Show and Tell: A Neural Image Caption Generator - Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan

Clément HARDY, Ruth KUEVIAKOE

April 6, 2019

Introduction :

In this article we will see how to automatically provide a description of an image thanks to deep learning. This process is called Image Captionning.

The difficulty of this task compare to an object detection is the fact that the model doesn't need to only detect the object but also to detect interactions between them. For example, if the input is a image with children playing soccer in a garden. The object detection model only need to detect the children, the ball and eventually the fact they are in a garden. Here, the model has also to detect the fact the children are playing soccer, this add a really complex task to the previous one. In addition, the model has to translate is result to a language understandable for a human being. The Recurrent Neural Networks (RNNs) offer the possibility to do such a translation with state-of-the-art performance.

The model

Motivations

The usual method of encoder-decoder for a RNN is the following : the goal of the encoder is to encode an input sequence. By that, we mean transform a "human" sentence into a fixed length vector representation. The decoder which comes after this encoder use this vector representation to produce a series of words (a sentence).

Here, the input is an image so we will follow this approach but instead of choosing RNN as an encoder, we'll take a deep convolution neural network that will transform the information contained in the image. Indeed, CNN can produce a rich representation of the input image by embedding it to a fixed-length vector. But training a complex neural network from scratch could be very difficult because it needs a really big dataset as well as an important computational power. Transfer learning solve this problem by offering to use pre-train model in a different task-dataset has initial state. In our case, we will use a pre-train CNN for an image classification task (a dataset like imagenet).

After this CNN model, a RNN model will take place, it will use the informations transmitted by the CNNs to produce a caption for the initial image.
In the end, our model can be seen as a single network that generates descriptions of images.

In the datasets used for this task, images are associated with captions written by humans. So the goal of the model will be to learn the "correct" caption. In other word, to maximize the probability that the caption given by the model is the correct one, we can formulate it by:

$$\theta^* =_x \sum_{(I,S)} log(p(S \mid I; \theta))$$

with θ the parameters of our model, I an image, and S its correct transcription.

The correct transcription S is a sentence which has an unbounded length. We therefore apply the chain rule for the joint probability over $S_0, ..., S_N$, with N the length of the sentence. Dropping the dependency on θ for convenience, we finally obtain :

$$log\ p(S \mid I) = \sum_{t=0}^{N} logp(S_t \mid I, S_0, ..., S_{t-1})$$

RNN Model

As we seen above the goal is to maximize the probability of a predicted sentence. This probability can be decomposed with the chaine rule:

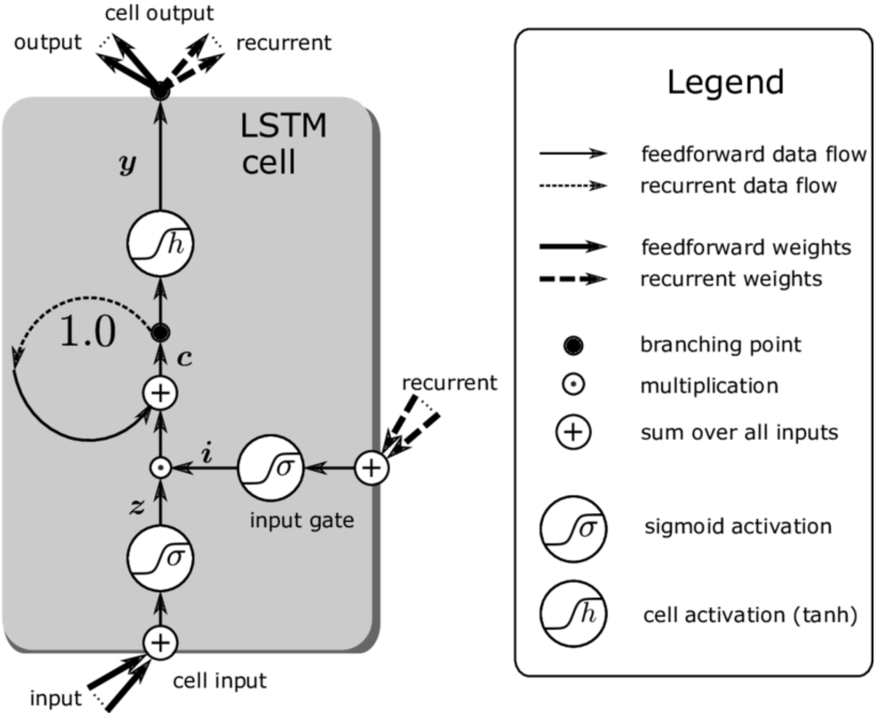
$$\begin{aligned} log\ p(S \mid I) &= log\,(p(S_t \mid I, S_{t-1}..S_0)p(S_1 \mid I, S_0) * p(S_0)) \\ &= \sum_{t=0}^{N} logp(S_t \mid I, S_0, ..., S_{t-1}) \end{aligned}$$

With this reformulation, we can see it like maximizing the probability of next words given the previous ones. So to maximize $p(S_t \mid I, S_0, ..., S_{t-1})$ for every t, this task will be done by a recurrent neural network. The variable number of words up to t-1 is expressed by a fixed length hidden state or memory h_t. The memory is then updated after seeing a new input x_t by using a non-linear function f such that $h_{t+1} = f(h_t, x_t)$. x_t represents the words through an embedding model. As for f, we use the long short term memory (LSTM) to deal with the problem of vanishing and exploding gradients that arise with RNN.
The main characteristic of LSTM is the memory cell c that encodes knowledge at every time step of the inputs observed up to this step. How the cell behave is controlled by different gates.

In LSTM, there are three different layers :

- a forget gate, to control wether to forget the cell or not
- an input gate, to control if ti should read the inputs
- an output gate to control wether to output the new value or not

In our case, LSTM will be used We can see in the figure below how a LSTM cell works :



To be more easier to picture, we will think of the LSTM in its unrolled form where all recurrent connections are transformed to feed-forward connections. As all the LSTM will be linked together the output of one LSTM (time t-1) is the input of the next LSTM (time t); we need that all those LSTM share the same parameter (for each word of the sentence) so in that aim a copy of LSTM will be done for each word of the sentence (to make sure they have the same parameters). More theoretically, we can see the process of translating the image through three step :

- At t = 1, the vector representation of the image is the input of the first LSTM. (The vector is input once to avoid overfitting and the exploitation of the image's noise).
- At time t we use the one-hot vector S_t representing a word in the sentence (S_0 is the start word and S_N the stop word). We compute $x_t = W_e S_t$
- an output gate to control wether to output the new value or not

During this process, we would to maximize the probability see above, so we have to minimize the loss $L(I, S) = -\sum_{t=1}^{N} logp_t(S_t)$ with respect to all the parameters of the LSTM, the top layer of the image embedder CNN and word embeddings W_e

We will use two different approaches to generate a sentence given an image.

The first one is the **Sampling method**, we sample the first word according to p_1 (by taking the max of the probabilities), then we use the result as input for the next step which give the second word, so on like this until we sample the stop word token or some maximum length.

The second one is the **BeamSearch**. Instead of keeping the more probable word at each step, we keep the k more probable sentences. So, at time $t-1$ we have the k most probable begin of the sentence, for each being of sentence we predict the k most probable following word (which give in total k^2 begin of sentence) and we then keep only the k best and so on until all possible sentence are finish or the maximun length is reach.

Metrics

In our case we want to evaluate the description generation task rather than ranking the description sentences because the complexity of the image will grow with the dictionary. The sentences will have to grow too. The sentence will grow exponentially and this will be hard to handle. To evaluate the sentence describing the image, we can use the rate given by the users from 1 to 4 qualifying the quality of the sentence. But we can also use metrics that can be computed automatically like the BLEU score. BLEU (bilingual evaluation understudy) score has been shown to correlate well with human evaluations. It is based on the principle that "the closer a machine translation is to a professional human translation, the better it is". Its score is for comparing a candidate translation of text to a reference translation. It can be used to evaluate text generated for a suite of natural language processing tasks. This metrics works by counting how many n-grams in the candidate translation match n-grams in the reference text. 1-gram or unigram represents each token and a bigram comparison take into account each word pair (and so on for trigram..). The comparison is made regardless of word order. The more we can find matches, the better the candidate translation is. A perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0. The counting of matching n-grams is modified to ensure that it takes the occurrence of the words in the reference text into account, not rewarding a candidate translation that generates an abundance of reasonable words. This is referred to in the paper as modified n-gram precision.

Implementation

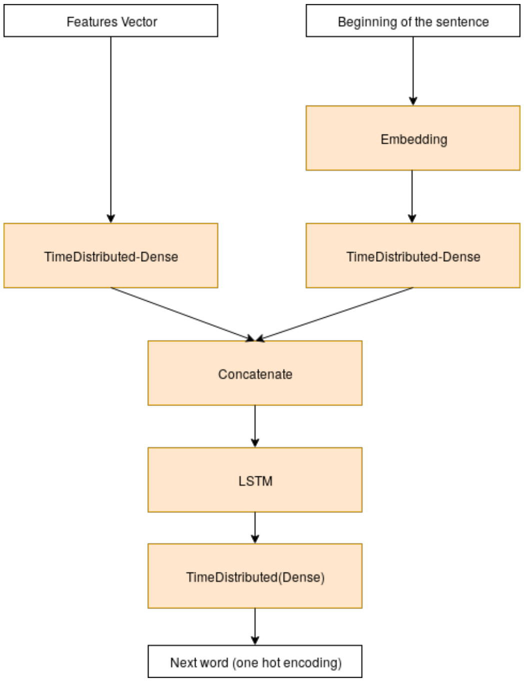
Methods

We have implemented the model in two different way. The first idea was two to implement the complete model (cf code in model 1 folder), by that we mean the feature extractor and the recurrent network connected to each other like in the paper. But, we quickly realize that it was really computationally difficult to train such a model. One iteration was long to proceed and we could only take a small batch size as our GPU hasn't enough memory. The idea to train the recurrent network and then fine tuning the complete model was maybe a good idea but unrealizable for us. Moreover as we used model with weights pretrained in imagenet, fine tuning the features extractor will lead to overfitting.
So we change our approach, we have consider the fact that the pretrained model give "perfect" result and doesn't need to be train. In consequence, we put each image of the dataset inside the features extractor one time and then save the result. The results ave will then served as inputs for the recurrent network. This method offer the possibility to increase

the size of the batch which lead to faster learning, moreover one iteration was quicker. The results presented in this rapport has been done with this method (this model is implement in the folder model 2).

Architecture of model

We have decided to use two differents features extractor which are Resnet50 and InceptionV3 both for their performance. For the recurrente network, we have implemented the following architecture:



For inference, we implemented both the sampling and beam search methods.

Results

During the exprements, we used three differents datasets, flickr 8k, flickr 30k and IAPR TC-12. In the beginnig, we only use flickr 8k because of the reasonnable number of images inside the dataset, 8000 images with 5 caption for each image. But in the end, when our code was working and we had our first results, we were curious how the performance will increase with more images and caption for training.

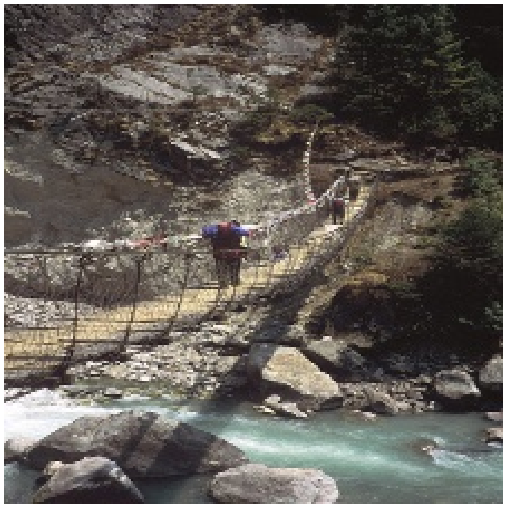
Flickr 8k

The dataset was separated in the following proportion:

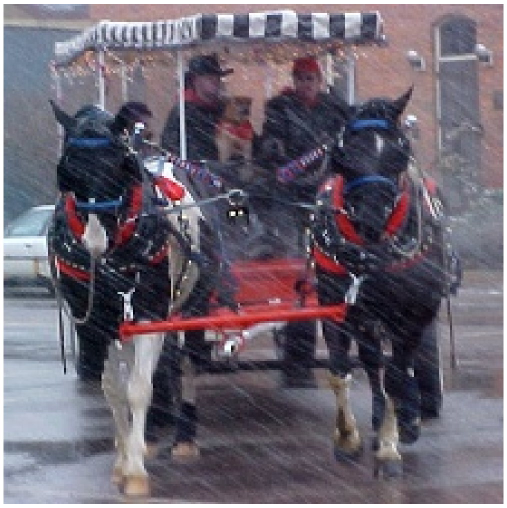
- 85% for training
- 10% for the validation set

- 5% for testing

We had train our model until the validation accuracy stop to improve, the accuracy was 0.63 in the training dataset and aroung 0.51 in the validation set.

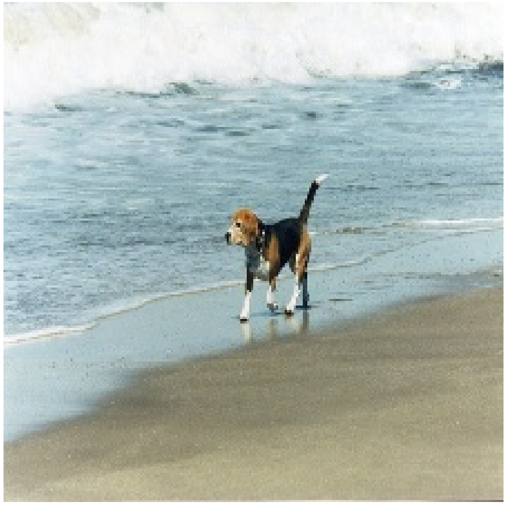


Real caption: Hikers cross a bridge over a fast moving stream and rocky scenery.
Sampling method: People are crossing bridge in a wooded area.
BeamSearch with k=2: The lady in the pink dress is out of three people look at the ocean.
BeamSearch with k=5: The girl in the pink dress is standing in the ocean.
BeamSearch with k=20: A lady walking across the beach.

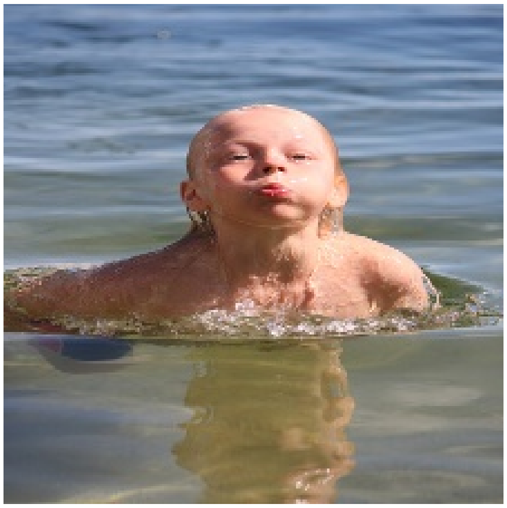


Real caption: A cart containig two men being pulled by horses in the rain. Sampling method: Two horses are racing along a rainstorm.
BeamSearch with k=2: A woman in a blue vest and gold holds a black bull.
BeamSearch with k=5: A woman in a blue coat is holding a black bull.
BeamSearch with k=20: A woman in a hat is holding a bull while a man watches.

By the two photo above, we can remarks that the precision and the meaning aren't really satisfactory. The beam search method seem to have some trouble to find even the general context, the sampling method seem to do a little better job.
It's the opposite as we can see in following pictures.

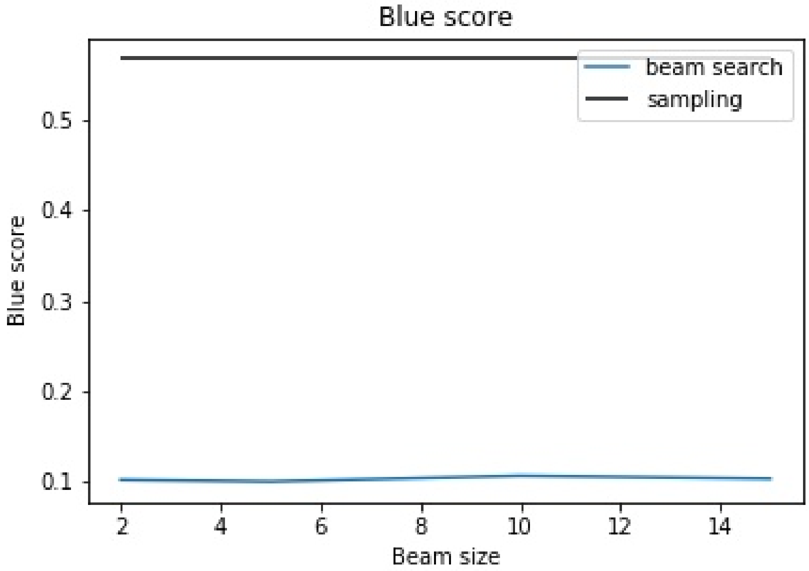


Real caption: A beagle dog is walking on the shoreline at the beach.
Sampling method: A man in a white dress points to a young girl on a beach.
BeamSearch with k=2: A man doing a trick as others jump.
BeamSearch with k=5: A man doing a trick as others jump.
BeamSearch with k=20: The dog is running across the beach.



Real caption: A blonde child swims.
Sampling method: Two young boys of one one three of one three of one one one of one one of one of them play of of of them of of them of of them of of them of of of of.
BeamSearch with k=2: A little boy smiles.
BeamSearch with k=5: A little boy smiles.
BeamSearch with k=20: A young boy submerges from the water.

We can remarks that both sampling and beam search method have variable performance depending one the picture. This could be cause by the fact that we only have 8000 images for training. But, in general the sampling seem to give better result, at least in understanding the most important aspect of the picture, as a presence of a person, a horse...
Let's look at the blue score to determine which method are the best in our case (even if the model doesn't seem to have train enough).
The following score has been calculating by doing the average of blue score of all images in the test dataset.



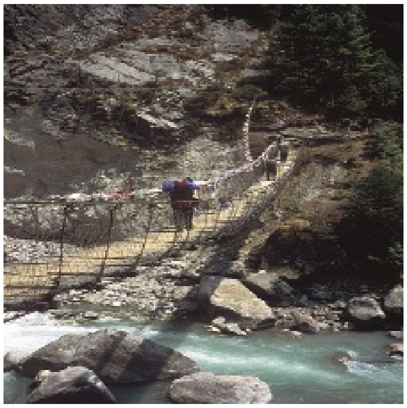
Even if the sampling method has a beam size, we have choose to draw a line to represent his score. We can notice that beam search method's score is low . So in general, the sampling method is better for us. The beam search may need more training data to offer a better score.

Flickr 30k

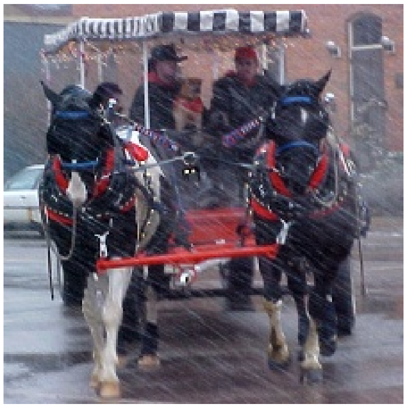
Let's see if a bigger training dataset will lead the model to generalize better the prediction of captions. Indeed, this dataset contains 30 000 images compare to the previous one. As this dataset has been created by the same peoples, we can expected that the type of captions is nearly the same making the comparison of score beetween the two datasets more interesting. We have separated the dataset in the same proportion than the previous one:

- 85% for training
- 10% for the validation set
- 5% for testing

Let's take the same pictures and look if we can see any improvements.



Real caption: Hikers cross a bridge over a fast moving stream and rocky scenery.
Sampling method: A man in a blue shirt and a woman in a black shirt are standing in front of a large waterfall.
BeamSearch with k=2: A man in a black jacket and a hat is standing in the snow with a large red umbrella.
BeamSearch with k=5: A man in a red jacket is standing in the snow surrounded by snow.
BeamSearch with k=20: A man and a dog are walking through the snow.

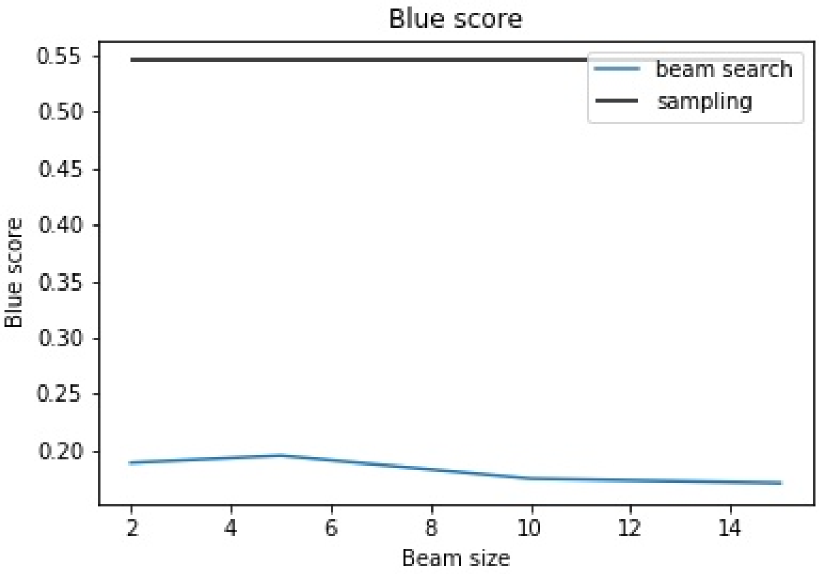


Real caption: A cart containig two men being pulled by horses in the rain. Sampling method: A man in a red shirt is riding a horse.
BeamSearch with k=2: A man is riding a horse in a rodeo.
BeamSearch with k=5: A man in a red shirt is riding a horse in a rodeo.
BeamSearch with k=20: A man in a cowboy hat is riding a bucking horse in front of a crowd.

Real caption: A beagle dog is walking on the shoreline at the beach.
Sampling method: A man in a black shirt and a woman in a black shirt are walking along a beach.
BeamSearch with k=2: A dog is running through the sand.
BeamSearch with k=5: A dog is running through the sand.
BeamSearch with k=20: A black and white dog is running on the beach.

Real caption: A blonde child swims.
Sampling method: A young boy in a blue bathing suit is swimming in the water.
BeamSearch with k=2: A man in a blue shirt is standing in the water.
BeamSearch with k=5: A little girl in a pink bathing suit is swimming in the water.
BeamSearch with k=20: A young boy is swimming in a body of water.

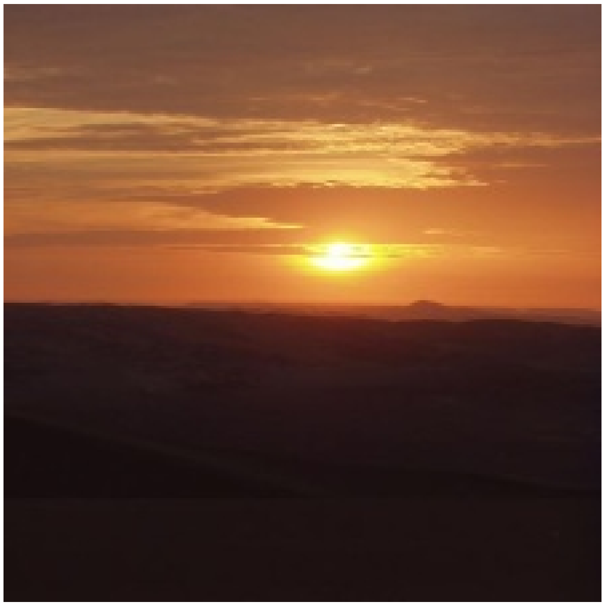
The performances of the predictions seem to be similar with the previous dataset (not a big improvement has been done). Except the that the grammar seem to be a little better, we found less sentence with a repetition of words like with the previous dataset (the swimming pool picture).
However, the beam search perfomance has increase as the blue score show:



IAPR TC-12

We tought it was interesting to choose this dataset because this one is completely different to the previous datasets, the type of captions as well as the type of images (lot of landscape). Indeed, this dataset contains 20 000 naturals images taken from all around the world.

The result is comparable with the previous, the predicted captions are great for the easiest images as we can see above but when the image is a bit more complex the model seem to have some difficulties to capture the context of the image and to provide a good caption.



Automatic caption: a dark orange sky in the background

Improvements

The easiest improvement we could have done is to fine tune the CNN model, as we said in the introduction the CNN have to give to the RNN the object, personn present in the image but also the interaction between them. For a computational issue and amount of data we choose to ignore this and only use a pretrain CNN for classification on the dataset imagenet.

Criticism

The approach done in this article permit to give a global description of the image but many image doesn't contain only one action, important thing. It could be intersting to see if an approach like it's done in the object detection task by dividing the image in region and to give for each region of the image a description. Eventually, only the most important caption could be kept.

This is one of the weakness of this approach, a dense captioning approach is probably more appropriate in most cases.

An other improvement which lead to better result is the fact to introduce attention. Indeed, with attention the model can focus on different part of the image during the generation of description. This could offer better result has the important part of the image isn't necessary the same for the beginning of the sentence and for the end of the sentence. This approach also have the advantage to offer a better interpretation of the result (as we could show where the model had focus in the image).

References

[1] *Flickr 30k*. URL: https://www.kaggle.com/hsankesara/flickr-image-dataset.

[2] *Flickr 8k*. URL: https://www.kaggle.com/srbhshinde/flickr8k-sau.

[3] *IAPR TC-12*. URL: https://www.imageclef.org/photodata.

[4] Oriol Vinyals et al. "Show and Tell: A Neural Image Caption Generator". In: *CoRR* abs/1411.4555 (2014). arXiv: 1411.4555. URL: http://arxiv.org/abs/1411.4555.