Homework 3 - Visual Question answering

January 31, 2021

Task The task we must perform is called Visual Question answering: the model takes as input an image and a question and tries to answer the question (having a priori set of possible answers, so it doesn't have to generate them). The problem can be thought of as a classification problem because the answers are fixed and we don't have to generate them.

Dataset We performed a priori study on the dataset in order to see if the model is learning or randomly choose a question. There are 29333 images for training and 58832 questions. It's

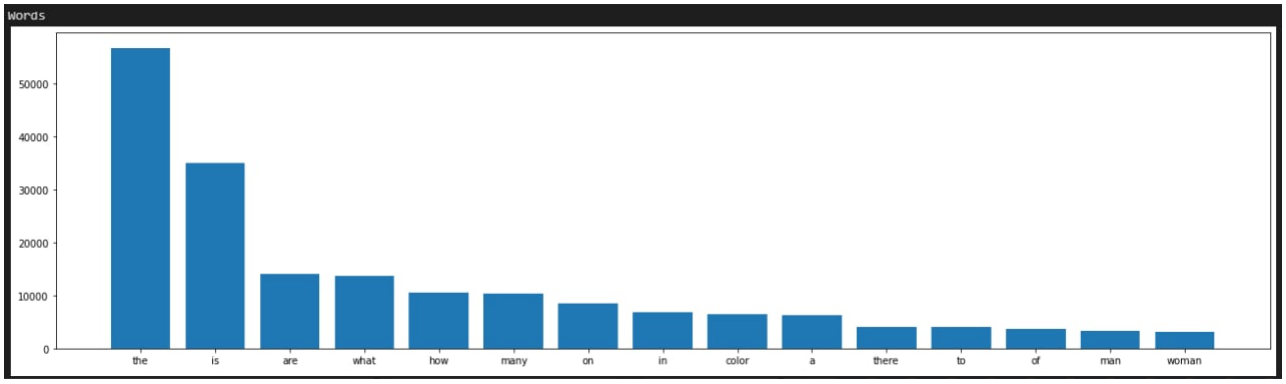

Figure 1: These are the most frequent words in the training questions.

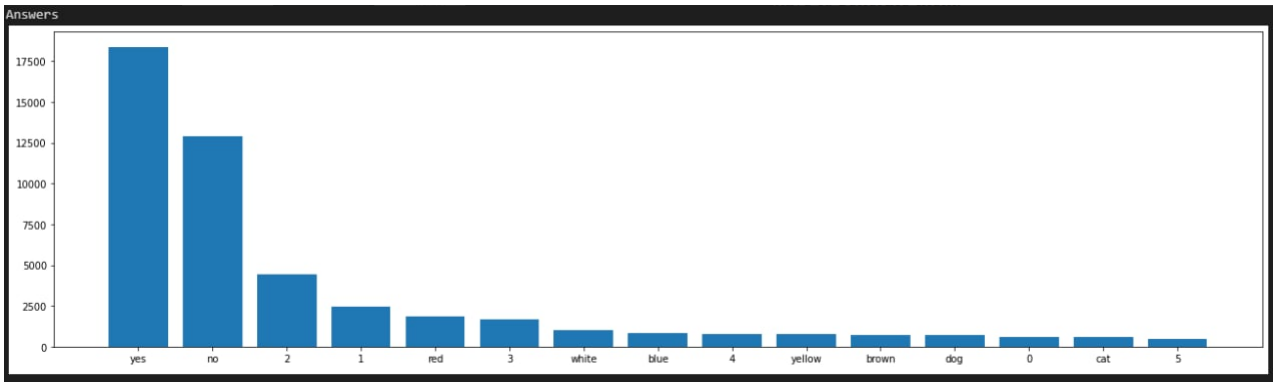

Figure 2: These are the most frequent answers.

interesting to see that half of the questions have a Yes/No answer. This worried us because the model will perform with a training accuracy of 52.7% just by learning to answer correctly yes or no. If the model randomly selects yes or no (worst case) the accuracy in training will be (approximately) 26.35%.

$$P(yes - answer) = \frac{18}{58.832};$$

$$P(no - answer) = \frac{13}{58.832};$$

$$P(selecting - yes) = P(selecting - no) = 0.5;$$

x = accuracy with a random guesser with only yes/no answers

$$x = \frac{18}{58.832} \times 0.5 + \frac{13}{58.832} \times 0.5 \simeq 26.35\%$$

Models tried Since the beginning we've decided to use transfer learning in order to extract features from the images. We divided the model in two portions: features extractor for the images and the encoder for questions. The feature extractor is composed of a CNN (Xception in this case) followed by a GAP layer and a dense layer. The encoder for the questions is composed of a LSTM followed also by a dense layer. In order to merge the two models (to produce a single output and merge the information) we concatenated the two dense output layers and then we used a Batch Normalization layer after flattening the concatenation. The flatten is then connected to a feedforward neural network as its input. The produced output will be 1 of the possible 58 answers (the last layer is a softmax composed by 58 units).

Training We had a problem for the training: the dataset is way too big to be fitted in the RAM during training and the generators provided by the framework are not able to load multiple inputs with a one to many relation between them (one image is associated with multiple questions). To solve this we implemented a custom generator which is able to generate the batch. Using a batch size of 32 we have 32 images associated with 32 questions and 32 answers as target. This is optimal for space requirements (RAM overhead for training is now negligible) but it's very slow and we experienced long training times (700 seconds for training depending on the images size and number of parameters to tune). To reduce the training time we resized the whole dataset (from 400x700 to 224x224) before training the model so the generator will load less at runtime, we then achieved a 2-3 minutes/epoch training time.

Hyperparameters Tuning and Loss This model has a lot of hyperparameters to tune: Learning rate; Percentage of the CNN to fine tune; Number of units in output from the CNN; Number of units in output from the question encoder; Number of units and layers for the final feed forward network; Dropout rates;

To find the best combination of those hyperparameters we trained different models at the same time with the same validation set and then we took the best hyperparameters values and trained a model with early stopping and a smaller batch size in order to take the best out of it.

Final result We reached a final result of approximately 63% on the test set.