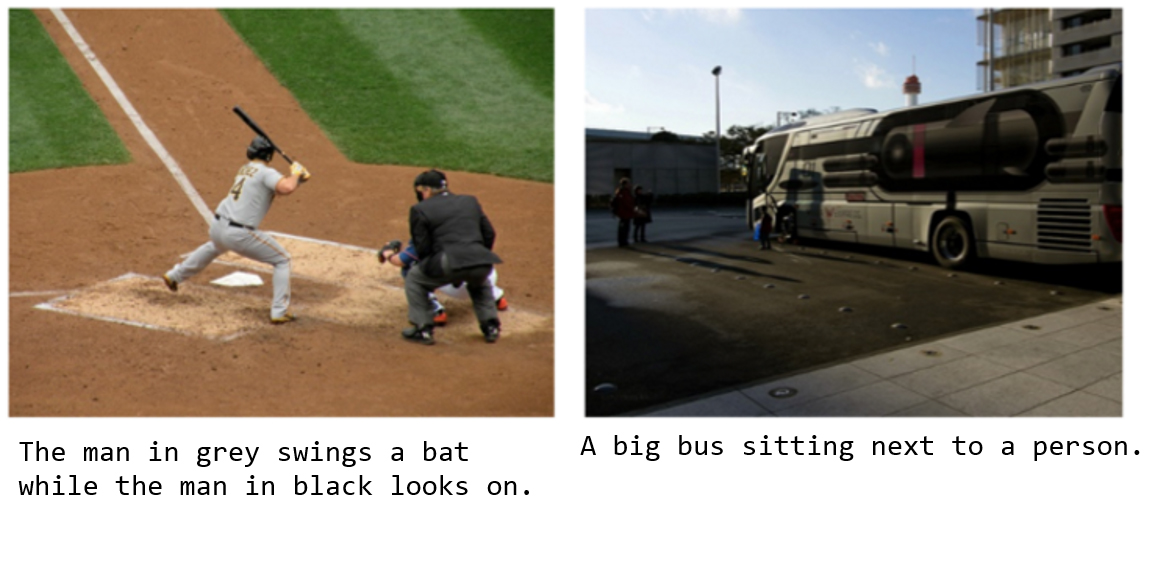
**Image Explanation Generator**

By: Shaksham Sharma

Cse 2 6th sem

# Image Explanation Generator



© O'Reilly

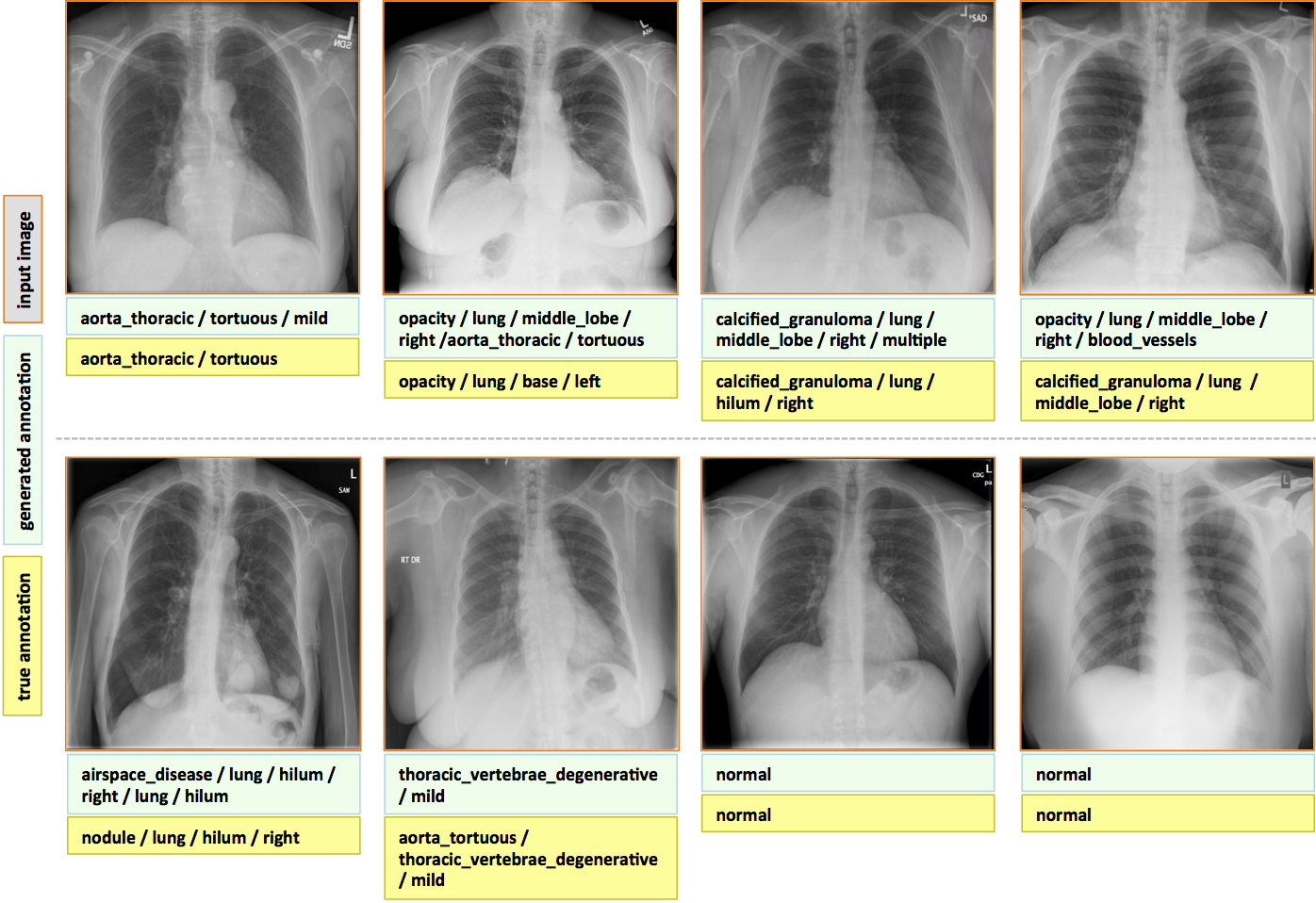
\* This is implementation of a image caption generator from [Yumi's Blog](https://fairyonice.github.io/Develop\_an\_image\_captioning\_deep\_learning\_model\_using\_Flickr\_8K\_data.html). which generates a caption based on the things that are present in the image. Image captioning is a challenging task where computer vision and natural language processing both play a part to generate captions. This technology can be used in many new fields like helping visually impaired, medical image analysis, geospatial image analysis etc.

## Use cases

\* Some detailed usecases would be like an visually impaired person taking a picture from his phone and then the caption generator will turn the caption to speech for him to understand.

\* Advertising industry trying the generate captions automatically without the need to make them seperately during production and sales.

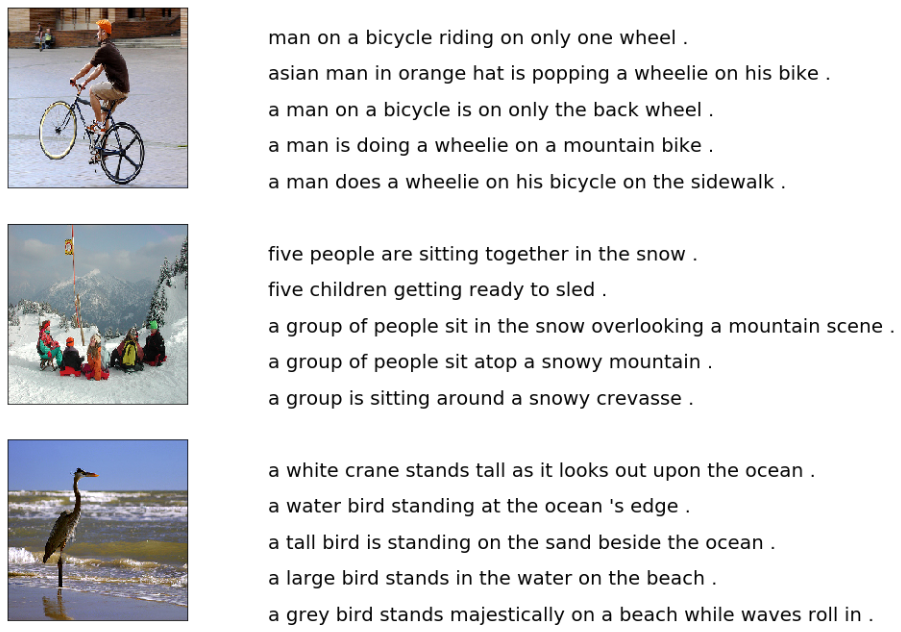
\* Doctors can use this technology to find tumors or some defects in the images or used by people for understanding geospatial images where they can find out more details about the terrain.



## Dataset:

[FLICKR\_8K](https://forms.illinois.edu/sec/1713398).

This dataset includes around 1500 images along with 5 different captions written by different people for each image. The images are all contained together while caption text file has captions along with the image number appended to it. The zip file is approximately over 1 GB in size.



## Flow of the project

#### a. Cleaning the caption data

#### b. Extracting features from images using VGG-16

#### c. Merging the captions and images

#### d. Building LSTM model for training

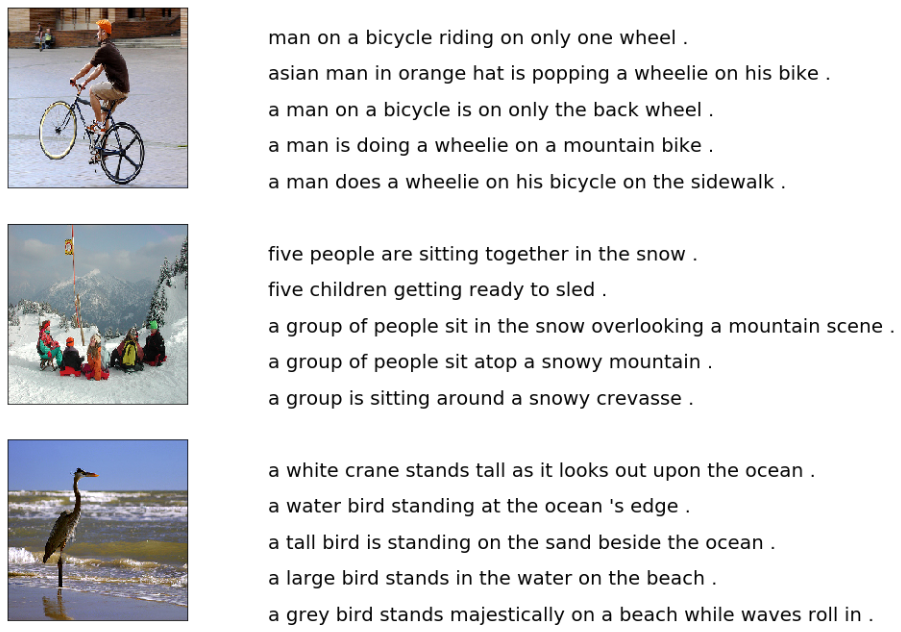
#### e. Predicting on test data

#### f. Evaluating the captions using BLEU scores as the metric

## Steps to follow:

### 1. Cleaning the captions

This is the first step of data pre-processing. The captions contain regular expressions, numbers and other stop words which need to be cleaned before they are fed to the model for further training. The cleaning part involves removing punctuations, single character and numerical values. After cleaning we try to figure out the top 50 and least 50 words in our dataset.



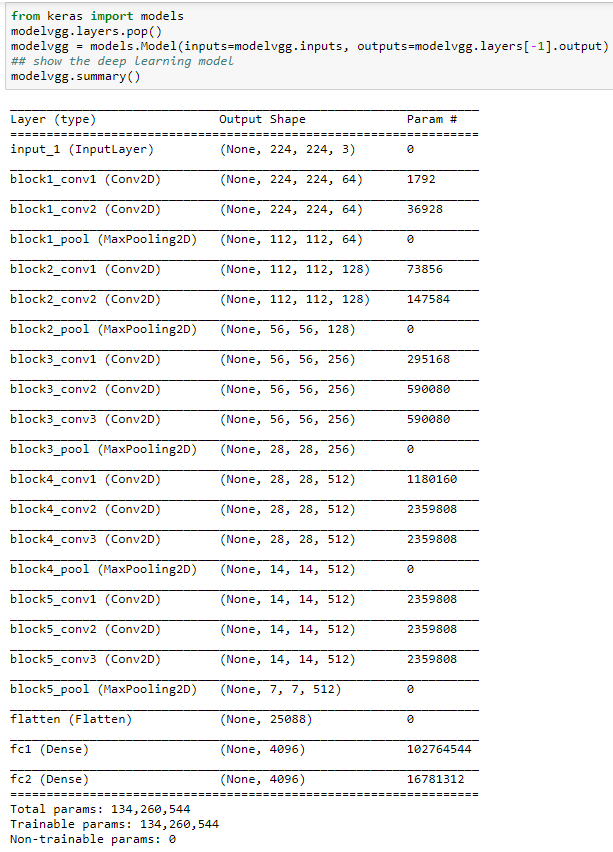
### 2. Adding start and end sequence to the captions

Start and end sequence need to be added to the captions because the captions vary in length for each image and the model has to understand the start and the end.

### 3. Extracting features from images

\* After dealing with the captions we then go ahead with processing the images. For this we make use of the pre-trained [VGG-16](https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16\_weights\_tf\_dim\_ordering\_tf\_kernels.h5) weights.

\* Instead of using this pre-trained model for image classification as it was intended to be used. We just use it for extracting the features from the images. In order to do that we need to get rid of the last output layer from the model. The model then generates \*\*4096\*\* features from taking images of size (224,224,3).



### 4. Viewing similar images

When the VGG-16 model finishes extracting features from all the images from the dataset, similar images from the clusters are displayed together to see if the VGG-16 model has extracted the features correctly and we are able to see them together.



### 5. Merging the caption with the respective images

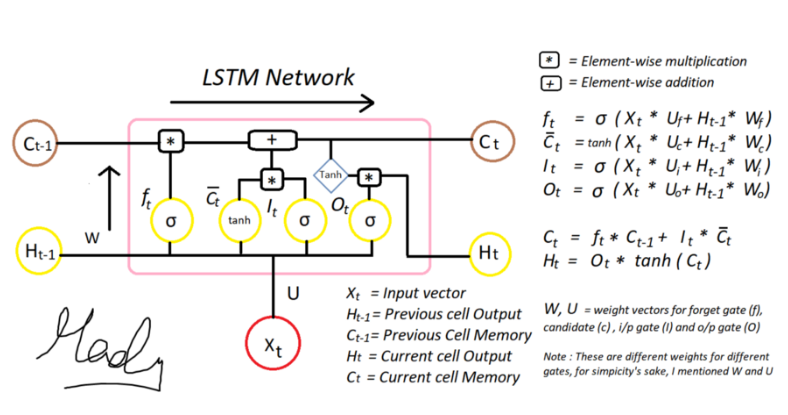
\* The next step involves merging the captions with the respective images so that they can be used for training. Here we are only taking the first caption of each image from the dataset as it becomes complicated to train with all 5 of them.

\* Then we have to tokenize all the captions before feeding it to the model.

### 6. Splitting the data for training and testing

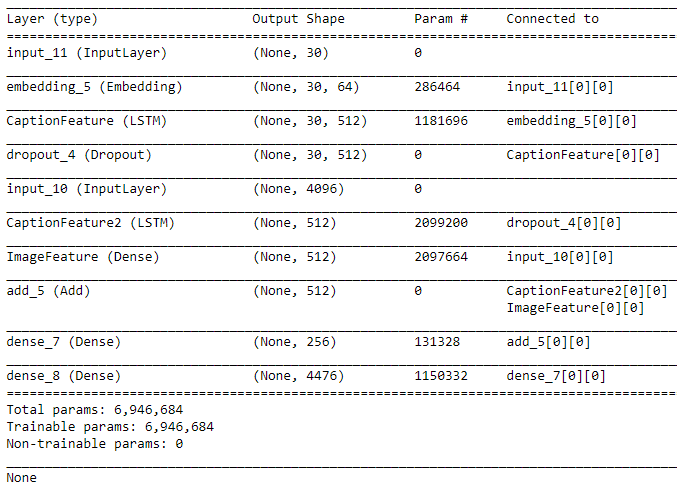
The tokenized captions along with the image data are split into training, test and validation sets as required and are then pre-processed as required for the input for the model.

### 7. Building the LSTM model



LSTM model is been used beacuse it takes into consideration the state of the previous cell's output and the present cell's input for the current output. This is useful while generating the captions for the images.<br>

The step involves building the LSTM model with two or three input layers and one output layer where the captions are generated. The model can be trained with various number of nodes and layers. We start with 256 and try out with 512 and 1024. Various hyperparameters are used to tune the model to generate acceptable captions



### 8. Predicting on the test dataset and evaluating using BLEU scores

After the model is trained, it is tested on test dataset to see how it performs on caption generation for just 5 images. If the captions are acceptable then captions are generated for the whole test data.

![](images/pred2.PNG?raw=true)

These generated captions are compared to the actual captions from the dataset and evaluated using [BLEU](https://machinelearningmastery.com/calculate-bleu-score-for-text-python) scores as the evaluation metrics. A score closer to 1 indicates that the predicted and actual captions are very similar. As the scores are calculated for the whole test data, we get a mean value which includes good and not so good captions. Some of the examples can be seen below:

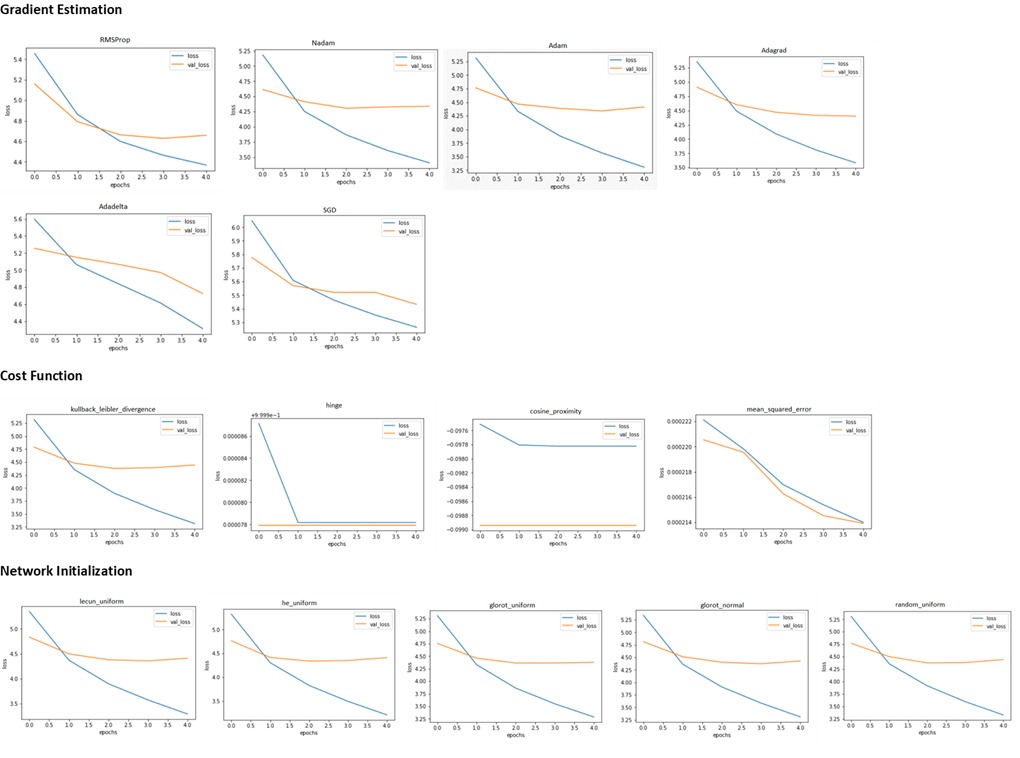
#### Good Captions



#### Bad Captions



### Hyper-parameter tuning for the model







## Conclusion

Implementing the model is a time consuming task as it involved lot of testing with different hyperparameters to generate better captions. The model generates good captions for the provided image but it can always be improved.