Challenge 1

Image Classification

Matteo Pacciani (Codice Persona 10525096 - Matricola 945104)
Oscar Pindaro (Codice Persona 10568511 - Matricola 946671)
Francesco Piro (Codice Persona 10534719 - Matricola 946450)

1 Dataset Management

For this challenge, we decided to split the dataset in two parts, one for training and one for validation. All of our models exploit early stopping regularization: the validation set in fact is used for this purpose, that is to check that the model does not overfit the training data. We reserved 90% of the dataset for the training set and 10% for the validation set (please see the attached script). This split has been done only once, at the beginning, and all our models use it. We tried to change that split, in terms of both the seed of the random function and the percentage of pictures dedicated to the validation set, but with no noticeable results.

2 Convolutional Neural Networks

Our first models, not based on transfer learning, are convolutional neural networks and use a quite basic infrastructure, with repeated couples of a convolutional layer, activated by a ReLU, followed by a max pooling layer, as presented during lectures. By manually exploring the hyperparameters space (essentially, the starting depth that is doubled with every new couple of layers, and of course the number of those couples), we managed to reach a 86% accuracy on the test set. We used the Adam optimizer with a learning rate of 1e-4. We gradually increased the complexity of the model and thus the total number of parameters, finding our best result with 40 millions. To handle this increased complexity, and prevent the risk of overfitting, we added some data augmentation and a small dropout in the dense layers.

3 Transfer Learning

Then we switched to transfer learning: our idea was to take the convolutional part of a well known architecture and append some dense layers and train them for our classification task. We initially focused on the Xception network but, by mistake, we didn't freeze any of the Xception layers and we also included the top of that network, that was made by a global average pooling (GAP) layer followed by a dense layer of 1000 neurons, of course trained on the 'imagenet' project. On top of those layers, we added other dense layers. With some data augmentation, but without any other form of regularization, we got 94% accuracy on the test set. We noticed the mistake, so we tried removing the top and adding only some "new" dense layers, preceded by a GAP layer, but this led to no improvement. We then tried following the Keras documentation, that advises to first make the Xception layers not trainable and to set those layers to "inference mode" (because of batch normalization), then append our classification part, and finally make the network converge on the new weights. As a very final step, the documentation suggests to refit the model with a 10 times smaller learning rate, but most importantly, to unfreeze some or even all of the Xception layers. Unfortunately, all of this was not successful. We then decided to exploit the "weight decay" method, because we noticed that the training error was able to get to almost zero (thus the model was probably complex enough), but with overfitting starting at about 90% of accuracy on the training set. Therefore, we added L2 normalization to all the Xception convolutional layers' kernels, without any improvement. We even tried regularizing also the bias and the output of those layers, with no success. Other attempts involve some hyperparameter search, for example by changing the batch size.

4 Conclusions

Seeing no further improvement using the Xception network, we tried some other Keras applications, like VGG16, DenseNet, NASNetMobile, but we didn't manage to get past the 94% accuracy obtained before. As a last resort, we tried bagging some of the models that gave us the best results, by majority voting on the output classes and this led to a slight improvement (95%) on the test set accuracy (please see the attached scripts).

5 Execution

To run our notebooks, you can run the script `splitTrainValid.ipynb` to split the dataset the same way we did, or you can download the zip file with the following Google Drive link: https://drive.google.com/file/d/1MJb_QOUb-WgOou_uH5FhzJ-O-1Rorl18/view?usp=sharing. Then you should make the variable "zip_path" point to that zip file.
All the attached files have a small description at the beginning.