MASK OR NO MASK?

First of all, we loaded and unzipped the folder with all the data on Colab to train with GPU.

We suggest to concentrate on the second and third model (best performers), but we included also the first one to show the training from scratch procedure.

FIRST MODEL: (ClassificationFromScratch.ipynb)

We started the project building a network from scratch as we have seen in the lectures, starting with a simple one and growing in complexity in order to have a more powerful architecture. The best architecture that we have built in this way is a 1.4 million parameters CNN. Its key features are:

* Heavy data augmentation: We have few data in training set, so we used nearly every possible transformation except those that would have completely changed them (vertical flip…)
* High batch size and low image resolution: small need of computational power and, as we saw when plotting images, in 128x128 resolution important features are still recognizable
* 7 Convolutional layers with increasing number of 3x3 filters; 2x2 max poolings and Relu: standard best choices to let network simple able to learn fast
* Dropout between fully connected layers and early stopping to prevent overfitting, so it was not a big problem with this model and no regularization was needed
* Xavier initialization and batch normalization in various steps to speed up optimization. Adaptive learning rate to refine the solution

We reached an accuracy of 0.86888 in the test set. We could have done more continuing in this direction, but we decided to apply transfer learning because it was more promising.

SECOND MODEL: (TransferLearningEfficientnetB5.ipynb)

We tried different architectures, in particular the ones with the best results in increasing order are Densenet (0.90), InceptionResnetV2 (0.94) and Efficientnet (0.97). Our final choice was EfficientnetB5, best compromise between speed and accuracy. It’s near to the state of the art in classification task.

We didfine tuning since our problem is very different with respect to the one for which models were built for, imagenet. In the best network we used a complete fine tune (freeze = 0) so that weights couldbe specific to our problem. We used a global average pooling at the end to reduce the size.

Having a lot of parameters to train the risk of overfitting was high and to avoid it we used heavy data augmentation (same as first model), early stopping and we have also added a dropout layer. Overfitting then was nearly non-existent.

We had to use small batch size because of OOM errors and to be coherent with the original architecture we used the same input size used there, thinking that it could capture better features.

With this model, trained using an adaptive learning rate, we reached an accuracy of 0.96888 in the test set.

THIRD MODEL: (TransferLearningDoubleClassifierefficientnet.ipynb)

Our third approach is the most ambitious one. We thought that for a CNN for classification would be easier to solve 2 subsequent tasks that resemble more the problem of "finding a specific feature in an image".

So we have divided the model in two consecutive classifiers: “Is there someone with a mask?” and then: “Is there someone not wearing a mask?”. With the first classifier we want to distinguish the pictures of class 0 from the rest, while the second is trained to distinguish a picture of class 1 from 2.

We used EfficientnetB5 for both the two classifiers by applying the same model choice done before (complete finetune, batch and image size, etc.). With the first classifier, we achieved an astonishing validation accuracy of 1 (at peak), with the second “only” 0.97. On the test set we achieved 0.96222.

ENSEMBLE: (Ensemble.ipynb)

To produce our final results we used an ensemble of our best models. We tried many solutions (most voted class, weighted average…) but the best was averaging the probability attributed by each one of the methods to the test set images. We achieved our best results on the test set, with an accuracy of 0.97111.