POLITECNICO
MILANO 1863

Report of Homework 3
Artificial Neural Networks and Deep Learning 2020/2021

Tommaso Fontana Daniele Comi

29/01/2021

1 Introduction

In this competition it has been tackled the Visual Question Answering (VQA) problem on the following proposed data set. The data set is composed by synthetic scenes, in which people and objects interact, and by corresponding questions, which are about the content of the images. Given an image and a question, the goal is to provide the correct answer. Answers belong to 3 possible categories: 'yes/no', 'counting' (from 0 to 5) and 'other' (e.g. colors, location, ecc.) answers. An example can be seen in the following Figure 1.

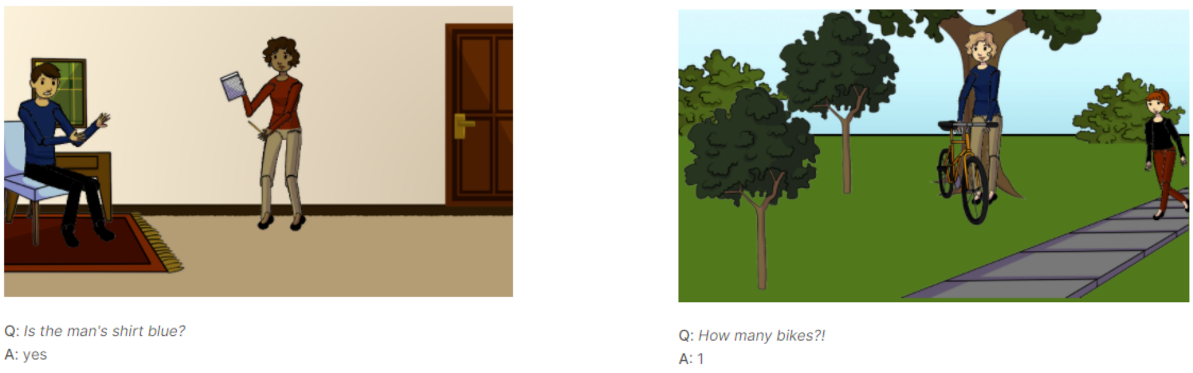

Figure 1: Example of possible Visual Question Answering samples in the data set and their provided answers.

2 Data set handling

The data set is constituted of 29333 RGB images of 400 x 700 and 58832 different textual questions, so multiple questions per image. Being the data set composed of synthetic scenes it didn't required any kind of augmentation in this specific case.

While the RGB images were of an original dimension of just 400 x 700, we choose to handle 200 x 350 RGB images so that we are both able to loose the minimum of information from the original, double-sized, input images but also be able to handle such an amount of images given our limited resources. We then also loaded in a lazy way the images in the RAM and pre-produced the elaboration, further discussed next, on the questions in order to get faster training per epoch.

3 Model Manual Tuning

The data set has been then further divided for validation purposes: 80% of the VQA data set was for training and the remaining 20% was for validation.

Having better hardware and time we could have used multiple holdouts, K-fold, cross-validation or Monte-Carlo Cross-validation to have a statistically significant evaluation of the performance of the model. Furthermore, having more resources we could adopt Hyper-parameters optimization strategies such as Grid-search and Bayesian Optimization which could improve the performance of the model but requires massive computational power.

4 Model Choice

For the image handling part it has been chosen as first architecture the VGG16 model which showed good results in validation but then we moved to using the InceptionResNetV2 model, given that gave us better validation results. Regarding instead the architecture for the question handling we decided that, instead of training our model from scratch or using some old embeddings like Glove or Word2Vec, to use a state-of-the-art model which is using Self-Attention with Transformers architecture. We initially chose BERT base of 12 layers but given that it was quite heavy we moved to an its simplification in depth using the DistilBert model. DistilBert, while being smaller than the actual BERT one, has too many parameters so we are not able to train it using our GPU. Therefore, we render the DistilBert embeddings of every question, this allows us to use the embeddings of the State-Of-The-Art model and speed up the training but the embeddings are not fine-tuned for the task. Moreover, DistilBert generate embeddings which size scales with the length of the question, but we need a constant size input to feed to the classifier, to handle this we naively average the emebddings on the words axis. Summarizing, this enables us to better handle each kind of question letting the model already have the best possible inference on sequence text data through the Self-Attention technique.

For each various model we always applied fine-tuning and we always added some regularization factor such as Early Stopping.

5 Architecture of the final model

The final model is so composed of InceptionResNetV2 for the image handling while DistilBert is being used to pre-produce (to enhance training speed) its interpretation of the question which is then given in input to a Dense layer as a vector of 768 elements.
We also added as said Early Stopping technique during training in order to prevent over fitting and stop the training at a reasonable epoch.
The actual model here described can be seen in the below Figure 2.

```
image_input = Input(shape=(*IMAGE_SHAPE, 3), name="image")
vision_model = tf.keras.applications.InceptionResNetV2(
    include_top=False,
    weights='imagenet',
    input_tensor=image_input
)
encoded_image = Flatten()(vision_model.output)
features_extraction_pipeline = pipeline(
    'feature-extraction',
    model=NLP_MODEL, #DistilBert
    tokenizer=NLP_MODEL,
)
question_data = Input(shape=(768,), name="question")
h = Concatenate()([question_data, encoded_image])
h = Dense(128)(h)
output = Dense(len(labels_dict), activation='softmax')(h)
vqa_model = Model(inputs=[encoded_question, image_input], outputs=output)
vqa_model.compile(
    loss="categorical_crossentropy",
    optimizer=tf.keras.optimizers.Nadam(),
    metrics=["accuracy"],
)
```

Figure 2: Architecture of the final model.

The final model has a total of 61520538 parameters.

6 Training

The training parameters are summarized in Table 1.

Table 1: Training parameters

Training argument	Parameter	Reason
loss	Categorical cross entropy	It's a multi-label classification.
steps_per_epochs	2942	Using a batch-size of 16.
validation_steps	736	Using a batch-size of 16.
Weight initialization	Xavier	To have better starting weights which should speed up the training and let us have a better final result of gradient descent.
optimizer	Nadam	Dozat (the author) says that is should give better results than Adam which is the de-facto standard.
learning_rate	0.001	Value which allowed the model to learn.
beta_1	0.9	Value suggested in the paper.
beta_2	0.999	Value suggested in the paper.
Callback	Early Stopping	To stop the model when it reaches convergence and it act as weight decay which help preventing over-fit.
delta	0.001	A small enough value, it's $\approx \frac{1}{100}$ of the final loss of the models.
patience	10	A patience big enough that the RLROP can act at least 3 times before stopping.
restoring best weights	True	When having a validation set True is the best option.
Callback	RLROP	Reduce Learning Rate On Plateau, this should help the model achieve better performance in the last part of the training.
RLROP factor	0.1	This value was taken from an example.
RLROP patience	3	We set the patience so that it can trigger 3 times before the early stopping kills the process.
RLROP delta	0.001	The same delta of the early stopping.

For the weight initialization with Xavier Initialization we used GlorotNormal to better initialize weights W and letting back propagation algorithm start in advantage position, given that final result of gradient descent is affected by weights initialization.

$$W \sim \mathcal{N}\left(\mu = 0,\ \sigma^2 = \frac{2}{N_{in} + N_{out}}\right)$$

We also set a checkpoint folder in order to save our model state during the training process after each epoch.
For reproducible result we set a constant seed in order to have the same pseudo-randomization on data by TensorFlow and by Numpy. The chosen seed was "randomly" 1234.

Also, to really speed up the training process at each epoch we decided to load and prepare all these training data into RAM without letting it read and elaborate every time from the disk, this really improved speed and reduced latency allowing us to explore more options.

7 Results

With this architecture as it's shown in the below Figure 3 it has been reached an accuracy in the validation as best value of 59.43%, given that we set the parameter restore_best_weights to True.

On the loss graph we can observe over-fitting, so further regularization might be needed but the accuracy on the validation set does not differ much.

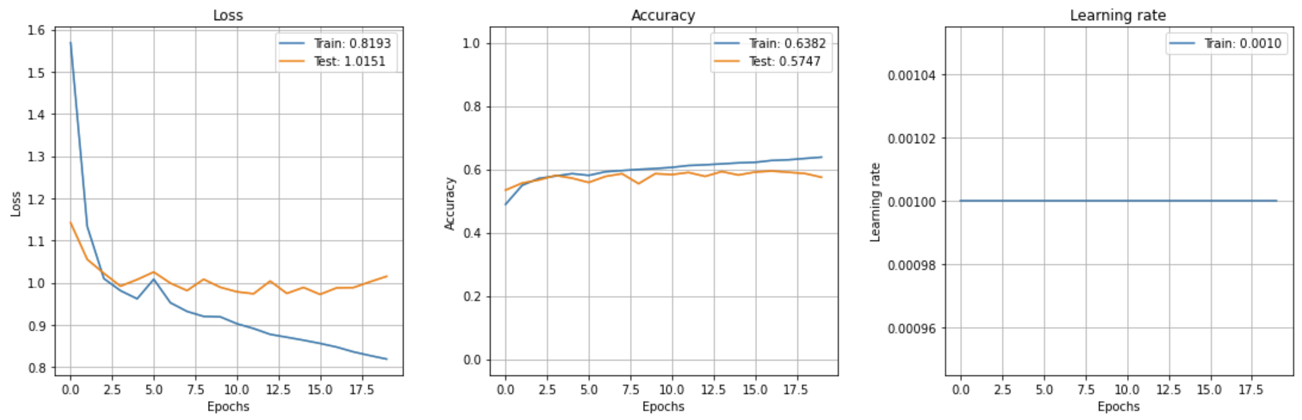

Figure 3: The training results of the VQA model. (What in the graph is called test is actually the performance on the validation split, this is due to a bug in the library)

Here in Figure 4 it's also shown just for explanatory reasons a small sample of the provided test set with the corresponding predicted answers.

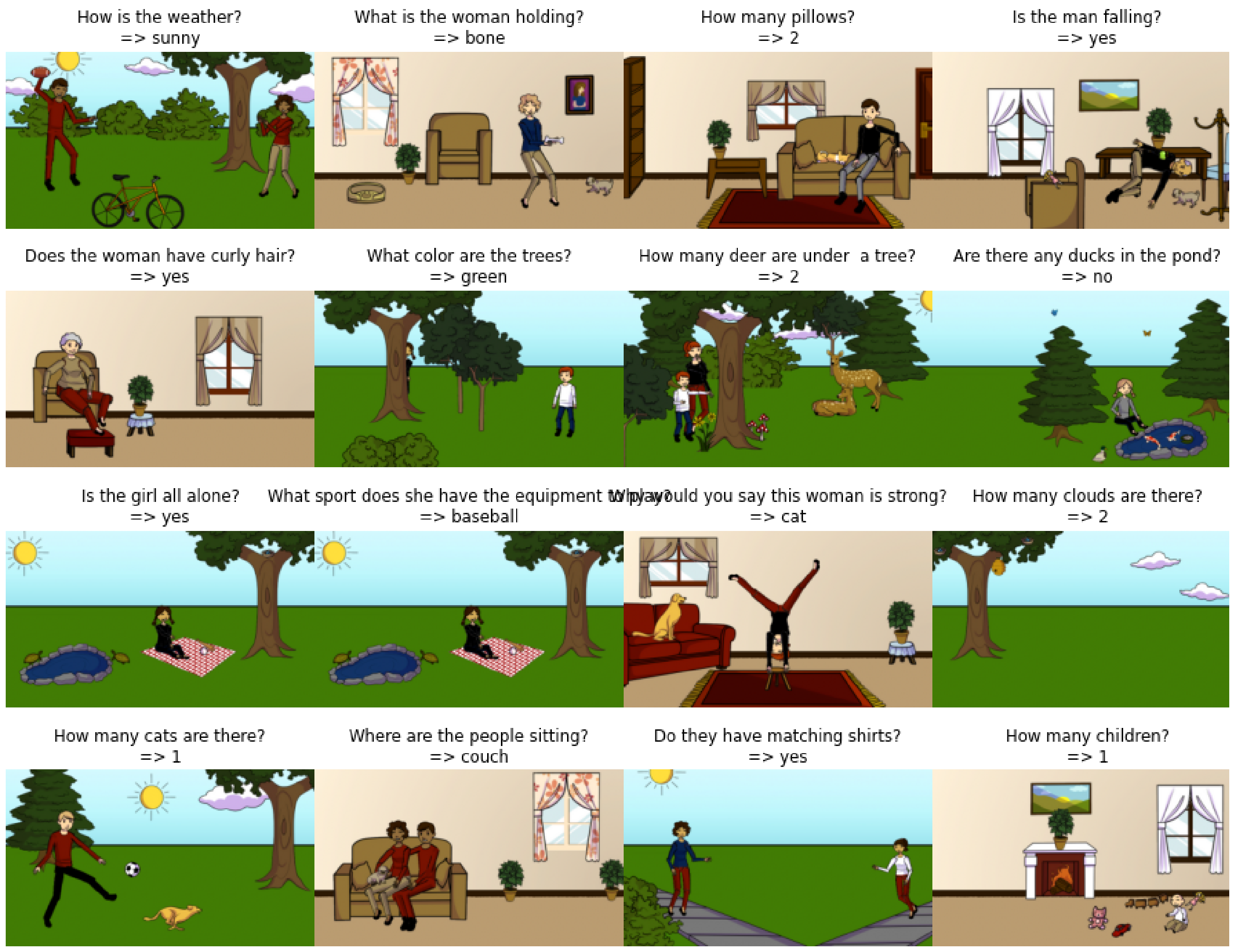
How is the weather?
=> sunny
What is the woman holding?
=> bone
How many pillows?
=> 2
Is the man falling?
=> yes
Does the woman have curly hair?
=> yes
What color are the trees?
=> green
How many deer are under a tree?
=> 2
Are there any ducks in the pond?
=> no
Is the girl all alone?
=> yes
What sport does she have the equipment to play?
=> baseball
Why would you say this woman is strong?
=> cat
How many clouds are there?
=> 2
How many cats are there?
=> 1
Where are the people sitting?
=> couch
Do they have matching shirts?
=> yes
How many children?
=> 1

Figure 4: Sample of prediction results of the VQA model on the test set.

Nevertheless on the test data on Kaggle for some reason we couldn't figure out why we've reached a very low accuracy of 9.21%, while on validation we get 59.43%. This could mean that the validation split has a bias, but argue that we used the same ShuffleSplit method from SKlearn which we used also in the other homework, or we might have data miss-alignment when computing the test answers or covariate-shift in the test data.

8 Conclusions

Again, we've come to the same conclusion that such Deep Learning and Machine Learning problems are much affected by little but important changes, there is much more than, for example, the number of layers in a model to be considered. We've seen the importance of having to regularize a model in presence of little data in order to get the best of it, helping us also various other techniques.
In this challenge we have witnessed also the power of Neural Networks in their finding of non-linearities patterns and features in such a complex task like VQA where it would have been impossible otherwise without this approach and it's quite interesting how it is possible to combine results of essentially two completely different networks and architectures in order to predict a combined task of the two.

9 Possible improvements

We want the network to see most of the possible details, so one possible improvement would be to use the tiling technique so that we would be able to virtually train the network on the full-sized images and so let it being able to spot most of details.
Another possible improvement would be trying to apply the concept of Self-Attention not only to the text data but also to the features maps generated by the CNN model part of this VQA architecture. Another embedding aggregation method that we think it's promising, when aggregating the DistilBert embeddings, is doing a weighted average using techniques such as binned TF-IDF to identify which are the most important "features" of the embeddings of each word.