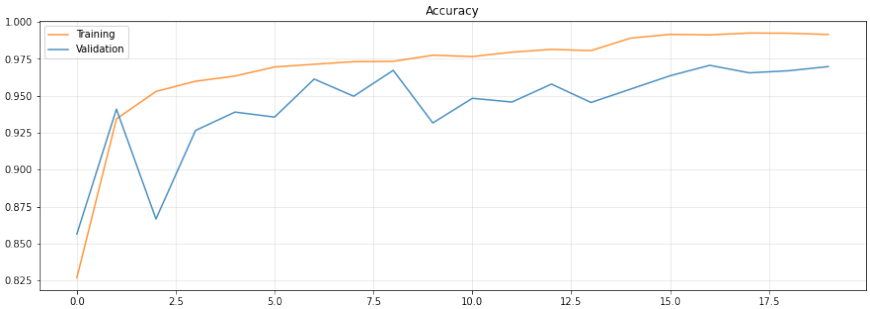

Figure 1: Accuracy of the training and validation sets for our best model (VGG16 + Conv2D + MaxPooling + Conv2D + MaxPooling + dense layers, see section 2.3)

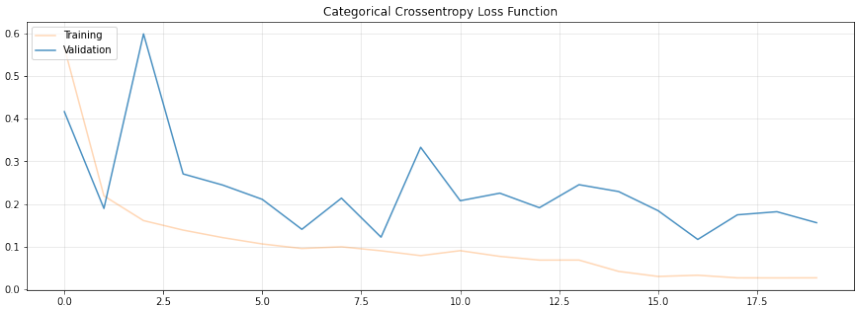

Figure 2: Loss on the training and validation sets for our best model (VGG16 + Conv2D + MaxPooling + Conv2D + MaxPooling + dense layers, see section 2.3)

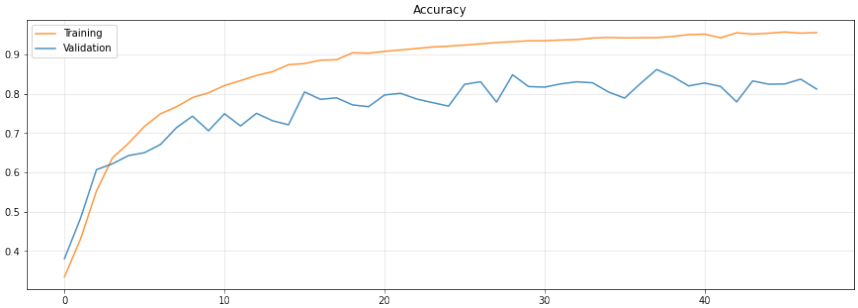

Figure 3: Accuracy of the training and validation sets for the architecture created by us (see OurCNN section 1.1 and 1.2)

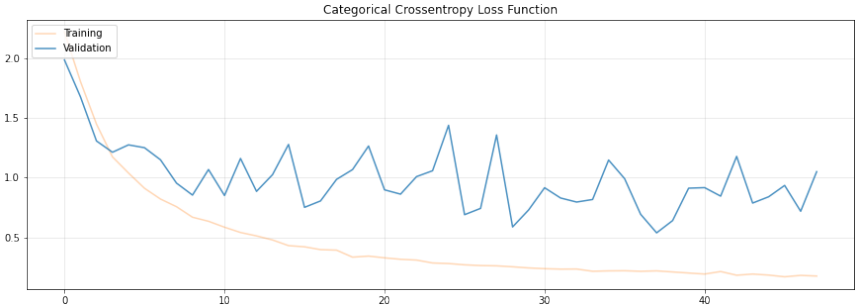

Figure 4: Loss of the training and validation sets for the architecture created by us (see OurCNN section 1.1 and 1.2)

```
Confusion Matrix
[[ 196    0    0    0    0    0    1    0    0    0    0    0    0    0]
 [   0   91    2    0    0    0    0    0    0    0    0    0    0    0]
 [   1    0  112    0    1    1    0    0    0    0    0    0    0    1]
 [   0    0    0  238    0    0    0    1    0    0    0    0    0    2]
 [   4    0    0    0  284    0    0    0    0    0    1    2    0    0]
 [   0    0    3    0    0  342    1    0    0    0    1    0    0    2]
 [   2    0    1    1    0    1  190    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0  150    0    0    1    0    0    2]
 [   1    4    2    0    1    0    0    2  112    0   19    0    0    2]
 [   0    0    0    0    0    0    0    0    0   52    0    0    0    0]
 [   0    0    0    0    0    0    0    1    0    0  322    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0  114    0    0]
 [   1    0    0    0    1    0    0    0    0    0    0    0  132    0]
 [   6    1    2    0    1    0    0    1    2    0   12    3    0 1110]]
```

Figure 5: Confusion matrix of our best model that reached 86% avg accuracy on test set