# CAPTION GENERATOR

#### Deep Learning Model to Automatically Describe Photographs

**Team Members:**

Sahithi Reddy Paspuleti

Mugdha Bajjuri

Sushmitha Boddireddy

Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph.

Caption generation requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order.

**Objective:**

Aim of this project is to develop a deep learning model to generate a caption, given an image.

## Photo and Caption Dataset:

We found that, a good dataset to use when getting started with image captioning is the Flickr8K dataset. The reason is because it is realistic and relatively small so that you can download it and build models on our workstation using a CPU.

**Flickr8k\_Dataset**: Contains 8000 photographs in JPEG format that are each paired with five different captions which provide clear descriptions of the salient entities and events. S

**Flickr8k\_text**: Contains a number of files containing different sources of descriptions for the photographs.

**Text Data preparation**

The Flickr dataset contains multiple descriptions for each photograph, as we cannot go straight from raw text to fitting a machine learning or deep learning model, we started with cleansing the text data first.

For data preparation, we started with

* splitting lines by whitespace into tokens and took first token as the image id and the rest as the description.
* Next, we proceeded with converting words to lowercase
* Then, removed all punctuations
* Then, removed all the words that are 1 character or less in length
* Finally, removed all the words with numbers in them.
* Later we observed that there are a few common words used in the descriptions like “an, the, is so on” which don’t count towards any meaning to the sentence and so doesn’t help in training the model. So, we decided on removing those for which we used stopwords functionality by NLTK. NLTK provides a list of commonly agreed stop words for a variety of languages.

**Photo Data preparation**

We have used a pre-trained model VGG to interpret the content of the photos. VGGNet is a neural network that performed very well in the [Image Net Large Scale Visual Recognition Challenge](http://www.image-net.org/challenges/LSVRC/2014/) (ILSVRC) in 2014. It scored first place on the image localization task and second place on the image classification task.

Localization is finding where in the image a certain object is, described by a bounding box. Classification is describing what the object in the image is. This predicts a category label, such as “cat” or “bookcase”. we have pre-computed the “photo features” using the VGG model and saved them to a file. In this way we optimized the model by making it faster and consume less memory. We loaded the VGG model in Keras using the VGG class. We removed the last layer from the loaded model, as this is the model used to predict a classification for a photo. We are not interested in classifying images, but we are interested in the internal representation of the photo right before a classification is made. These are the “features” that the model has extracted from the photo. As the VGG takes input image of size 224 \* 224 \* 3 (RGB image) we have used keras reshape method to convert the image into preferred shape. Finally, we arrived at the image features that are a 1-dimensional 4,096 element vector.

**Modeling**

We will be describing the model in three parts:

* VGG model used as a photo feature extractor.
* Word embeddings: Word2Vec, a statistical method that allows words with similar meaning to have a similar representation. (yet to implement)
* Long Short-Term Memory (LSTM) recurrent neural network layer used as an encoder to encode linguistic feature. (yet to implement)

**Ongoing Issues**

* So far, we haven’t faced noticeable problem using the 64-bit CPU. Of course, we wish to reduce the ETA for which we are planning to use Google colab. Colab notebooks execute code on Google’s cloud servers (GPU or TPU) with which we can reduce the execution time.
* As we are working as a team, we were facing issues in merging the individual parts of code written by each of us. To avoid this issue, we have decided on using GitHub.