Visual Question Answering

Davide Brescia
Matteo Ilari
Francesco Montanaro

1. Data loading

The first approach to the challenge was to load the whole dataset into a Pandas data frame (question, image id, answer). The questions relative to each image of the training set have been encoded into vectors of integer numbers by using the Keras API for word embedding, as shown in the notebook. Therefore, the original data frame has been split into training and validation data frames, with their relative percentages of observations, in order to perform cross validation.

After noticing that the models' training phase was extremely slow due to the large number of data, it has been decided to modify the dataset by reducing the size of the input images with a new one that consistently preserves their aspect ratio.
Moreover, to improve the speed of training further, it has been tried to load the resized data (about 1GB) into the RAM before feeding it to the network. However, the loading was extremely time consuming (3h. Colab session is resetted after 12h) and consequently it would not have speeded up the whole process, therefore, it has been discarded.

Finally, a custom Data Generator has been implemented before feeding the network. In particular, it has been used to preprocess the input data and to split the observations into batches of fixed size.

2. Experiments

General approach

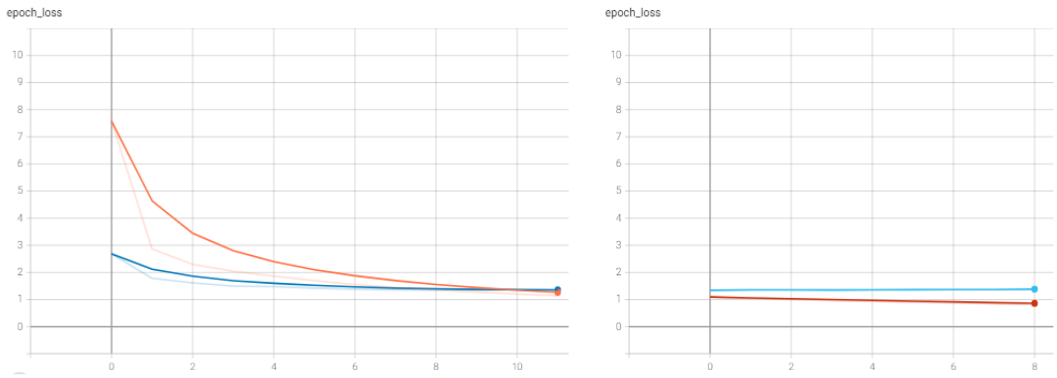
All the presented models are structured in this way:

- CNN to extract input image features. The choice of the particular CNN is the only difference in the architecture between the experiments.
- LSTM network to extract input question context. It consists of Embedding layer + Bidirectional LSTM layer, its complexity is characterized by *embedding_size* and *lstm_units* hyperparameters.
- Concatenation of image features and context as input of one hidden Dense layer and the output Dense layer (with softmax activation function).

Experiment 1

In the first experiment, a pretrained VGG16 with freezed weights has been used to extract image features. A Lambda layer simply ignores the 4° channel of the input image, as VGG supports only RGB format. After flattening of VGG output there is a trainable Dense layer with '*num_image_features*' units (hyperparameter).
The hyperparameters (lstm units, probability of dropout, number of images' features) have been tuned by minimizing validation loss, the best values are reported in the notebook. Learning rate: 5e-4.

loss:



Orange: training
Blue: validation

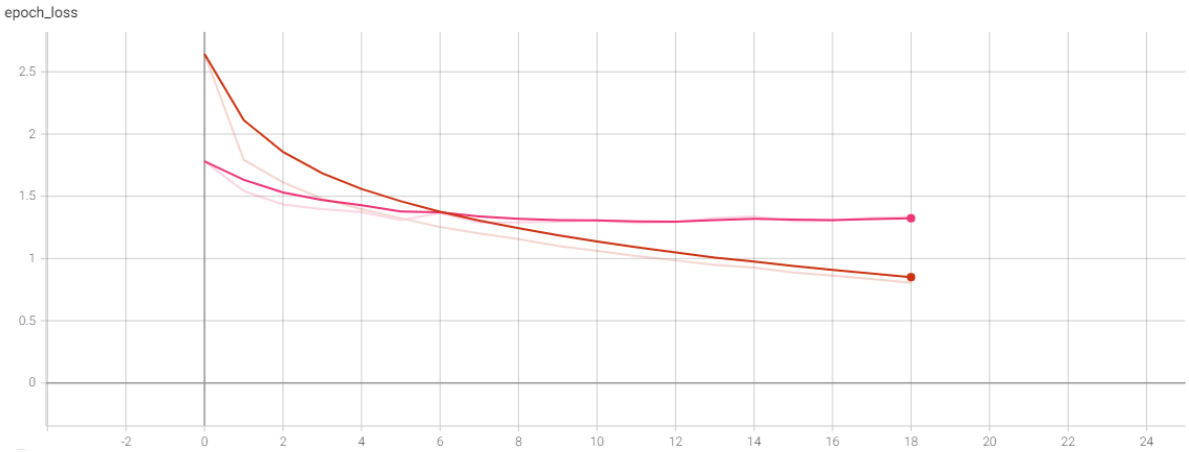
The achieved accuracy on the test set is ***0.54786.***

Experiment 2

In the second experiment, a model with a pretrained MobileNetV2 architecture has been used. For what concerns the hyperparameters used, the only difference with respect to the other models is on the learning rate (of *2e-4*), which has been found to be the best choice for this type of architecture.

Here's the final training trends of the best found hyperparameters:

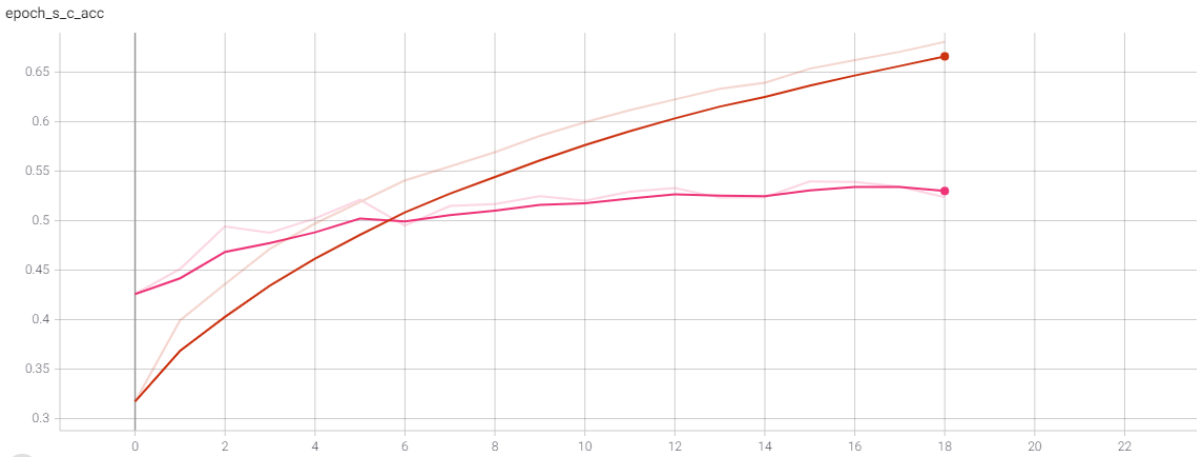
loss:



Orange: training
Purple: validation

Early stopping, monitoring loss, interrupted the training at this point.

accuracy:



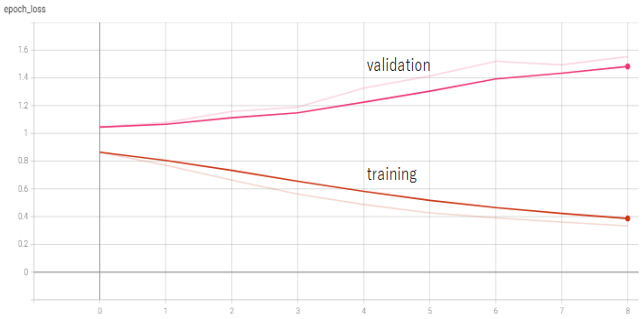
Orange: training
Purple: validation

The achieved accuracy on the test set is ***0.55069***.

Experiment 3

The best model.
A trainable custom CNN has been used to extract image features. The 4° channel of the input is considered; it is an important information for extracting the image content, but it may have resulted in an extension of the training time. The effective time per epoch was surprisingly lower than the previous experiments, the reason is that the preprocessing function just rescales the images and the number of CNN parameters (although trainable) is smaller. After flattening of CNN output there is a trainable Dense layer with '*num_image_features*' units.
The hyperparameters (lstm units, probability of dropout, number of images' features) have been tuned by minimizing validation loss, the best values are reported in the notebook.
Learning rate: 5e-4 for the first 'fit', 2e-4 for the second 'fit'.

loss:



accuracy:

The achieved accuracy on the test set is ***0.58176.***