

Project 5 Summary

“Darling, what am I looking at?”

(Automatic Image Text-to-Speech Captioning for the Visually Impaired via Deep Learning)

Charlie Lew

December 10th, 2019

Abstract

This project combines the world of Computer Vision and Natural Language Processing where the model not only predicts and prints out a sentenced caption of a photograph but also audio. The application is targeted towards those with some form of visual impairment. A Convolutional Neural Network is used for feature extraction of an image. To predict and generate a caption, the image features and tokenized vocabulary are fed into a Recurrent Neural Network with a Long Short-Term Memory layer. The predicted captions in some cases are generally in good agreement with an already annotated image dataset though not perfect. The image caption generator is wrapped within a Flask web application. Future work involves improving the Recurrent Neural Network, obtaining a larger dataset and potentially extending this application to paintings or even videos.

Introduction & Motivation

In the summer of 2019, I took a trip to Spain, France and Italy. I was there for almost two months, traveling light, eating fantastic food and took literally a thousand photos on my iPhone. During my travels, I would send my parents photos. One of them in particular is shown in Figure 1 below. When I was on the phone with my mother, she would ask, “Darling, what am I looking at? It looks like a really long building with lots of windows.” I would respond to her, “No Ma, it is the famous 2,000-year-old Roman Aqueduct in Segovia. Remember you mentioned that I should go and visit that take a picture of it for you?” In any case, although my lovely mother will not admit it, she has a slight vision impairment. Hence, I wondered to myself, wouldn’t it be nice if she had a web application where she can drag-and-drop a photo in a web browser, and not only will it caption it for her in large font but also read it out to her?

I took a look around for caption generators with sound that is specifically catered towards those that are visually impaired. Microsoft has Captionbot.ai which is more of a fun tool to play with but does not have audio, and my mother will have a hard time using it. The other web applications are from Instagram and Google Vision but are geared towards a more social media savvy crowd and not necessarily for my mother. So, I took the initiative to create

my own that would hopefully make not only my mother proud but perhaps useful for millions of other around the world who visually impaired.



Figure 1: The Roman Aqueduct in Segovia, Spain

Data, Tools & Methodology

The generation of text from an image is in the realm of Deep Learning (DL) and thus this will be the main tool that is utilized. In order to train a deep learning model, it requires quite a substantial amount of data to train. This data has to contain both images and annotated texts. For this, I used Flickr's 8,000 image dataset where all are fully annotated by a Mechanical Turk. Eight thousand images are by no means a large dataset but perhaps good enough to start this endeavor. An assortment of Python libraries is used. These include, Matplotlib, NLTK, Seaborn and Scikit-Learn. Since deep learning is involved, TensorFlow and Keras are utilized. Finally, Flask is used to create a front-end web application where a user such as my mother could use to caption a photograph. Figure 2 below shows a high-level overview of the captioning process.

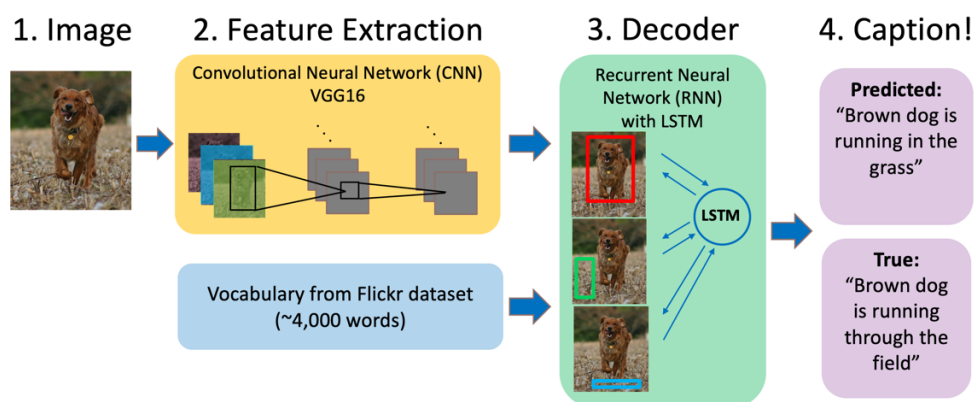


Figure 2: Methodology pipeline for automatic image captioning

The dataset was split as 60-20-20 style for training, validation and testing. The methodology is as follows for an image in a test set. First, the example image of a dog is fed to a feature extraction model which is based on a Convolutional Neural Network (CNN). The specific model used is called VGG16 which is 16-layer CNN from Oxford University's Visual Geometry Group. Other models tried were InceptionV3 and Xception but VGG16 appears to be sufficient. In this phase, no training is done but instead, a method called Transfer Learning is implemented. Here the feature extractor layer is used with the weights obtained from ImageNet. At the same time, a tokenized vocabulary of words is created from the annotated training dataset. The third and final step involves feeding the vocabulary and image features to an Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) layer to finally create and predict a caption. Figure 3 below shows the RNN model used. Note that in addition to the LSTM layer, dropout layers with a weight of 0.5 is used to avoid over-fitting.

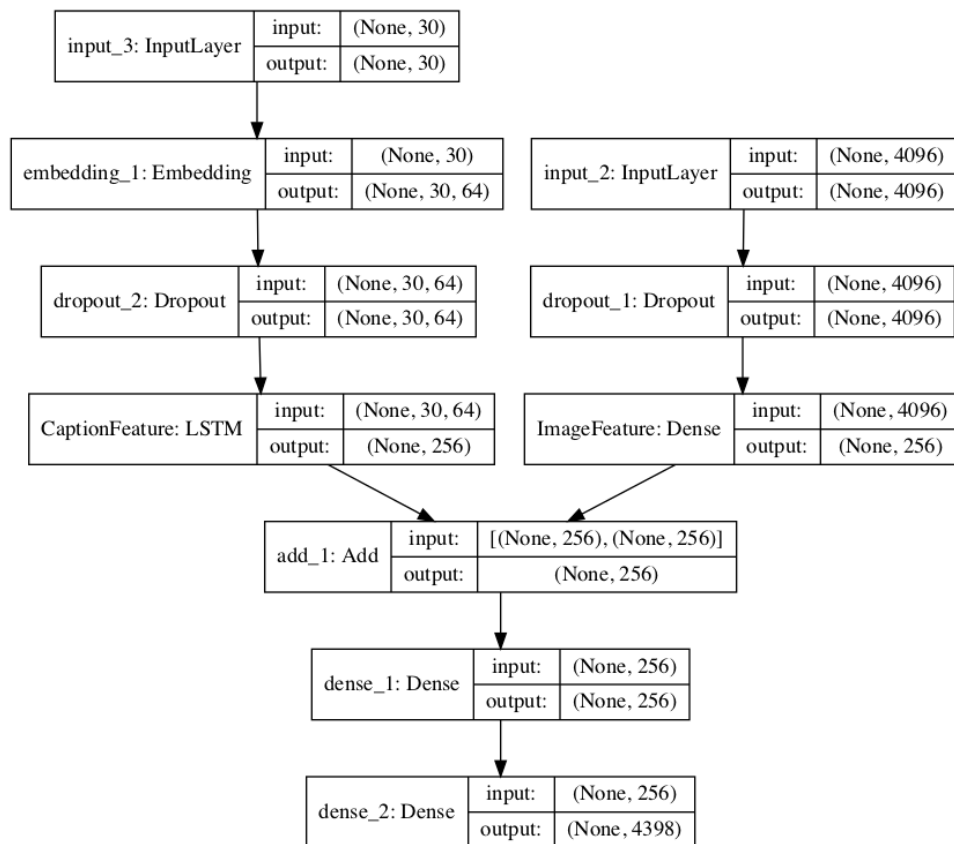


Figure 3: Recurrent Neural Network Model with LSTM and Dropout layers

In this portion, the combination of the RNN and LSTM is to look at what image features it thinks it sees and look at the vocabulary of words. If the probability is high that it sees a 'dog', an 'action' and 'grass', it will then create a meaningful sentence that is human readable. In the above example, the predicted caption is "Brown dog is running through the grass" whereas the annotated text that came along with the dataset has it as "Brown dog is running through the field". Here both the predicted and annotated 'ground truth' are acceptable but with slight

nuances. Deep learning is a stochastic process and thus one may not be able predict an image with 100% match word-for-word. A few more details on the RNN-LSTM model. The cost-function, which is a loss function, is based on categorical cross-entropy with Adam being used as the optimizer. The model training took approximately over three minutes. A validation dataset was used to determine the optimum number of epochs to train the model. Figure 4 below shows that the optimum number of epochs is five based on the lowest validation loss and no further training is required.

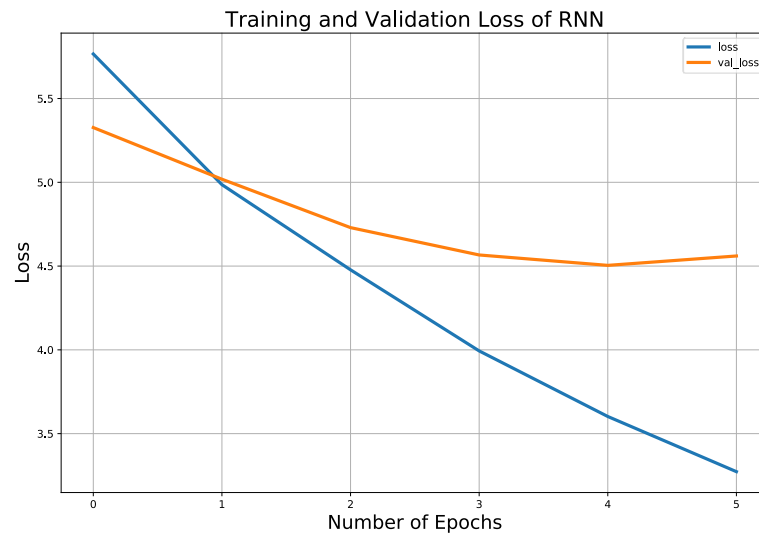


Figure 4: Loss and validation loss as a function of number of epochs. Here total number is 5.

Flask Web Application

In order to have the above application be accessible to a user, a Flask web application was created. Figure 5 below shows the application in action for an image of a surfer.

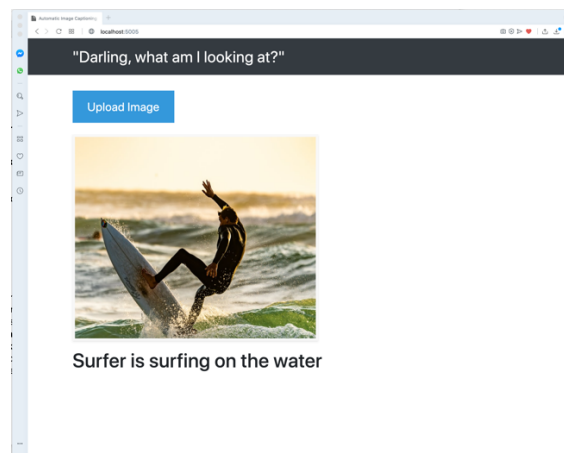


Figure 5: Front end Flask web application for image captioning

Here is the result with audio captions, “Surfer is surfing on the water” which is acceptable in simplistic terms. This is slightly conservative but safe, nonetheless. Figure 6 below shows the outcome for an image of a rider on a motorcycle and finally on my aqueduct photo.



Image	Generated Caption
	<i>Man in red helmet is riding his motorcycle</i>
	<i>Man is standing on the background of</i>

Figure 6: Two more image test captions

The motorcycle picture is captioned quite well although not perfect since the helmet is not entirely red. Finally, I tried the aqueduct photo and unfortunately it did not make sense. It does see some people in streets but improperly assumes a man. It cannot determine what background that is only because there are no aqueducts in my training set which is understandable. A future version of the dataset will have to include aqueducts, bridges and perhaps the word Spain in it.

Closing Remarks & Future Work

In summary, this project has combined the world of Computer Vision (CV) and Natural Language Processing (NLP) to create a sentenced caption of an image. Not only is the task to create a visual text caption but read and voice the caption a person is visually impaired. A Convolutional Neural Network (CNN) with Recurrent Neural Network (RNN) was used to generate the caption. The entire pipeline was then wrapped in a Flask web application. The

known limitations are that there is not enough training data, and in my case, the data is missing an aqueduct feature. As mentioned, 8,000 images is a small dataset. Flickr has an even larger dataset with 30,000 images, Microsoft's COCO has over 100,000 and finally, Google has over a million. The best step is to make sure that the RNN is properly tuned via Hyper-parameter testing, adding an Attention Model and adding a metric feature based on Bi-lingual Evaluation Understudy (BLEU) which compares two sentences, in this case the predicted and actual annotation and give it a score. A score of over 0.7 is considered good whereas anything below it is considered poor. By using the BLEU score, one can actually measure to some degree performance of the caption generator. Finally, other potential uses of a caption generator for the visually impaired could be for paintings or even videos.