Image Captioning with Keras

Salvatore Arienzo
Internet of Things Master Degree Student
Department of Computer Science
University of Salerno
Email: s.arienzo@studenti.unisa.it

December 12, 2019

Abstract

Image Captioning is the automatic generation of natural sentences that describe the contents of a given image. In order to realize this task, has been used Keras with TensorFlow as backend. The data used for the project derives from the Flickr Image dataset, an open dataset obtainable online. Two types of Neural Network has been used: a Convolutional Neural Network to extract features from the images, then a Recurrent Neural Network as a decoder. The achieved results are not perfect, but quite acceptable, the generated captions are always related to the given image, but sometimes with some mistakes. To evaluate the accuracy of the model the BLEU (Bilingual evaluation understudy) metrics and a custom metric have been used

1 Introduction

Given an image, it's easy for us, as humans, to give a glance to the picture and describe it in our language. But, what about computers? Is it anyway easy? The caption generation - an auto-generated description of the given image - is a challenging artificial intelligence problem where a textual description must be generated for a given picture. It requires a computer vision approach to understand the content of the image and a a language model, in order to let the computer describe the picture with natural language with a strong semantic understanding of the visual scene and all of its elements. The image captioning task can be an interesting task in many fiedls, like some applications blind-fiendly: we can create a product, an application, an envoirement that can help them and improve their life quality. An example can be Nvidia that is working on a product that can convert a scene into text and describe this text with voice in otder to guide blind people during walks, or travels without the support so anyone else[1]. Image captions are useful also for security cameras: if cameras can also generate relevant captions, we can raise alarms as soon as there is some malicius activity going on someghere, and so it can help to reduce some crime or accident. Also self driving cars, one of the biggest challenges in artificial intelligence can benefit from image captions: if we can properly and in real-time generate captions of the scene around the car, it can give a boost to

the serlf driving system. This paper is composed by four chapters, the first one is a short introduction about image captionin, the second chapter will presents some related works, the third chapter will explain all the process used in order to achieve our goal. In the last chapter the results will be shown e with some examples of captions evaluated by an evaluation metric.

2 Related Work

This task has been well researched by the actual Director of AI at Tesla: Andrej Karapathy, that exposed his work in DenseCap: Fully Convolutional Localization Networks for Dense Captioning[2] and Deep Visual-Semantic Alignments for Generating Image Descriptions[3]. Some other useful things to know about Image Captioning have been retreived on websites like Machinelearningmastery and medium and are referred in [4],[5],[6]. While in [7] there is an explaination of how to deal with BLEU Scores.

3 Model

To achieve our goal, we basically need: data, vector of words, a Convolutional Neural Network to encode our images: a convolutional neural network is a neural network that that uses some mathematical operations called "Convolutions" instead of general matrix multiplication in at least one layer. It consists of an input and output layers with multiple hidden layers, the hidden layers are a series of convonutional layers that "convolve" with a multiplication. And finally a Recurrent neural network as decoder: a recurrent neural network is a neural network in which connections between nodes have also a temporal sequence, this allows it to provide a temporal dynamic behavior.

3.1 Dataset Used

For this project has been used the Flickr Image dataset, in both 8k and 30k version. This dataset has become a standard benchmark for sentence-based image description. The dataset contain a set of images representing a person, people, or animals while doing normal actions. Each image is labeled with five different captions for a total of 158k captions. In order to have as much labeled examples as possible the dataset has been divided in 29k for the training set, and 1k for the test set.

3.2 Data Preparation

The first thing to do while dealing with a machine learning / deep learning project, is to understand and prepare data for our goals. In our case of study with the Flickr dataset, has been provided a list of five captions for each image. These captions, are stored i a token file with the format imagename + .jpg + # n + Caption.

```
1000092795.jpg#1        Two young , White males are outside near many bushes .
1000092795.jpg#2        Two men in green shirts are standing in a yard .
1000092795.jpg#3        A man in a blue shirt standing in a garden .
```

Now, we need to split this file in more parts: a txt file that contains all the pairs imagename + caption (witouth extension and special characters).

```
1000092795 two young white males are outside near many bushes
1000092795 two men in green shirts are standing in yard
1000092795 man in blue shirt standing in garden
```

With this file we create a sort of dictionary of all the captions, now querying the name of an image we will have all its captions as response.
In order to let the text more understendable, we should "clean" this descriptions removing

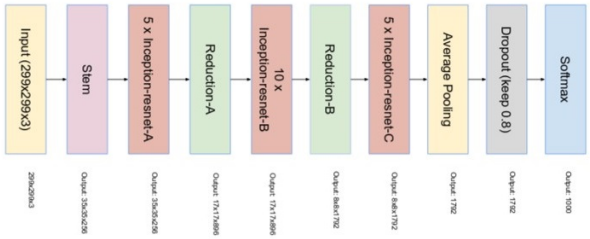
```
descriptions['1000092795']
```

```
['Two young guys with shaggy hair look at their hands while hanging out in
the yard .',
 'Two young , White males are outside near many bushes .',
 'Two men in green shirts are standing in a yard .',
 'A man in a blue shirt standing in a garden .',
 'Two friends enjoy time spent together .']
```

the punctuation, special characters, transforming all the text in lowercase letters and eliminating all the words containing numbers, for instance "Hello!" became "hello", otherwise, Hello and hello will be classified as two different words. Counting all the words generated, we have a voucabulary of 19712 different words. Now, we need to split our dataset, in training and test set by creating two more text files with just image names, the trainset will contain 29k images and the test set 1k images. Now our data is properly cleaned and splitted.

3.3 Inception Model

Now, we need to encode all our images in a fixed size (in our case, 299x299), and to give them in input to a convolutional neural network. I've used the Inception Model, a pre-trained network that due to its low error score and efficency has become one of the most used networks for image classification. The inception model is trained on the ImageNet dataset, with over 15 millions labeled-high resolution images.There are four versions of Inception, for this project i've tryed the InceptionV3 Model and the InceptionResNetV2 and finally used the second one. The structure of the InceptionResNetV2 is described by the image below. It takes as input images of 299x299x3 and give us as result a 2048 lenght vector, cause we remove the last softmax layer cause it performs a 1000-class classification, that we dont need.



3.4 Word Embeddings

We need to encode each word into a fixed sized vector, this process is called Word Embeddings. The word embeddings are the text converted into numbers and there may be different numerical representations of the same text. So we need to take our dictionary, that looks like ['hello', 'world'], and convert each word in a 200-vector. A vector is a one-hot encoded vector where 1 stands for the position where the word exist and 0 everywhere else. After some attempts with some handmaded vectors, i've used two pre trained models: GLOVE Model with 400000 wird vectors and word2vec with more than 4millions vectors. This models provide a better accuracy than others because are trained on thousand of wikipedia text. So, using GLOVE vectors we create an embedding matrix of (7640, 200) (vocabulary size, vector dimension) what will be loaded into the model before training.

3.5 RNN Model

In this case, the recurrent neural network is used as a text generation model. Our model takes as input, both the images vectors and the partial captions (input layer 1 and 2), than in the embedding layer, every index of the captions is mapped to a 200 dimensional vector. Then we have two dropout layers, to try to avoid over-

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 74)	0	
input_2 (InputLayer)	(None, 2048)	0	
embedding_1 (Embedding)	(None, 74, 200)	1528000	input_3[0][0]
dropout_1 (Dropout)	(None, 2048)	0	input_2[0][0]
dropout_2 (Dropout)	(None, 74, 200)	0	embedding_1[0][0]
dense_1 (Dense)	(None, 256)	524544	dropout_1[0][0]
lstm_1 (LSTM)	(None, 256)	467968	dropout_2[0][0]
add_1 (Add)	(None, 256)	0	dense_1[0][0] lstm_1[0][0]
dense_2 (Dense)	(None, 256)	65792	add_1[0][0]
dense_3 (Dense)	(None, 7640)	1963480	dense_2[0][0]

Total params: 4,549,784
Trainable params: 4,549,784
Non-trainable params: 0

fitting. Then we add some Dense (for image model) and LSTM (For caption model) LSTM, Long Short Term Memory, is a specialized Recurrent Neural Network that processes not only single data point, like images, but also entire sequences of data(like speech or text). The final output layer, that uses a softmax activation, will give us the probability distribution across the 7640 words of the vocabulary.

3.6 Evaluation

The image captioning task is so time expensive: the encoding of images took around 4 hours for each train set and test set with the InceptionV3 Model, while with the InceptionV2ResNet it took around 8 hours. The training of the recurrent neural network, took more than 2 hours for each epoch on a machine with an Intel Xeon E5 and 128GB of RAM (on the 30k Flickr dataset). The final loss was around 3.1, but it was still slowly improving. To build the final predicted caption, has been used a greedySearch algorithm: since the caption is generated iteratively, one word for each iteration, the greedy algorithm choose greedly: that mean that for each iteration it choose the word with the maximum probabily in that moment. Now, we need a metric to evaluate how good the model is, we can use the BLEU Scores (Bilingual evaluation understudy) propsed by Kishore Papineni [9]: is a score for comparing a candidate translation of text to a reference text. Has been developed for translations, but it can be used to evaluate auto generated text. Using this metric a perfect match, will score 1.0, while a perfect mismatch results 0.0. How it works: our caption is used as "candidate", all the words will be splitted and placed in an array, for instance ['this','is','a','caption] and compared to a "reference" caption of that image, for instance if my reference is ['this','is','a','caption], i will have a score of 1. Of course, beeing our captions completely auto generated, is non possible to have a perfect, or even high score, because this should mean that our completely automatically generated caption should be exactly (or almost) identical to the reference caption, so, using this metric, we will have a very low score also in captions that are correct. So, to try to evaluate better our model we have used a custom metric: the BLEU Score try to match the words in the auto- generated caption, with the words in the reference caption, but it gives a negative value also if the word is present, but in a different position: as result we have good caption evaluated bad just because the words are sorted in a different way. So, since we can consider "Accepted" also a caption that identify just the subject or the action happening in the image, in our evaluation model we consider accepted the captions with one or two words in common with the reference: having a score of 73% for 1 word, and 48% for two words matching.

4 Conclusions

In conclusion, this model, using the Inception-ResNetV2, and a 30k images dataset perform the image captionin task quite well, but since no model is perfect, basing on the Convolutional neural network used, the captions style changes: for instance, InceptionV3 model, recognize animals very well, while has many problems with people. The InceptionResNetV2, recognize people much better but rarely has some mistakes on animals, an both networks have problems recognizing colors. The captions depents also on the dataset, since the Flickr dataset has a lot of images about dogs, the dog's captions are almost perfect. Finally, some example of both correct and wrong captions.

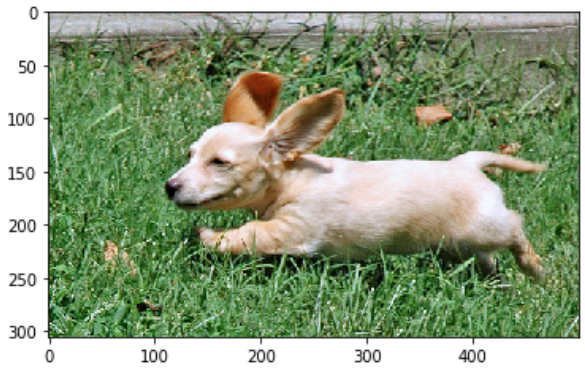


Image: dog is running through the grass

Image: man is doing trick on his bicycle

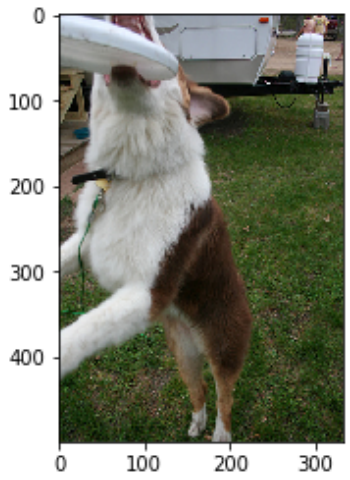


Image: dog is jumping up to catch frisbee

Image: man in white shirt is cooking food

Image: group of people are sitting at table eating food

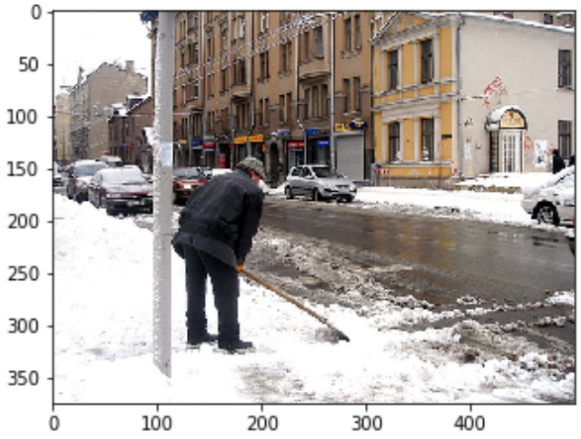


Image: man shovels snow off of the road

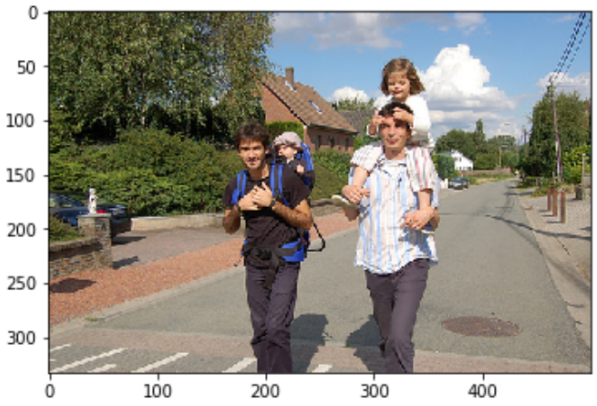


Image: two men are walking down the street

Image: man in blue shirt is driving tractor

References

[1] Horus by Nvidia - https://blogs.nvidia.com/blog/2016/10/27/wearable-device-for-blind-visually-impaired/ [2] Justin Johnson, Andrej Karpathy, Li Fei-Fei - DenseCap: Fully Convolutional Localization Networks for Dense Captioning [3] Andrej Karpathy Li Fei-Fei - Deep Visual-Semantic Alignments for Generating Image Descriptions [4] How to Prepare a Photo Caption Dataset for Training a Deep Learning Model- https://machinelearningmastery.com/prepare-photo-caption-dataset-training-deep-learning-model/ [5] How to Develop a Deep Learning Photo Caption Generator from Scratch - https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/ [6] InceptionV3 Review - https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c [7] Introduction to BLEU Score Text - https://machinelearningmastery.com/calculate-bleu-score-for-text-python/
[8] Flickr Image Dataset - https://www.kaggle.com/hsankesara/flickr-image-dataset [9] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu - BLEU: a Method for Automatic Evaluation of Machine Translation