ANN Homework3

We first approached this challenge thinking about the model structure: in fact understanding how to solve the problem of extracting information from both images and questions, and consequently combine them, was fundamental.
Our idea to solve this kind of task has been to split input into two different models. The first sub-model is based on usual Convolution to extract features from the images, while the second one is used to handle and embed the questions by tokenization. Finally the output of the two sub-models is combined via merge/concatenation, and processed to finals Dense layers, in order to assign probabilities to each possible answer/class.

Dataset Handling

At first we started gathering the questions and answers from the json file. Then we performed the question tokenization with either *texts_to_sequences* and *texts_to_matrix* methods. Thus we modified the Custom Dataset class in order to retrieve tuples of the form ([image, question], answer), to feed the network. Something similar has been done to create Test Custom Dataset, which is meant to return tuples like (image, question). We decided to use Categorical Crossentropy as loss function, thus we had to encode answers to one-hot vectors.

Resnet + LSTM Model

The first model we implemented exploited ResNet as transfer learning for the Convolutional part, and one LSTM layer for the question processing. The output of the convolutional part, appropriately resized by a Flatten layer and a Dense layer, has been finally combined to the LSTM part using a Multiply layer. This model made us reach an accuracy of 0.32.

Inception + LSTM model

After some reasoning on the problem, we noticed that we were not taking into account that features of the images appeared at different scales. Therefore we tried to use InceptionV3 as transfer learning to see if we could exploit its convolutions of different sizes to mitigate this problem. The improvement obtained with this model was not as good as expected; we reached 0.37 in accuracy.

CNN + Skip Connections + LSTM model

Since the previous model didn't perform as we expected, we decided to discard transfer learning and try to implement an other architecture, following the same intuition. The backbone of this new model is the standard sequence of CNN blocks, but using filter of different dimensions (7,5,3); consequently the feature maps obtained with this blocks have been concatenated through the use of skip connections, in order to build a more comprehensive vector containing informations of both high and low level. At the end we obtained pretty much the same results in test accuracy.

Simple CNN + Bag of Words

After various attempts with other similar models, leading to more or less the same results, in parallel we evaluated another approach to this problem: since the performances obtained with previous models were not too satisfying, we decided to reduce the overall complexity of the networks.
In this case we didn't use the custom dataset; the images have been simply loaded and resized with *nearest* metric to 32x32. The split between training and validation has been done by using the *validation_split* parameter of model.fit().
At first, we implemented a simple CNN to extract information from the images, that involved few classical conv blocks. On the other hand for the questions processing part, we decided to use a classic Bag of Words representation (given

the short length of the questions) to turn each question into a vector, used as input of a standard ff neural network. Finally, we used softmax to turn our output values into probabilities. This model allowed us to reach 0.60 accuracy. This last result exceeded our expectations, especially with reference to the simplicity of the model. For this reason we performed many different trainings to verify the result obtained, and all of them led to the same score.

We considered to use Attention for question's embedding, but given the short length of them all, we judged it inadequate.