Image Captioning
Machine Learning for Natural Language Processing 2020

Etienne Sullice
ENSAE Paris
etienne.sullice@ensae.fr

Laroche Charles
ENSAE Paris
charles.laroche@ensae.fr

Abstract

Image captioning is a very popular topic because of its broad scope of application. For instance, it can help visually impaired people, or be used for image and video indexing. Image captioning also enjoys great popularity as it involves two of the most popular fields of deep-learning: computer vision and natural language processing. In this paper, we elaborate an approach to image captioning that combines features extraction with ResNet 101, language generation using LSTM and Attention mechanism.

1 Problem Framing

Our objective is to translate a given image into a sentence. For this purpose, we first need to analyse properly the image in order to get the objects, their position, their colours... Then, we can generate sequentially, i.e. word after word, the full caption of our image. We treat each generation process of a word as a classification problem of our vocabulary. From a probabilistic point of view, this means that we want to find the transition matrix, given the features and the previous word, namely: $P(word_{t+1}|word_t, h_t)$, where h_t is the LSTM hidden state.

2 Experiments Protocol

Data: We used the Common Object in Context (COCO) database of 2017[1] which groups images with at least five captions each. Because of time constraint and GPU restriction on Google Colab, we could only use 10,000 images, split into training and validation sets of size 2/3 and 1/3 respectively. For pre-processing, we converted all images into (256, 256, 3) tensors, then we normalized them. To handle the captions, we attributed a number to each word of the vocabulary with a correspondence dictionary. Then, we transformed all the captions into vectors of integers. Formally, we started every sentence with a token ”<start>” and ended them with a token ”<end>”. Finally, we padded the sentences so they all have a length of 51 words.

Model: Our model is composed of two main algorithms. First, we encode the image to extract all of its information with the **Encoder**. The output is then a small image of size 14x14 with 2048 learned channels. We call this image the *features vector*. The second step consists in generating the caption, which is what our **Decoder** does. It decodes the features vector sequentially. That is, we predict one word at a time, using the last predicted one and a hidden state. For initialization, we use the token ”<start>”. Then, once the Decoder generates the token ”<end>”, we interrupt the process. The architecture of our image Encoder and Decoder are described in the following sections.

Encoder: The Encoder converts the input image made of 3 colour channels into a smaller picture composed of 2048 learned channels. To do this, we took a pre-trained **ResNet 101** (Kaiming He, 2015) on which we removed the dense layers and the last max-pooling. Thus, the resulting encoded vector is a summary of the original image with all the most important features in it. As a reminder, ResNet is a deep convolutional neural network (CNN). It is differentiated from other CNNs by its residual connections that prevent the gradient from vanishing or exploding during the back-propagation of deep CNNs. It is pre-trained on ImageNet database and known as one of the best models for image classification.

Decoder: The Decoder has to generate the caption word after word, which orders a recurrent

[1]Available at: http://cocodataset.org/#home

structure. A recurrent neural network (RNN) is a network that works sequentially. For example, to predict a given sequence $(s_1, ..., s_p)$, a RNN will first return $\widehat{s_1}$ and a hidden state h_1. For the second prediction $\widehat{s_2}$, the RNN will use h_1 and the input ; for the third, h_2 and the input ; and so on. The advantage of a recurrent network is that we consider the information given by the previous predicted element to get the next one. Here, we use the Long Short Term Memory (LSTM) network (Sepp Hochreiter, 1997). LSTM networks are a type of recurrent networks with special activation functions that prevent gradient from exploding or vanishing. Their architecture is summarized in the Appendix (Figure 1). Knowing the words of the caption we have predicted so far, we would like to know on which part of the image we must focus next, in order to predict the following word. This is what the Attention mechanism (Hu, 2018) allows us to do. It gives us the ability to focus on parts of the image by giving weights on the pixels (features in our case). The more important are the pixels, the greater are their weight. We are then able to use their weights to display the pixels and visualize which part is used for the prediction of each word (Figure 3). The Figure 2 summarizes the full architecture of our model. We used PyTorch to code the model.

Training: As we consider the problem as a classification task, we use the **Cross Entropy** loss for the training. We also compute the BLUE criterion (Kishore Papineni and Zhu, 2002) in the validation step to do the early stopping: if the BLUE does not increase after many steps, we interrupt the training. In sum, our final model was run on forty epochs for a period of four hours in total, using checkpoints. As a Google Colab stops in case of inactivity and the training was quite long even on the 10,000 images, we decided to stop here because of time constraints. We also resort to *teacher forcing* for the training stage. Instead of using the last word prediction as an input to predict the next one, we took the "ground truth" i.e. the true word, for a quicker and more efficient training. Besides, we have five captions for each image, so we pick on one randomly at each epoch to make it more general.

Evaluation: We test the quality of our predictions with two quantitative metrics. As we don't really need our model to predict the exact word of the caption, we consider first the **top-k accuracy**. It checks if the ground truth is in the top-k predictions or not. Moreover, we use the **BLUE score**. To compute BLUE, we predict a sentence and for each word generated we set a score of 1 if the word is in the true caption and 0 otherwise. In the end, we average the score of all the words predicted. Unfortunately, the BLUE metric has some weaknesses. For example, if we always predict words like 'a' or 'the' our BLUE is going to be very high. This is why we assume that the two metrics combined can provide a better measure of the quality of our model. On the other hand, we base our qualitative evaluation on the generation of captions on both validation set images and external images (see Figure 5 for fun results on our Facebook's profile pictures).

3 Results

On the validation set, we finally get a BLUE score of 0.18 and a top-5 accuracy of around 70%. For qualitative results, we provide in Figure 4 captions generated on random images of the COCO dataset. The model generates captions that make sense even if it has some weaknesses. Regarding the low number of samples and epochs on which the model was trained on, we are generally pleased by the predictions. We made all our results reproducible in our Google Colab (see also here the GitHub of the project).

4 Discussion/Conclusion

As a baseline, we investigated the possibility of a non-deep-learning model for image captioning. We thought about a logistic lasso regression that predicts a word from both the features vector of the image and the features vector of the previous word, projected using word2vec. Unfortunately, we could not run the model because of RAM issues.
Besides, on the deep-learning part, we believe that it could have been interesting to introduce a GAN based loss on the predicted sentence to improve the quality of the generated caption.

References

Jürgen Schmidhuber Sepp Hochreiter. 1997. Long short-term memory.

Todd Ward Kishore Papineni, Salim Roukos and Wei-Jing Zhu. 2002. Blue: a method for automatic evaluation of machine translation.

Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. 2015. Deep residual learning for image recognition.

Dichao Hu. 2018. An introductory survey on attention mechanisms in nlp problems.

Appendix

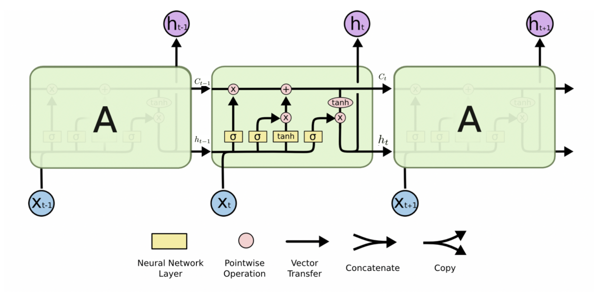

Figure 1: Behavior of a LSTM cell

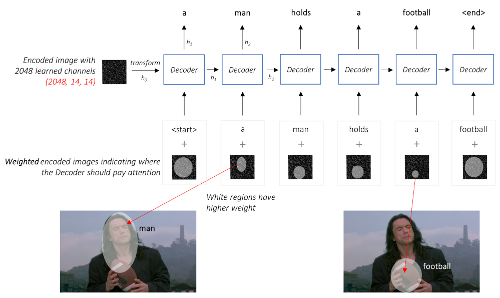
a
man
holds
a
football
<end>
Encoded image with 2048 learned channels (2048, 14, 14)
transform
Decoder
<start>
a
man
holds
a
football
Weighted encoded images indicating where the Decoder should pay attention
White regions have higher weight
man
football

Figure 2: Architecture of the Decoder

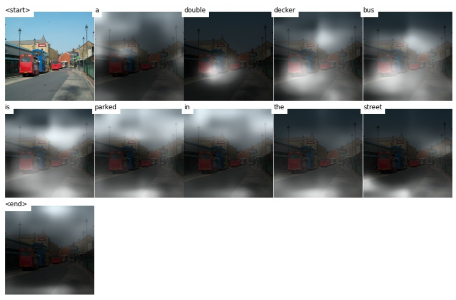

Figure 3: Visualization of Attention

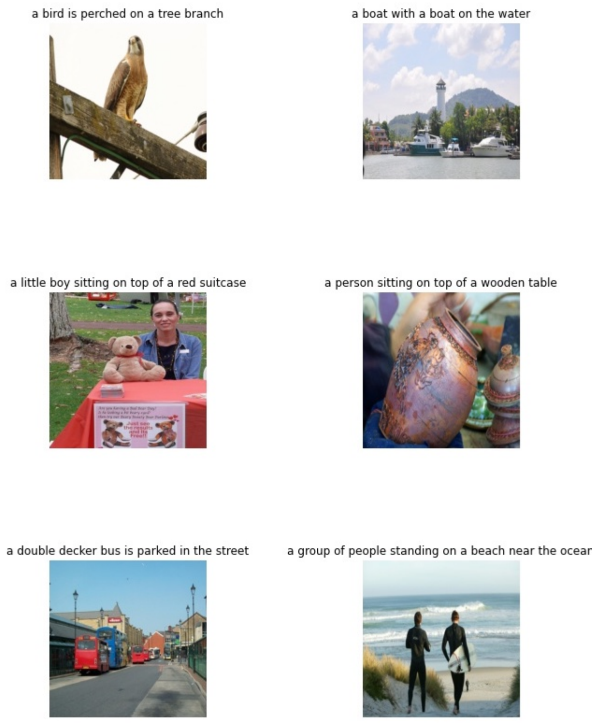
a bird is perched on a tree branch
a boat with a boat on the water
a little boy sitting on top of a red suitcase
a person sitting on top of a wooden table
a double decker bus is parked in the street
a group of people standing on a beach near the ocear

Figure 4: Examples of captions generated

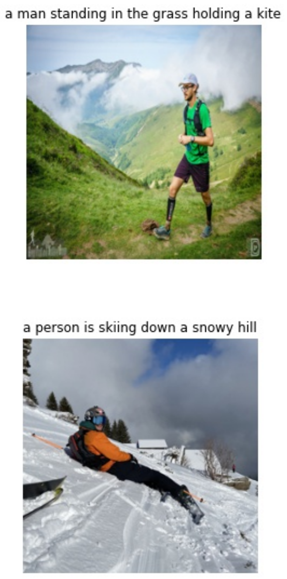
a man standing in the grass holding a kite
a person is skiing down a snowy hill

Figure 5: At the top, the caption generated on Etienne's Facebook profile picture and at the bottom, on Charles' one