Visual Question Answering

21/01/2018

François Darmon
francois.darmon@telecom-paristech.fr

Yousri Sellami
yousri.sellami@telecom-paristech.fr

Abstract

In this poject, we are interested in open-ended Visual Question Answering : a task where given an input question about an image, one should provide an accurate answer. Using the MS Coco dataset, we follow the work of Agrawal, Lu et al. [4] and try to improve their results by implementing Multimodal Compact Bilinear Pooling (MCB). We compare quantitatively our results to theirs and to our baselines, study parameters sensitivity and interpret the results qualitatively.

1. Introduction

Trans-disciplinary tasks are one of the future challenge of AI. While CNN and RNN have shown great results in respectively computer vision and NLP, one remaining question is how to combine these architectures to perform complete AI task such as Visual Question Answering. In this project, we will first describe our experiment setup (training and validation sets, training of different architectures, implementation concerns...). Second, we propose different baselines based on text only or on text and image. Third, we will present the MCB combination method and discuss its integration into the overall architecture. Finally we compare the results, study parameters sensitivity and give qualitative interpretations and future possible improvements.

2. Article Summary

Visual Question Answering was introduced in [4]. In this article, the authors present a new database created for VQA. They also present a network to perform VQA as a baseline for forecoming VQA challenges.

2.1. Database

The dataset consists of a set of images and a set of questions associated with one of these images. The image dataset is composed of COCO Dataset [3] and a set of generated images that we have not used for the project. The question dataset was collected with the help of workers that have proposed questions related to images. The labelling was performed similarly by selecting the most frequent answer out of ten provided by other workers. In the end, the train/validation/test split is chosen from coco dataset's split. There are 82,783 images associated with 248,349 questions in the training set and 40,504 images associated with 121,512 questions in the validation set. We did not use the test set because the labels are not available on the web since the dataset was part of a challenge. So we used the validation set as our testing set.

2.2. Network Architecture

The method proposed by [4] is a deep learning method for both the image and the question part of the data. First, questions and images are dealt with separately respectively with a LSTM and a CNN. This part is the feature extraction part of the network. Then an embedding layer reshapes the images and questions features to a common size then both are combined with a pointwise multiplication. Finally a perceptron performs the classification into the 1000 most common answers.

As training a CNN may be really long, the authors have chosen to use an existing architecture and its pretrained weights: VGGnet [8]. These convolutive layers are frozen for the training and in fact, for memory efficiency, images are seen as features vector from the beginning. However, the LSTM part was trained from scratch.

3. Baselines

In this section, we present the results of our baselines. Especially, we split these baselines into two categories: those which only use questions (text-based) and those which use both question and image (image and text-based). It is a natural split because there is a strong biais between text and image in VQA. Indeed, a correct answer is likely to be provided by just understanding the question without the image information (especially if it's a yes/no or number questions) whereas the contrary is impossible. ‘

3.1. Text-based baselines

The predicted answer is the answer associated to the nearest neighbour found with LSTM features and cosine similarity.

3.2. Text and image based baselines : k-nn

We implemented the following k-nn baseline : first find the k nearest questions (we tried TFIDF BoW, Word2Vec), then among the k associated images find the nearest image using VGG net features and return the corresponding answer. As a result, the parameter k quantifies the trade-off between prioritizing text or image information. For a small k, the baseline will only focus on understanding the question whereas for a large k, it will find the closest image regardless of the question. We set k=4, the results are shown in Table 1.

3.3. Comments on the results

Unsurpringly, text-based baseline performs better than k-nn on Yes/No questions because the image information is less important and it can hurt the understanding of the question in k-nn trade-off. On the contrary, k-nn performs better than text-based baseline on question ”other” because there are more difficult question that requires an understanding of the image.

The baselines results are close to those of [4], the small range difference comes from the fact that we evaluated on the validation set instead of the testing set.

4. Multimodal Compact Bilinear pooling

4.1. Motivation

As assumed by [2] concatenation or element-wise multiplication implemented in is not probably as expressive as an outer product between visual and textual vectors. Intuitively, the reason is that an outer product contains the information about all the interactions between the textual content and the image content. Unfortunately, an outer product is unfeasible because of its high dimensionality. MCB is a way to address this problem.

4.2. Algorithm

Let x_{txt} and x_{im} be respectively the text and image representation vectors. The goal of MCB is to project the outer product $x_{txt} \otimes x_{im}$ to a lower dimensional space without computing the outer product directly.

As suggested by [5], we use the Count Sketch projection function ψ to project x_{txt} and x_{im} to a vector of dimension d. The outer-product projection is computed using the fact that:

$$\psi(x_{txt} \otimes x_{im}) = \psi(x_{txt}) \star \psi(x_{im}) \quad (1)$$

$$= \text{FFT}^{-1}(\text{FFT}(x_{txt}) \odot \text{FFT}(x_{im})) \quad (2)$$

where $\star$ is the convolution operation and $\odot$ is the element-wise multiplication. The first equality was proven by [7] and the second equality is a fast way to compute convolution using Fourier Transform properties. The pseudo code of the algorithm can be found in Annexe D.

5. Experimentation setup

We focused on the open-ended task. We tried first to replicate the results from [4] using the LSTM part as a feature extraction. We used the trained LSTM from [4] and did not train it at all. An image and a question associated are then stored as two features vector. We tested several configuration, with/without normalization, MCB with several dimensions or pointwise multiplication.

We used the data and the trained model from [6]. As explained in section 2.1 we used the training set of MS Coco 2014 and its associated question as a training set and the validation set of MS Coco 2014 as a testing set.

The question feature extraction was performed once and for all by modifying Lua/Torch code from the github repository [6]. Then we processed the database and trained the neural network using Python and Keras package. The implementation can be found on [1].

6. Results

6.1. Baselines

The baselines are evaluated in terms of precision on the whole dataset and on the separation on questions (Yes/No, numbers, others) in table 1.

	Open ended questions			
Baselines	All	Yes/No	Number	Other
Text based baselines:				
Nearest Neighbour on LSTM features	35.8%	67.2%	21.7%	14.8%
Text + image based baselines:				
k-NN + TFIDF on BoW	32.3%	54.4%	16.6%	19.4%
k-NN with Word2Vec	35.8%	63.3%	16%	19.8%

Table 1: Baselines performances

6.2. Neural Networks

We then tried a neural network classification for Visual Question Answering. Each question feature vector and image feature vector are forward passed throught the Neural

Network. The output is a length-1000 vector that corresponds to confidence in each of the 1000 possible answers. The best score is selected as the final answer of the network. Then a precision score can be computed for the entire dataset, for the Yes/No questions, for the numbers questions and for the other questions.

We first evaluated the performances using pointwise multiplication as a way of combining the text and image part. Then, we evaluated the performances for the MCB combination.

6.2.1 Pointwise multiplication

After the embedding layers that changes the input to a common size, we tried two different perceptrons with one and two layers and tried with a L2 normalization of the image features and without. The result are shown in table 2.

We did not have access to the detail of the performance of [4] on the validation set but on average, our performances are below theirs. This may be due to the fact that the LSTM layers were frozen during training. There is a slight improvement when using L2 normalization. Adding a hidden layers to the perceptron has led to poorer performances. This may be because the LSTM layers were frozen to a specific training performed in an architecture using a single layer. It can also simply be a consequence of overfitting.

	Open ended questions			
Method	All	Yes/No	Number	Other
Model trained in [4]	54%	-	-	-
One layer normalization	48.8%	72.8%	28.6%	36.0%
One layer no normalization	47.0%	72.7%	28.6%	32.5%
Two layers normalization	44.3%	70.5%	26.5%	30.7%

Table 2: Pointwise multiplication performance

6.2.2 MCB

We then tried MCB with various size of the output vector. This size must be between the largest size of input and the size of the outer product of the inputs. We tried sizes 5000, 8000 and 15000. Size 15000 models were very computationaly and memory expensive and we did not succeed in finding a good convergence of the neural networks. Table 3 shows the results for 5000 and 8000.

There are not much differences in performance for the 4 architecture tried. However there is a clear improvement of the performances using MCB instead of pointwise multiplication.

	Open ended questions			
Method	All	Yes/No	Number	Other
Pointwise multiplication	48.8%	72.7%	28.6%	36.0%
MCB 5000:				
No normalization	50.6%	75.0%	31.5%	37.2%
Normalization inputs	51.1%	75.0%	31.7%	38.0%
MCB 8000:				
No normalization	50.2%	74.7%	31.6%	36.6%
Normalization inputs	51.0%	75.0%	31.6%	38.0%

Table 3: MCB performances

7. Conclusion

Our overall results are 3% below the results of [4]. As mentioned earlier, this is probably due to the fact that the LSTM layers were frozen during training. However, MCB shows encouraging improvements (+2% in average, see Figure 4, Appendix for more details) and seems able to better combine visual and textual information.

An interesting way of pursuing this work could be to investigate whether this improvement is still visible when training the complete network, without freezing the LSTM part.

References

[1] F. Darmon and Y. Sellami. VQA project. `https://github.com/fdarmon/VQA`, 2018.

[2] A. F. et al. Multimodal compact bilinear pooling for visual question answering and visual grounding. *CoRR*, abs/1606.01847, 2016.

[3] L. T. et al. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[4] S. A. et al. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015.

[5] Y. G. et al. Compact bilinear pooling. *CoRR*, abs/1511.06062, 2015.

[6] J. Lu, X. Lin, D. Batra, and D. Parikh. Deeper lstm and normalized cnn visual question answering model. `https://github.com/VT-vision-lab/VQA_LSTM_CNN`, 2015.

[7] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *KDD*, 2013.

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Appendices

A. Qualitative evaluation of the question features

It is possible to roughly evaluate the question features by looking at the distances between the feature vectors. We recall that we obtained these features by performing a forward pass on the training set with the LSTM part of a pre-trained model from [6]. To do so, we display for a given question in the test set, the three nearest neighbours in the training set (using cosine similarity). In the large majority of cases, we get satisfying nearest neighbours as shown in the first part of Table 4. However, in some cases (see part 2 of Table 4), the features fail to order correctly the closest questions.

How many fruits and vegetables are there ?
How many fruits and vegetables are green ? (0.944)
How many fruits are in the image ? (0.82)
How many fruits are on the table ? (0.814)

(a) Coherent example

Is this a whole orange ?
Is this dog full grown ? (0.853)
Is this cat full grown ? (0.837)
Is this a whole cake ? (0.815)

(b) Incoherent example

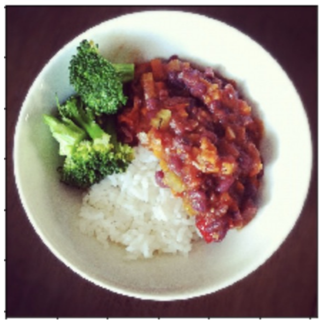
Table 4: Example of nearest neighbours for questions features

This qualitative evaluation encouraged us to use the text feature as an input of our model. This allowed us to drop the LSTM part which is the most computationally expensive part.

B. Qualitative evaluation of the models

In this section, we evaluate qualitatively the models on non-easy examples (we exclude yes/no questions and answers that can directly be deduced from the questions).

Figure 1 shows two examples where all models perform accurate object detection (broccoli) and activity detection (baseball). In Figure 2 are represented two difficult questions that were misanswered. In the first question, the difficulty comes from the fact there are lots of people to count (number questions are the most tricky for our models, see results) and not all of them are holding umbrellas (the question is complexified by a constrain). In the second example, the object detection is difficult because the image is dark. In both cases, although the answers predicted by our models are wrong ('2' and 'bench'), because they fail to incorporate enough image information, they are the most likely answers given the question.


(a) What vegetable is on the plate?

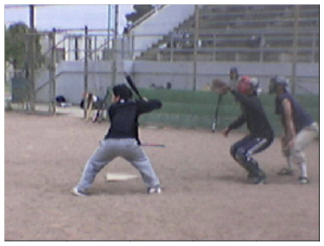
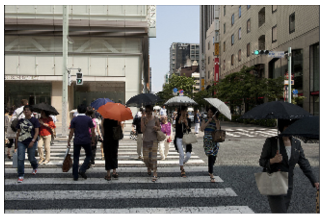

(b) What sport is it?

Figure 1: Example of images where both model output the correct answer


(a) What number of people are holding umbrellas?

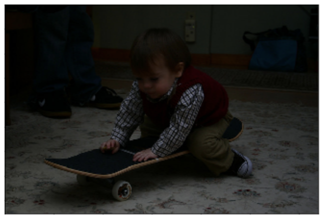
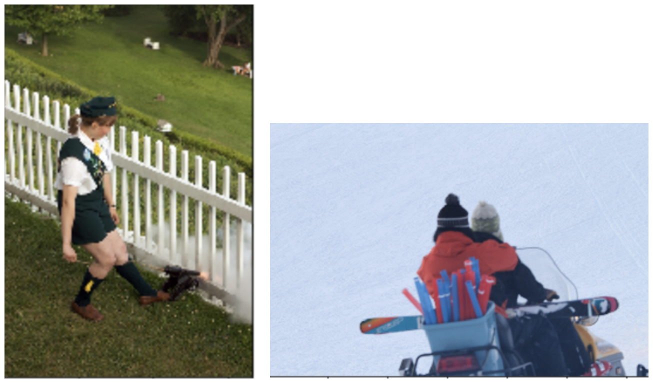

(b) What is the child seated on?

Figure 2: Example of questions hard to answer

Figure 3 highlights on two examples the advantage of MCB over pointwise multiplications. In the first example, to the question "What color is the fence?", MCB models answer correctly 'white' whereas pointwise models answer 'green' which is the color of the grass in the image. It shows that MCB improves phrase location accuracy (detects the word 'fence') to better combine visual and textual information. Intuitively, it makes sense since MCB approximates the outer-product which stores each interaction between the visual information (colors green and white detected) and the textual information (word fence). In the second example, to the question "Why would the UNK drive up the mountain for skier?", MCB models answer correctly "safety" whereas pointwise models answer "walking". Here we see that MCB manages to extract more information from the image and then combines it correctly with the question.

C. Comparison of the models by the question type

The dataset provides also a category for each question (64 categories in total). It is possible to analyze the differences of our models based on the question type. Figure 4 shows the relative accuracy difference between the best



(a) What color is the fence ? (b) Why would the UNK drive up the mountain for skier?

Figure 3: Example of images where MCB is better than pointwise multiplication

MCB model and the best pointwise model. For the sake of clarity, we only showed the category of the top 4 biggest differences.

Although MCB has globally better performances, there are 19 categories where the pointwise model performs better. It is interesting to note that two of the most important improvement of MCB are for similar questions that differ only by the use of plural. MCB shows largest improvements on questions which require more visual information (”Where is?”, ”What is the name”). It is coherent with the qualitative results shown above.

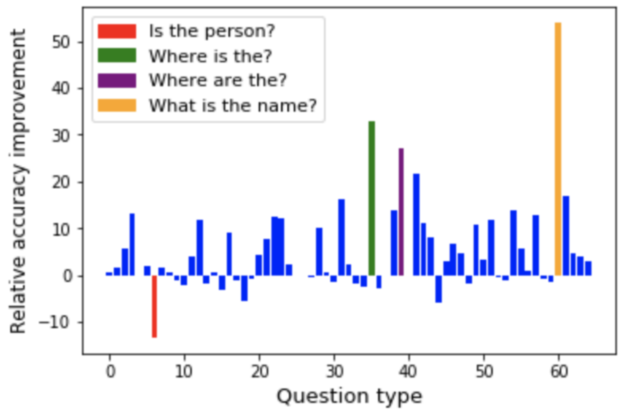


Figure 4: Relative accuracy improvement from the best pointwise model to the best MCB model

D. Pseudo-code MCB

The MCB operation is a convolution of the Count Sketch Projection of the image features vector and the Count Sketch Projection of the question features vector. The pseudo code for the Count Sketch Projection can be found in Algorithm 1. It needs an input data vector and the two randomly chosen vector h and s of the count sketch projection.

```
Algorithm 1: Count sketch projection
  Data: $v \in \mathbb{R}^d$, $h \in \{1 \ldots d_{out}\}^d$, $s \in \{-1, 1\}^d$
  Result: $y \in \mathbb{R}^{d_{out}}$
  $y = [0, \ldots 0]$
  for $i = 1 \ldots d$ do
      $y[h[i]] \leftarrow y[h[i]] + s[i] \cdot v[i]$
```

This function is used for combining the images and questions datasets into the final dataset. The complete algorithm is shown in Algorithm 2. The parameters h_{im}, h_{qu}, s_{im} and s_{qu} are chosen randomly at the beginning and fixed to the same value for the training dataset and for the testing dataset.

```
Algorithm 2: MCB combination algorithm
  Data:
  $images \in \mathbb{R}^{n,d_{im}}$
  $questions \in \mathbb{R}^{n,d_{qu}}$
  $h_{im} \in \{1 \ldots d_{out}\}^{d_{im}}$
  $h_{qu} \in \{1 \ldots d_{out}\}^{d_{qu}}$
  $s_{im} \in \{-1, 1\}^{d_{im}}$
  $s_{qu} \in \{-1, 1\}^{d_{qu}}$
  Result: $R \in \mathbb{R}^{n,d_{out}}$
  for $i = 1 \ldots n$ do
      $y_{im} \leftarrow$ csProjection $(images[i,:], h_{im}, s_{im})$
      $y_{qu} \leftarrow$ csProjection $(images[i,:], h_{qu}, s_{qu})$
      $R[i,:] = \mathrm{FFT}^{-1}(\mathrm{FFT}(y_{im}) \cdot \mathrm{FFT}(y_{qu}))$
```