Report - Homework 3

Mattia Calabrese, Federico Capaccio

To implement this Visual Question Answering model we had to make some research on-line to find the proper approach to the problem.

During the course we saw how this problem can be solved with Long Short-Term Memory (LSTM).

We came across the paper called “Hierarchical Question-Image Co-Attention for Visual Question Answering” by Lu, Yang, Batra and Parikh. In their paper they present a mechanism that jointly reasons about visual attention and question attention (from this the term “co-attention”). Sentences are embedded in a hierarchical way, at word level, phrase level and question level.

First of all, we make some analysis on the training data to find out precisely the numbers regarding possible questions and possible answers.

Here we found that the number of questions having “Yes” and “No” as answers are much higher with respect to the other questions.

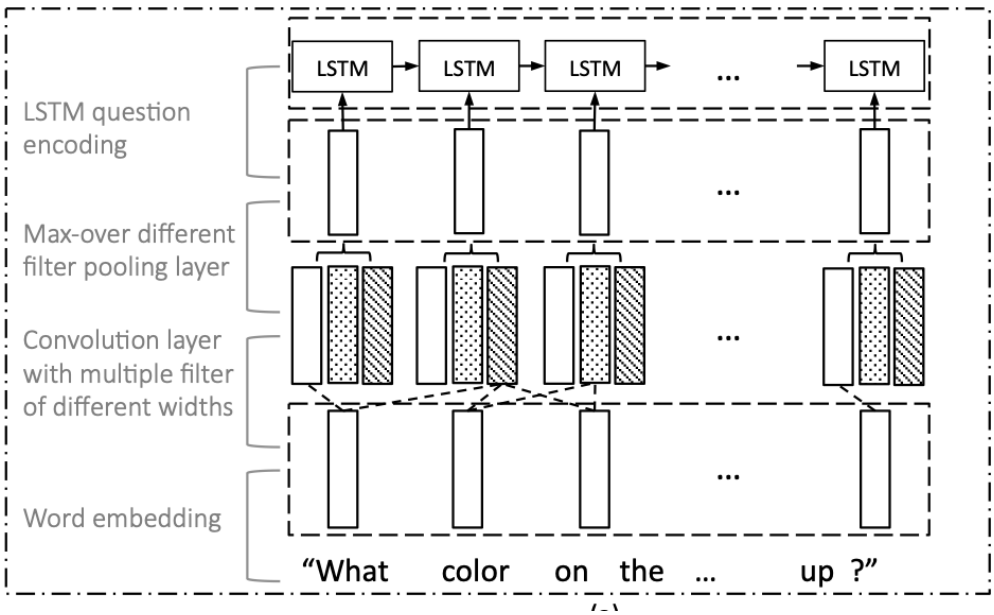
After that we start the image features extraction; this part is crucial to find the best features of the images to allow to model to know which are the most important parts of the images. We resized the images to make the computation less heavy and, to extract the features we uses VGG19 to process the images of the dataset.

After saving the image features we go on by cleaning the questions and analyze their length.

Then we went on by splitting the dataset in training and validation sets.

The model is based on a LSTM with attention mechanism: as said before, we combine the AttentionMap and ContextVector at three levels of depth, word, phrase and sentence, and embed them in the model with the LSTM

and the image features (the next picture is from the original paper and represent the structure of the model).



After training it, we encoded the possible answers to match the output of the model and made the predictions.

At the end the results were not really good (score˜0.3), and we started analyzing the possible reasons. The first thing we did was reducing the number of questions of the dataset that had "Yes" and "No" as answers to make the training dataset classes more balanced, but we could not figure out a way to remove non relevant questions/answers, so we had to remove them randomly, and this led the model to perform really bad. Then we tried to modify the image feature extraction part thinking that maybe the features found by VGG19 were not the right ones for our model. VGG16 did not find better features as the score did not improve.
At this point we thought that we could train an autoencoder to extract the features. The autoencoder had three convolutions, a flatten layer, a hidden layer (feature vector), a fully connected layer and three deconvolutions. We tried different sizes for the features vector (512, 1024, 2048) but our autoencoder either took too long to train because the model was too complex, or wasn't able to properly reconstruct the image, so the feature vector did not contain information that could make the model perform better on the test set.

The last thing we tried was to do class weighting when predicting the answers to fix the unbalanced classes in the test set. We thought that this approach could solve the problem, but after trying many formulas we could not find a way to make the prediction better.