No-mask detector

Davide Brescia
Matteo Ilari
Francesco Montanaro

1. Data loading

To perform the data loading operation in the first notebook the *pandas.dataframe* module has been used.
To ensure data randomness, a first shuffle on the whole dataset has been performed.
A script has been implemented to equally distribute the different classes among training (80%) and validation (20%) data frames.
Finally, each data frame has been shuffled.

In the second notebook, due to the faster solution, the *train_test_split* method of the *sklearn* library with a splitting of 10% for the validation set and 90% of the training set has been chosen. This final splitting has been decided after multiple experiments made on these percentages.

To deal with data scarcity and obtain better performances, a data augmentation procedure has been implemented. Only some types of data augmentation have been considered useful for the task provided, as shown in the notebook.

2. Experiments

General approach

The models shown are the ones which have been considered the most promising. The final hyperparameters (shown in the notebook) have been chosen after a long process of trials and errors, briefly and qualitatively described here.
Simple architectures (low numbers of layers and neurons) have been used at the beginning. With the help of Tensorboard, after noticing that training and validation losses were quite equal and pretty high (underfitting), complexity has been added. Models started to overfit, so the techniques reported below have been used.
The learning rate was set equal to 1e-3 at the beginning of each training. After some epochs, in order to zoom for the best performances, it has been reduced down to 1e-4. It has been noticed that using a small batch size involved significant oscillations of the losses trend. The best value for the fastest convergence is between 8 and 16.

Limiting overfitting

In addition to data augmentation, few other techniques have been implemented:
Early stopping: patience value 5 / 8.
Dropout: used between each dense layer of the fully connected part of the networks.
Weight Decay: for the second model, in addition to the previous ones, regularization of some dense layers has been implemented.
Once again, the overfitting hyperparameters shown in the notebooks are the final ones. They are obtained by many tuning operations, searching for the best validation loss.

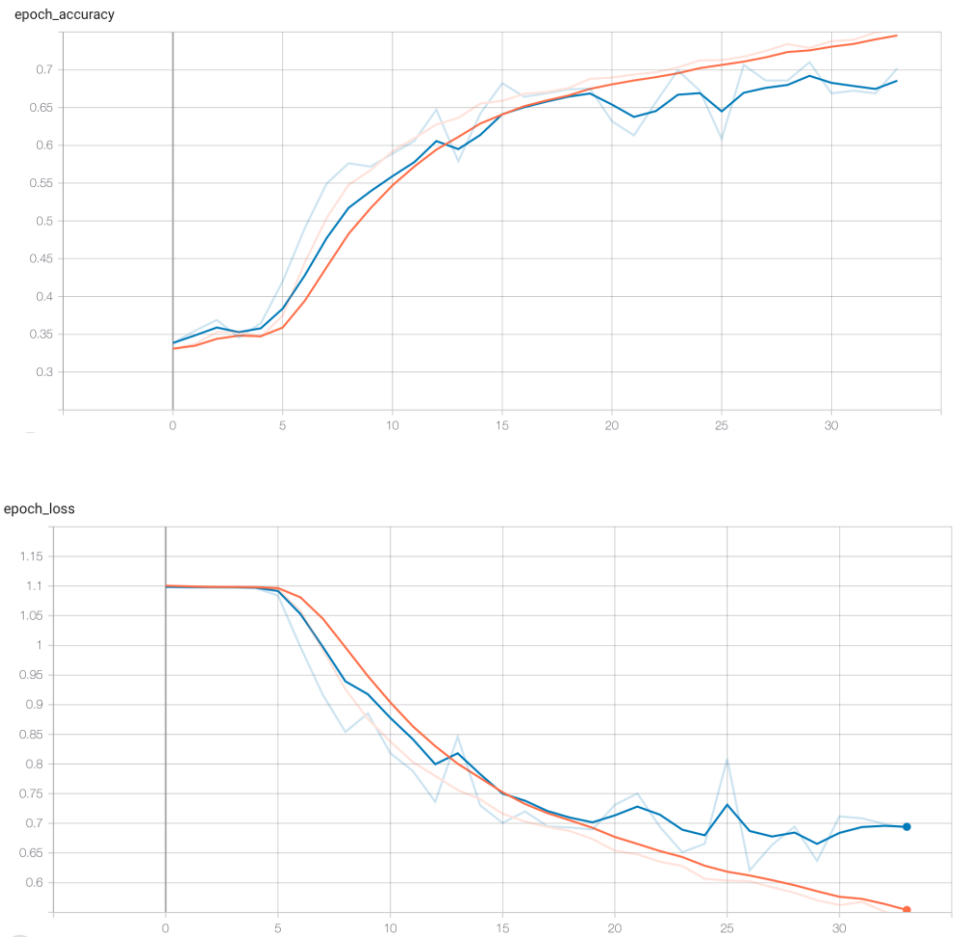
Experiment 1

In the first experiments, it has been tried to build a model from scratch.

Conv2D -> Activation (Relu) -> Maxpool (2,2) blocks have been concatenated, doubling the number of filters for each block. Depth and number of filters of the first block are free parameters. The best kernel size found was 3x3.

A cross-validation approach has been tried, to see if it could bring to better performances as shown in the *cross-validation* script; due to the poor results and higher time complexity, cross-validation has been discarded in favour of mini-batches.

After many tuning processes the best model has been reported in the notebook, and its trends are presented here.

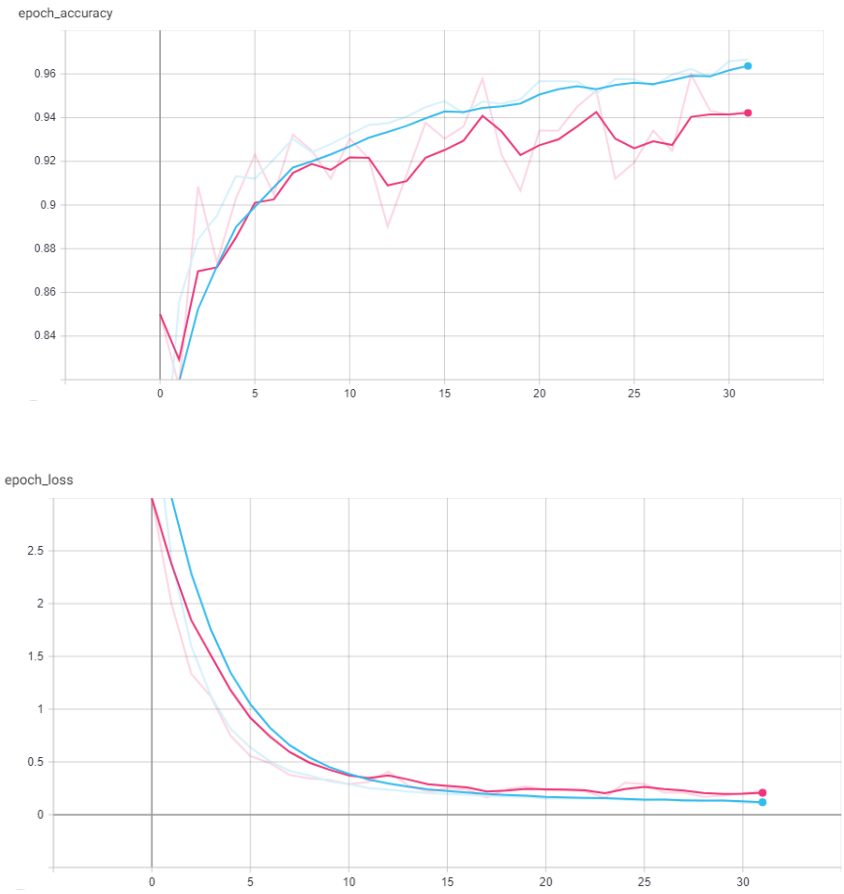


Orange: training
Blue: validation

The accuracy achieved on the test set is ***0.71111***.

Experiment 2

For what concerns the experiments made with the help of transfer learning, a first approach without neither data augmentation nor fine tuning has been performed using InceptionResNetV2 architecture. Due to the presence of a big amount of parameters, there wasn't such an improvement from the previous model, therefore fine tuning has been implemented, which brought to more than 80% of accuracy on the test set. In order to improve even more the performance by reducing overfitting, a Global Average Pooling has been added. Many models have been used like ResNet50V2, ResNet101V2, DenseNet101 and others, but ResNet152V2 brought to the best performances on the test set.



Light blue: training
Violet: validation

The accuracy achieved on the test set is ***0.93777***.