Artificial Neural Network and Deep Learning

Homework 1: Image Classification

Nicolò stagnoli, Fabio Stecchi, Andrea Bosia

November 27, 2021

First Model: Simple Convolutional Model
The first model used for solving the problem was a simple model with 5 convolutional block and each block has one convolutional layer with a 3x3 kernel and a ReLU activation function.
We put after this convolutional network a Flattening layer, Dropout layer, a Fully Connected with 256 neurons, with ReLU activation, and an output layer with softmax activation.

The training didn't go welll: it reached only 40% of training accuracy. Validation accuracy was even worse, and the loss function value was quite high.

Second Model: Transfer Learning with vgg16
We used the pretrained network vgg16, without the top part. We put after the supernet a Flattening layer, Dropout layer, a Fully Connected with 256 neurons, with ReLU activation, and an output layer with softmax activation, like before.

This model did a leap forward in the training phase, but still not satisfying. It reached about 75% of training accuracy. Loss was smaller than before.

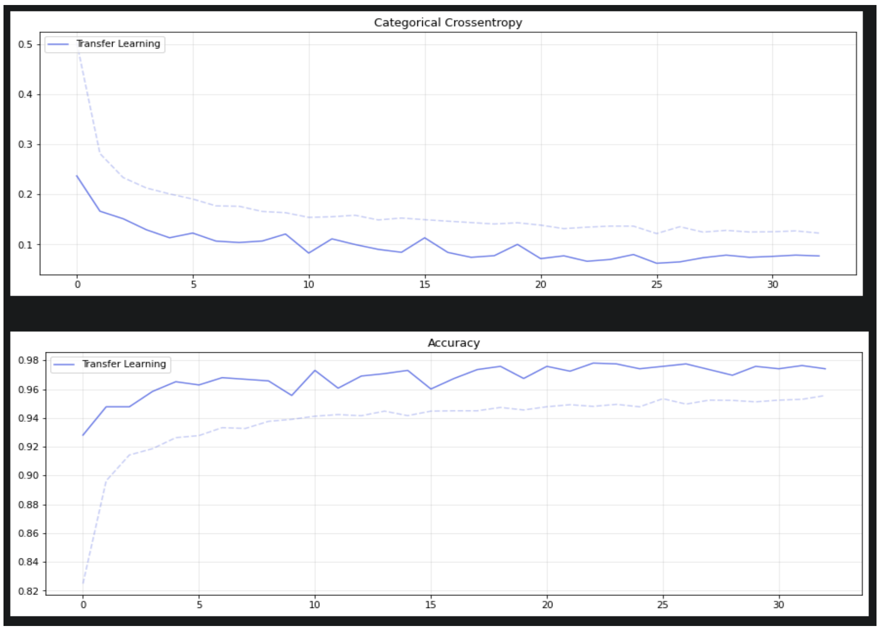
Third Model: vgg16 + class weights + Data augmentation
We used the same model as before, augmenting our dataset with random rotations, zoom, shifts and flips. We also noticed that the dataset was slightly unbalanced (tomatoes were like an order of magnitude bigger than all the other classes), so we created an ad hoc class weight vector that gave half importance to the tomato class with respect to the other classes.

All this slightly improved the training accuracy and the validation accuracy.

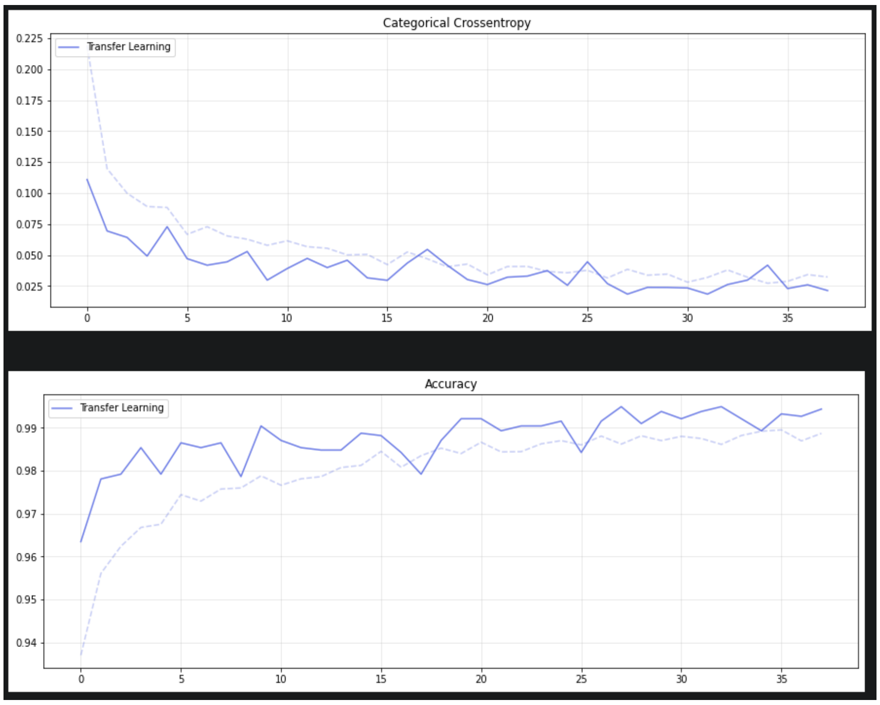
Fourth Model: Transfer Learning with InceptionResNetV2 (+ Data Aug & Class weights)
We tried Transfer Learning with this pretrained network because the name was cool.
Training and Validation Accuracy reached almost 90%

Fifht Model: Transfer Learning with EfficientNetB7 and GAP (+ Data Aug & Class weights)
In research we found that this pretrained network would have done for us. it seems to be the most efficient in the tensorflow.application package. We also added a GlobalAveragePooling layer after the supernetwork. This model Reached about 97% in Training and Validation Accuracy. The convergence was also faster than before.

Final Model: EfficientNetB7, Fine Tuning, GAP, Data Augmentation
After the satisfying results of the previous model, we trained the network as before, then unfreezed some of the last layers of the supernetwork and trained again with a lower learning rate.



First Training



Fine Tuning

Continuous line: Validation
Dashed line: Training

For all of our trials we decided to split our dataset in test(0.85) and validation(0.15).
For all our models for regularization we have used Early Stopping with patience 10 with option restore_best_wights=true in order to reload the best weights learned.