ANN Homework1

Before starting to submit results in the competition, we first experimented some models on Jupyter Notebook, to get an idea of the overall complexity of the homework.
We started approaching the problem without using transfer learning, because we wanted to put more effort on training our custom models instead of starting with a pre-trained one. Before starting the training, we splitted the dataset in training set (85%) and validation set (15%), as we explained clearly in the notebook "*0-dataset-organisaton.ipynb*".

"Custom" models

The initial model we trained had 5 convolutional layers (*convolution* + *maxpooling* + *activation*) with 3x3 filters shape and an initial number of filters equal to 8 (that number is doubled after each convolutional layer), and stride 1. To prevent *overfitting* we used Early Stopping (with patience=5). We set the input shape to (408,612,3), while for the fully connected part of the network we used one layer with 512 neurons.
This initial model allowed us to reach a validation accuracy close to 0.7, so there was definitely room for improvements. Looking at the accuracy we understood that we had to improve the features extraction, so we tried to increase the number of convolutional layers with the aim of extracting more deep informations from the images; in practice we increased the number of convolutional layers to 7 and the number of filters to 10; this tweak led us to a substantial increase on the overall accuracy and allowed us to reach a 0.842 value on the test set. After that we also tried to change other parameters on *image augmentation*, and others like *learning rate* and the *batch size* but they did not provide a steady improvement in the results of our model. To reduce overfitting we also tried to perform *Dropout* on the fully connected layer, which gave us small improvements on the performance of the model. After performing different tries on our custom models, we decided to start approaching a solution with *transfer learning*.

Transfer Learning models

After different models tested without transfer learning, we tried to implement a transfer learning model in order to see the improvements achievable with respect to trained models. At the beginning we went for *VGG16*, but we didn't end up with a satisfying result. We've found out that small images, with not much in depth information can be treated well with *VGG16* whereas some images which contains more information can be treated well with *InceptionV3*. After trying both the pre-trained networks, we reached better results with *InceptionV3*, so we did some experiments on *hyperparameter tuning* to refine the model as best as we could. In that sense we tried different *finetuning* configurations (we tried to exploit only the first layers whose extracted features are more general) but we found out that keeping all the layers "freezed" was the best solution, resulting in an 0.92 accuracy value for the test set. To reduce *overfitting* we applied *Dropout* at the end of the dense layer. We also experimented with *learning rate* and *Early Stopping* parameters, in particular increasing the *patience* value from 5 to 8 allowed us to reach an 0.02 accuracy improvement on the test set. For the fully connected part we used a layer with 512 neurons. The final accuracy value reached by this model is 0.9377 (corresponding to our best submission).

Final considerations

After training both the models we were able to gain a 10% accuracy improvement on the test set (comparing our custom model and the transfer learning model). We were definitely expecting such an increase, but we're also satisfied with the results reached by training the model without using transfer learning.

As side note we also tried to rotate some images of the dataset (you can see it better explained in the notebook "1-image-analysis.ipynb"), to make it more uniform, but we didn't notice relevant changes on the result.