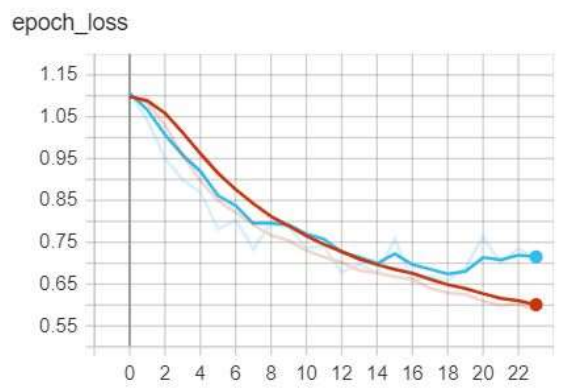
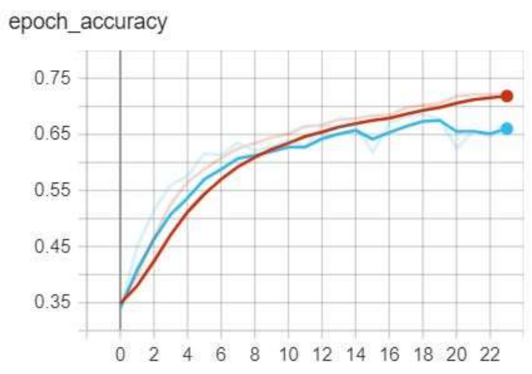
Challenge 1 Document

Note: we are aware that, by tuning and structuring our models on the performance on the test set, we are in some way biased towards it. But we think that, since this challenge was more about learning how to use keras instead of reaching the best performance, being biased towards the test set is not a problem.

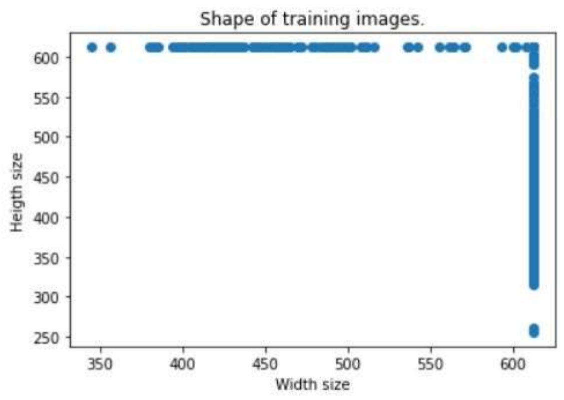
We will go through all the choices made during this challenge.

We started from the same model used during the exercise session, by resizing the images' width and height both to 256 pixels.
The initial number of filters has been left to 8, like the depth of the convolutional part left to 5.
We applied some image transformation, in order to increase the number of images in the dataset. The specific transformations applied can be seen in the notebook, in the ImageDataGenerator function call.
The performance with this network has been surprisingly good, with an accuracy of 0.7 on the test set.

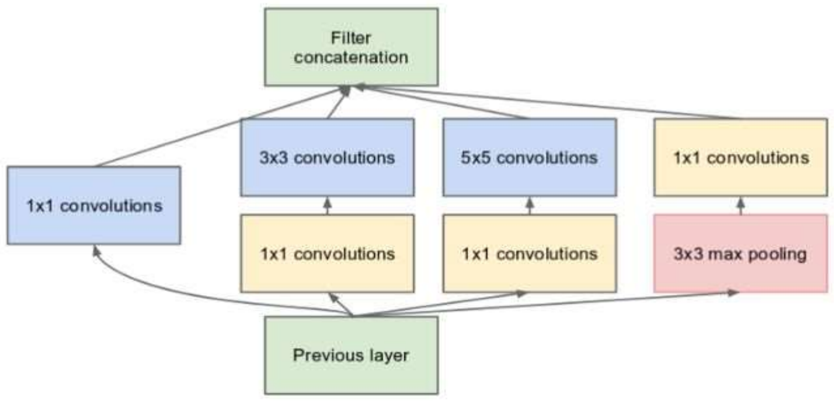




With respect to the baseline mode, we decided to start from a higher number of filters, going from 8 up to 32. Moreover, after some analysis on the dataset, we discovered that the average width and height of the images are, respectively, 512 and 384. By using those measurements, we were able to improve the performance of the model by 0.08, reaching an accuracy of 0.82 on the test set.



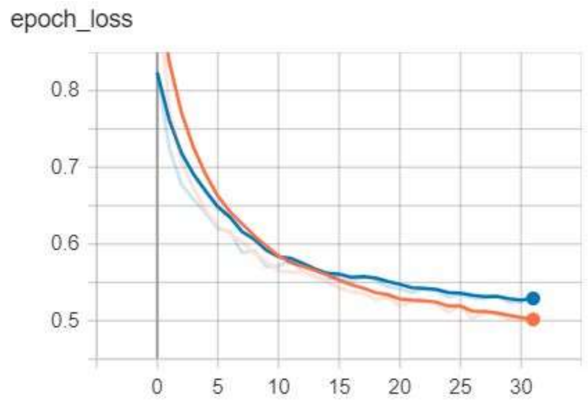
Reference notebook: InceptionModel
We noticed that, by looking at the images on the training set, the main parts of the image can have a very large variation in size (for instance, the masks and the people's faces). For this reason, we introduced inception modules in our network to have filters with multiple sizes that operate in the same level. To make it cheaper we implemented the version with dimension reductions.

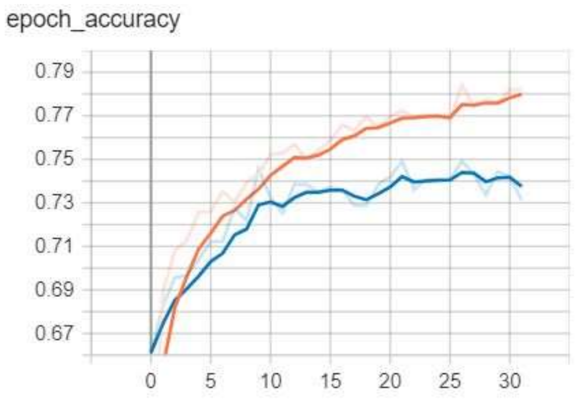


(b) Inception module with dimension reductions

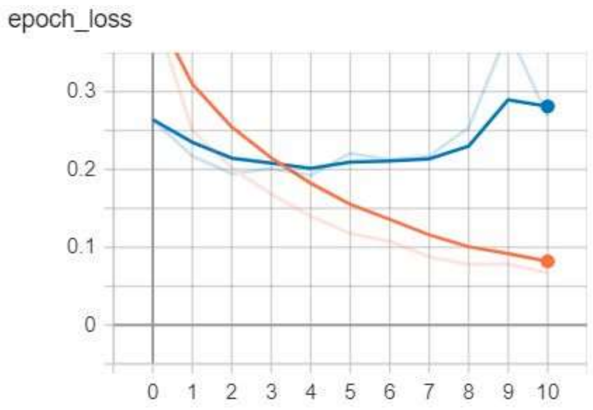
The first 3 blocks (convolutional + activation + max pooling) of the previous model were kept. Then 3 inception modules, separated by a max pooling layer 2x2, were introduced. At the end of the last inception module we inserted an average pooling layer 4x4 to reduce the spatial dimension before the flatten layer.
A more precise description of the new architecture can be found in the section 'Model building' of the notebook.
We also changed the set of transformations for data augmentation by adding the ones we considered useful to improve the performance of the new network. The best transformations for this model can be seen in the notebook.
At the end, the accuracy on the test set was quite good, i.e. 0.89555.

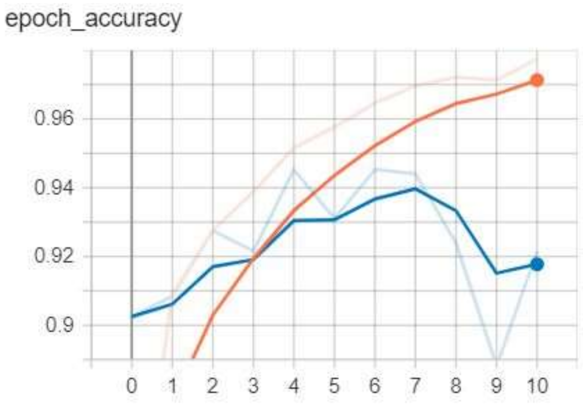
Reference notebook: TransferLearning
After achieving really good performances from scratch we decided to use transfer learning. There are a lot of pre-trained networks already trained with imagenet dataset. We tried both Vgg and ResNet but none of them had the same accuracy of Xception. The training process for Xception was divided into 2 different rounds. We first decided to use 'pure' transfer learning freezing all the layers of the feature extraction architecture. We deleted the output layer of Xception and we put a global average pooling and the final softmax output dense layer. The global average pooling was chosen to avoid overfitting and to maintain the number of learnable parameters relatively low. We had an accuracy of 0.73/4. Given that the imagenet has huge differences from our ds we had to also change the feature extractor by training some layers of Xception. The first round was fundamental to initialize the weights of the classifier in an appropriate way.





After some trial and error we decided to train all the layers after layer 95. We also reduce the learning rate of one order of magnitude since we didn’t want to lose all the features learned during the previous steps. The accuracy after just 5 epochs increased to 0.955 on the public test set. We were fine with this result even though we are sure that with more image augmentation and hyperparameter tuning we could do better. See you in the second challenge :)


epoch_loss


epoch_accuracy