A Report

On

**Image Caption Bot**

**BY**

**APOORV**

**ORGANIZATION INTRODUCTION**

**Section 1 – Introduction of Project – “Image Caption Bot”:**

In this section I have introduced the topic and the technology used in our project like OpenCV DNN, TensorFlow, Keras and resent architecture which is used as an image classifier. This model works very well while generating the captions for the images.

**Section 2 – work carried out :**

In this section I have discussed the complete methodology adopted and technology used by me to build this Image Caption Bot model. I have also explained everything about the dataset in the given section.

**Section 3– Project Results obtained :**

In this section I have shown my experimental results.

**Section 4– Conclusion and Future scope:**

In this section I have concluded my project with the future scope.

# 

# 

# 

# **Chapter 1**

**Introduction**

From the starting moment of the day when we open our eyes in the morning our brain starts collecting data from different sources.To keep up with this growth of data from different sources mankind was introduced with modern Data Driven Technologies like Artificial intelligence, Machine Learning, Deep Learning etc. These technologies have engineered our society in many aspects already and will continue to do so.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.Our human brain can easily differentiate between a cat and a dog. But how can we make a machine differentiate between a cat and a dog?

We have to train the machine similarly with a lot of images.What if the machine starts predicting for an image what is happening in the image it could be a great application.

What do you see in the below picture?



Fig 1 Intercity train passing the steel bridge over river

(source:https://www.google.com/imgres?imgurl=https%3A%2F%2Fpreviews.123rf.com%2Fimages%2Fremik44992%2Fremik449920909%2Fremik44992090900005%2F5463601-intercity-train-passing-the-steel-bridge-over-the-river.jpg&imgrefurl=https%3A%2F%2Fwww.123rf.com%2Fphoto\_5463601\_intercity-train-passing-the-steel-bridge-over-the-river.html&tbnid=NCFfmrCER5CdrM&vet=12ahUKEwi1nuy22prsAhXMkuYKHdZ-AO8QMygAegUIARCqAQ..i&docid=3HpJoMTzwi2SBM&w=1300&h=866&q=train%20passing%20bridge&ved=2ahUKEwi1nuy22prsAhXMkuYKHdZ-AO8QMygAegUIARCqAQ)

Well some of you might say “**A Train on a bridge**”, some may say “**Bridge on river**” and yet some others might say “**Plants near the river** ”Definitely all of these captions are relevant for this image and there may be some others also. But the point I want to make is; it’s so easy for us, as human beings, to just have a glance at a picture and describe it in an appropriate language. Even a 5 year old could do this with utmost ease. But, can you write a computer program that takes an image as input and produces a relevant caption as output?

Just prior to the recent development of Deep Neural Networks this problem was inconceivable even by the most advanced researchers in Computer Vision. But with the advent of Deep Learning this problem can be solved if we have the required dataset. This project is an Image caption generator.

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

The application of image caption is extensive and significant, for example, the realization of human-computer interaction The development of the image description system may help the visually impaired people to see the world in the future. We can create a product for the blind which will guide them travelling on the roads without the support of anyone else. We can do this by first converting the scene into text and then the text to voice. Both are now famous applications of Deep Learning. This can also be used in image search from the google .Automatic Captioning can help, make Google Image Search as good as Google Search, as then every image could be first converted into a caption and then search can be performed based on the caption This can also be used in the self driving cars. In the self driving cars we can use this Image caption bot which we detect the obstacle encountered in the path of self driving car.CCTV cameras are everywhere today, but along with viewing the world, if we can also generate relevant captions, then we can raise alarms as soon as there is some malicious activity going on somewhere. This could probably help reduce some crime and/or accidents. These applications are itself great motivation to work on this project. The image description is obtained by predicting the most likely nouns, verbs, scenes, and prepositions that make up the sentence.

This project involves data collection in the form of text and images than cleaning of the text data using the natural processing techniques. Flickr 8k dataset is collected from Kaggle containing 6000 images for training 1000 images for validation 1000 images for testing. A single image contain 5 captions. After cleaning of the data both image and text data is separately pre-processed .The image data is pre-processed using the very famous resnet50 Model of Convolutional Neural Network .

The Textual data is firstly cleaned and then a dictionary is created for the unique words. All those words having frequency less than 10 are rejected .Here 10 is the threshold frequency in my project .Then captions are appended with two special key words startseq and endsseq This is done so that our model get trained from where the sentence is starting and where the sentence is ending .After this two dictionaries are created word\_to\_num and num\_to\_word where we are converting our data into numbers.That is every word of the vocab will point to each unique number.Now before feeding this data data is passed from the pretrained embedding layer know as gloved vector which contain the vector representation for six billion words. Then matrix is formed in which each word of vocab is appended corresponding to their 50 dimensional vector. After the pre-processing of both the textual and image data Data generator function is used for the training in order to make small batches. Data generator function is generally used when the work is carried on the large dataset Data generator function is used .A machine can not generate the entire sentence ever so this problem is solved in the form **supervised machine learning problem**. Where every image corresponds to a group of words in which a single word is generated using the previous set of words already predicted. This is like we check the ground truth while training .we find joint Probability for the given sentences. This concept is also called as Language Modelling .

P(w(t+1)/w(t-1)\*w(t-2)\*..........\*w(1)\*w(0))

That is like if we want to predict “dog is running” than generation of running depends on dog and that depends on the previous predicted sentence which is dog and that of dog from startseq. After this, Since the input consists of two parts, an image vector and a partial caption, we cannot use the Sequential API provided by the Keras library. For this reason, we use the Functional API which allows us to create Merge Models.In the coming section I will be discussing the complete methodology and results obtained by this project.

# 

# 

# 

# **Chapter 2**

**Work Carried out**

**3.1 Dataset collection:**

There are many open source datasets available for this problem, like Flickr 8k (containing 8k images), Flickr 30k (containing 30k images), MS COCO (containing 180k images), etc. For this problem I have used Flickr 8k dataset due to unavailability of high power GPU .This dataset is provided by the University of Illinois at Urbana-Champaign.Also training a model with large number of images may not be feasible on a system which is not a very high end PC/Laptop.This is dataset is quite feasible to work on .Dataset is approximately around 1.14GB.

**3.2 Understanding the data:**

The dataset contains three folders names as images flickrtextdata and flickr8k textfiles. FlickrTextdata has two files for training and testing in which the image id for each image is written.This dataset contains 8000 images each with 5 captions as an image can have multiple captions, all being relevant simultaneously.This dataset is further divided into three parts.Training set has 6000 images,validation and test set has 1000 images simultaneously.One of the files is “Flickr 8k.token.txt” which contains the name of each image along with its 5 captions.

**3.3 Data Cleaning:**

First I have stored data in the dictionary named as description in which key is imageid and corresponding to it i have stored the list of captions which were relevant to that image.When we deal with text, we generally perform some basic cleaning like lower-casing all the words (otherwise“hello” and “Hello” will be regarded as two separate words), removing special tokens (like ‘%’, ‘$’, ‘#’, etc.), eliminating words which contain numbers (like ‘hey199’, etc.).In sentiment analysis we also remove the stopwords but we will not remove stopwords here because it will completely change the sense of sentence.For eg if dog is running it will not able to predict is which will be a senseless sentence.we will also avoid stemming here because running and run will also change the meaning of predicted sentence so we have to be precise.This all will reduce my vocab size after removing numbers and punctuations marks using regular expression.Reducing vocab size have great advantage because reducing the vocab size will reduce dense layer at end which will ultimately lead to less number of parameters .I Created a vocabulary of all the unique words present across all the 8000\*5 (i.e. 40000) image captions in the data set from my cleaned descriptions captions. We have now 8424 unique words across all the 40000 image captions.I then rejected all those words which have frequency less than 10. 10 is my threshold frequency for this project.

**3.4 Loading the training set :**

The text file “Flickr\_8k.trainImages.txt” contains the names of the images that belong to the training set.So I load these names.I load the descriptions of these images from “descriptions.txt”.However, when I load them, I added two tokens in every caption ‘startseq’ , This is a start sequence token which will be added at the start of every caption and ‘endseq’ This is an end sequence token which will be added at the end of every caption. startseq and endseq is added so that while training it from RNN the model can easily predict the starting and ending of the sequence.

**3.5 Data Preprocessing :**

Since there are two types of data which we have to feed in the final mlp layer so preprocessing for both the data will be different :

**3.5.1 Data Preprocessing for images(Image feature extraction):**

We are using a pre-trained model to extract the image features.Thai is an imagenet model named ad ResNet-50. ResNet-50 is a convolutional neural network that is 50 layers deep. We can load a pre trained version of the network trained on more than a million images from the ImageNet database . The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

We need to convert every image into a fixed sized vector which can then be fed as input to the final Image caption generator model. For this purpose, we opt for **transfer learning** by using the ResNet-50 model.Our purpose here is not to classify the image but just get fixed-length informative vector for each image.Hence, we just remove the last softmax layer from the model and extract a 2048 length vector (**bottleneck features**) for every image.we are not going to use this resnet model but we are using this only for the feature extraction.After this we will use functional API to make a new model in which we will use input as same given to resent and output will be the second last layer of resnet.This is done because we does not require the dense layer we only require the feature vector of the images.After this we encode our data in which we first preprocess the image and than pass it from the model as input and than store it as a feature vector.Finally we have to reshape our feature vector into 4d tensor because the input given to model is in the form of batches like this (1,224,224,3) and currently it is 3d.while preprocessing we also have to take care of Normalisation in which we use preprocessinput as function of keras. This preprocessinput is done by resnet because resnet is trained on this kind of input,Pixels are clipped in Range 0 to 255 basically it has subtracted the channel mean from all of its pixels.

Image

Image PreProcess

Resnet

Model

Encoded

Image

Fig 2:Flow chart mechanism for Image processing

After encoding for all the images of the training and testing set I stored it into disk using pickle for the future purpose.

**3.5.2 Data Preprocessing — Captions:**

Stating simply, we will represent every unique word in the vocabulary by an integer (index). we have 1845 unique words in the corpus and thus each word will be represented by an integer index between 1 to 1845.I am constructing two dictionaries that is word\_to\_idx storing word to index and idx\_to\_word storing word corresponding to every index.Since we would be doing batch processing we need to make sure that each sequence is of equal length. Hence we need to append 0’s (zero padding) at the end of each sequence. So we will append those many numbers of zeros which will lead to every sequence having a max length in all the captions which is 35.

Since we are using Transfer learning instead of training directly from the scratch from embedding layer.so we will use Glove vector which will convert each vector into 50 dimensional vector. Every word (or index) will be mapped (embedded) to higher dimensional space through one of the word embedding techniques.we will map the every word (index) to a 50-long vector and for this purpose, we will use a pre-trained GLOVE Model.GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Now, for all the 1845 unique words in our vocabulary, we create an embedding matrix which will be loaded into the model before training.so we will make a matrix which we have to input in the final model whose shape will be 1845\*50 this will get directly feeded into the model for training.The same is shown in the form of flowchart below.Here Numeri value is for all the words in captions whose frequency is more than 10.which is already stored in word to index.

Vector(Vocabsize\*50)

Model feed

Embedding Layer

Numeric value

Pretrained Gloved Vector

Fig 3 Flow chart mechanism for captions processing

**3.6 Data Preparation using Generator Function:**

This is one of the most important steps in this project. Machine can not output a complete sentence in once.The problem is dealt like a supervised machine learning problem where we will be using image name as input and it will produce a word corresponding to the image and also corresponding to the words already predicted in the sentence and will predict till endseq is not approached and all combined will lead to form caption starting from startseq and ending at endseq. We will prepare the data in a manner which will be convenient to be given as input to the deep learning model.

The way in which i predicted the caption is as follows:

For the first time, we provide the image vector and the first word as input and try to predict the second word, i.e.:

**Input = Image\_1 + ‘startseq’; Output = ‘the’**

Then we provide image vector and the first two words as input and try to predict the third word, i.e.:

**Input = Image\_1 + ‘startseq the’; Output = ‘cat’**

And so on…

Thus, we can summarize the data matrix for one image and its corresponding caption as follows:

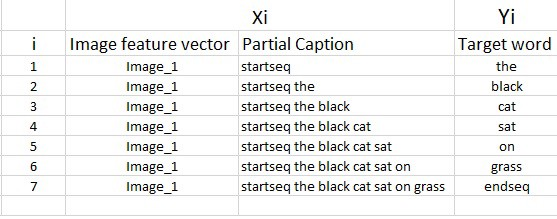


Fig 4:visualisation of supervised learning training

(source:[https://miro.medium.com/max/696/1\*0ixJXDqGxFYR6XZQc5Bh5w.jpeg](https://miro.medium.com/max/696/1*0ixJXDqGxFYR6XZQc5Bh5w.jpeg))

one image+caption is **not a single data point** but multiple data points depending on the length of the caption.Similarly if we consider both the images and their captions, our data matrix.

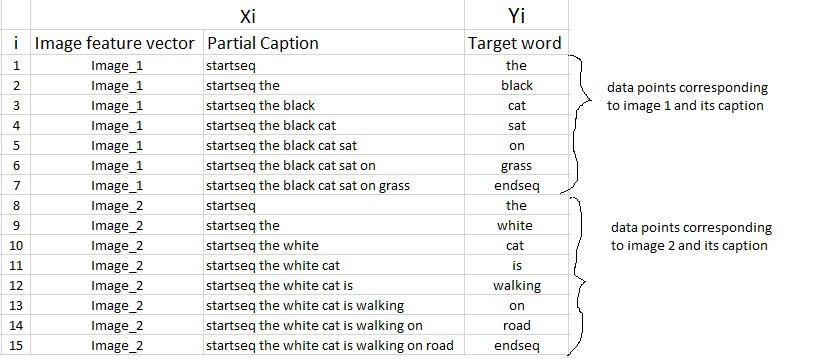


Fig 5:visualisation of supervised learning training for two images

(source:[https://miro.medium.com/max/875/1\*ME49hZnlJDtkA4cWtZjKNg.jpeg](https://miro.medium.com/max/875/1*ME49hZnlJDtkA4cWtZjKNg.jpeg))

Here both image id and partial caption goes as input and help in prediction.Since we are processing sequences, we will employ a Recurrent Neural Network to read these partial captions. we are going to pass the sequence of indices where each index represents a unique word rather than feeding direct english word caption. We have already created an index for each word in caption preprocessing.It will actually look like this.

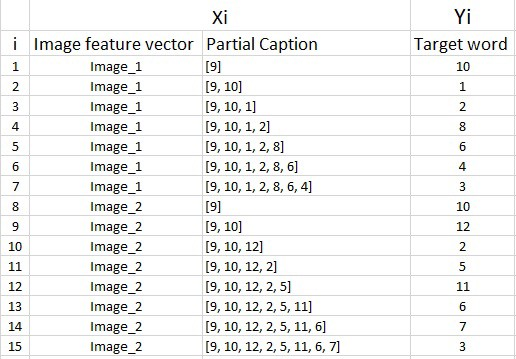


Fig 6:Visualisation after processing of captions

(source:[https://miro.medium.com/max/645/1\*6G1eDpwq11eRY4rhD0yXPg.jpeg](https://miro.medium.com/max/645/1*6G1eDpwq11eRY4rhD0yXPg.jpeg))

I have done batch processing so I have made all the sequences of equal length and I have appended 0's (zero padding) at the end of each sequence. I have already calculated the maximum length of caption in all the captions present around(40,000).which came out to 35.

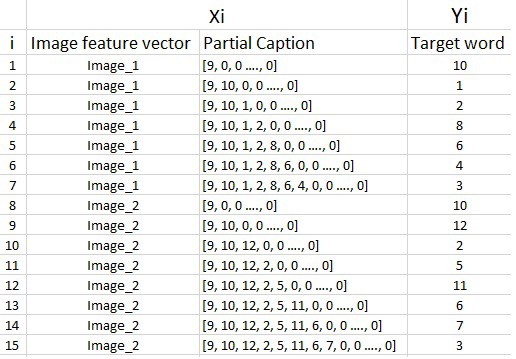
****

Fig7:Visualisation after padding

(source:[https://miro.medium.com/max/645/1\*gefPePe1I2-9pryw3axP1A.jpeg](https://miro.medium.com/max/645/1*gefPePe1I2-9pryw3axP1A.jpeg))

ince the data is very large it can not be directly feeded for training so we will use a datagenerator function.in our actual training dataset we have 6000 images, each having 5 captions. This makes a total of 30000 images and captions.

Even if we assume that each caption on an average is just 7 words long, it will lead to a total of 30000\*7 i.e. 210000 data points.

size of the data matrix:

Size of the data matrix = n\*m

n-> number of data points (assumed as 210000)

m-> length of each data point

Clearly m= Length of image vector(2048) + Length of partial caption(x)

m = 2048 + x

Since we know Every word (or index) is mapped (embedded) to higher dimensional space through one of the word embedding techniques each sequence contains 35 indices, where each index is a vector of length 50. Therefore x = 35\*50 = 1750

m = 2048+1750 = 3798

Finally, size of data matrix= 210000 \* 1750= 36,75,00,000 blocks.

This is a pretty huge requirement and even if we are able to manage to load this much data into the RAM, it will make the system very slow.For this reason we use data generators a lot in Deep Learning.

**3.7 Model Architecture :**

Since the input consists of two parts, an image vector and a partial caption, we cannot use the Sequential API provided by the Keras library. For this reason, we use the Functional API which allows us to create Merge Models.

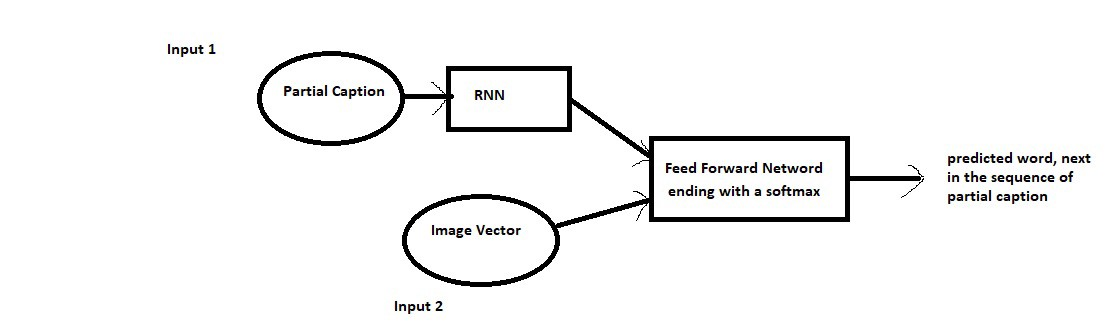


Fig8:Model High level Architecture

(source:[https://miro.medium.com/max/1250/1\*rfYN2EELhLvp2Van3Jo-Yw.jpeg](https://miro.medium.com/max/1250/1*rfYN2EELhLvp2Van3Jo-Yw.jpeg))

The partial caption and image vector will go in the main model and then output will go back again to the model to predict the next line.Partial caption goes into the embedding layer and then the number goes into RNN model.This is complete description how input and output is feeded into model.

Images

Input-(shape 2048)

Dropout(0.3)(Input)

Denslayer(256,relu)

(**Image is converted in 256 Dimesnioanl vector and it will be input to the final MLP).**

Text

Input-(shape(maxlen=35))

inp1=Embedding Layer(Input)

Dropout(0.3)(inp1)

Denselayer(256)**LSTNLayers**

(Text is also converted in 256 Dimensinal vector and will get feeded into MLP)

Input to MLP(Which is decoder here)

Concatenation = Image(256)+Text(256)

Out1 = Denselayer(256,Relu)(concatenation)

Now both images and partial caption is compressed in 256 neurons

Outputs = (denselayer = vocabsize , activation = softmax)(out1)

(Now it will give probablity distribution for the predicted next word from all the words of vocab size)

Feeding input and output to Model for final training

Fig 9:Complete flow chart of model

This was the complete summary for the model obtained by me:

First is an input layer consisting of two layers, one from a partial caption as text and one from the image.Then text is passed from the embedding layer and dropout of 0.3 is added in previous input layers of both input text and that of image.Than a text is passed from lstn layers with a dense layer of 256 neurons and relu activation function.similarity for image it is is also passed from denselayer of 256 neurons and relu activation function.Afer concatecating both we comprese it to 256 neuros dense layer and than pass it to model and final layer has dense layer with neuros has vocab size with a softmax function.It will give probablity distribution for all the vocab words.

**Model Summary**

Layer (type) Output Shape Param # Connected to

==================================================================================================

input\_3 (InputLayer) (None, 35) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

input\_2 (InputLayer) (None, 2048) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

embedding\_1 (Embedding) (None, 35, 50) 92400 input\_3[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 2048) 0 input\_2[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 35, 50) 0 embedding\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 256) 524544 dropout\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_1 (LSTM) (None, 256) 314368 dropout\_2[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

add\_17 (Add) (None, 256) 0 dense\_1[0][0]

lstm\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 256) 65792 add\_17[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 1848) 474936 dense\_2[0][0]

==================================================================================================

Total params: 1,472,040

Trainable params: 1,472,040

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Chapter 3**

**Results**

To complete this image caption project,various Deep learning models were used in this paper, Keras was also used this paper, which is an advanced-level artificial neural networks API.The machine was generating the captions which were quite relevant which i have shown in Fig10,Fig11,Fig12,Fig13,Fig14,,Fig15 and Fig16. The Bot is generating very relevant captions for the tested images.Some of the times it is missing some crucial points but that can also be improved if the size of dataset is increased.since i am using flickr 8k dataset.The flickr30k dataset could even predict like normal human being.since Flickr 30k dataset contains thirty thousand images in comparison to Flickr 8k which contains only eight thousand images.

Fig10 Result1 Fig11 Result2

In this photo bot has predicted quite well. This is not that perfect one but a good try

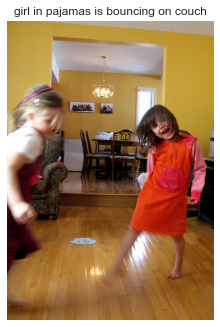


Fig12: Result3 Fig13:Result4

Here the prediction is done for football by This is quite a good prediction.

Machine.

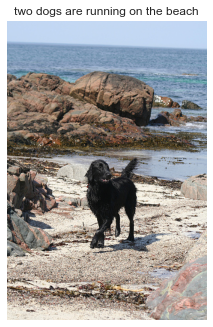


Fig14:Result5 Fig15:Resul6

This is again quite good and This is quite good try of prediction by the model

relevant one . but these men are not standing on a bench.

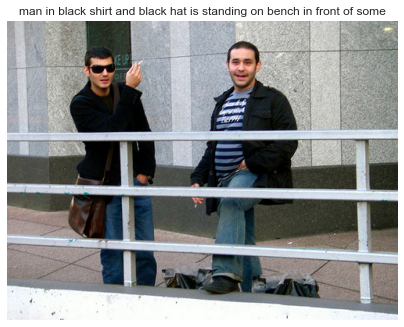


Fig16:Result7 Fig17:Result8

This is again a great picture. This is also a good try for prediction

# 

# 

# 

# 

# 

# 

# 

**Chapter 4**

**Conclusions and Future aspects:**

In my Image caption generator model,I successfully performed both the training and development part of the project in which I successfully cleaned the data and then preprocessed both image data and text data.After the training I saved my model weights and those weights were used to generate the caption for images.Images was able to generate the captions which were highly relevant.

Future aspects:

This project is generating caption we can transform this caption into audio.We can make a hardware device for the blind people in which we can fit this model and a live camera will see the pictures and using this model pictures will gen converted into text and then text can be further converted into audio which will be listened by blind people.

We can also use this project in sef driving cars.In self driving cars a live web camera will take the pictures and those pictures will get converted into text and can guide the care regarding the obstacle in front of car.

**References**

1 Chen, Jianhui, Wenqiang Dong, and Minchen Li. "Image Caption Generator Based On Deep Neural Networks." (2014).

2 Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

3 Wang, Haoran, Yue Zhang, and Xiaosheng Yu. "An Overview of Image Caption Generation Methods." *Computational Intelligence and Neuroscience* 2020 (2020).

4 Zhou, Luowei, et al. "Unified Vision-Language Pre-Training for Image Captioning and VQA." *AAAI*. 2020.

5..<https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8>

**Appendix:**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import keras#Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow.**

**import re#This module provides regular expression**

**import nltk**

**from nltk.corpus import stopwords**

**import string**

**import json**

**from time import time**

**import pickle**

**from keras.applications.vgg16 import VGG16**

**from keras.applications.resnet50 import ResNet50, preprocess\_input, decode\_predictions**

**from keras.preprocessing import image**

**from keras.models import Model, load\_model**

**from keras.preprocessing.sequence import pad\_sequences**

**from keras.utils import to\_categorical**

**from keras.layers import Input, Dense, Dropout, Embedding, LSTM**

**from keras.layers.merge import add**

**# In[3]:**

**# Read Text Captions**

**def readTextFile(path):**

**with open(path) as f:**

**captions = f.read()**

**return captions**

**# In[4]:**

**captions = readTextFile("Flickr\_Data/Flickr\_Data/Flickr\_TextData/Flickr8k.token.txt")**

**captions = captions.split('\n')[:-1]#Last line is basically empty so we can discard that**

**# In[5]:**

**print(len(captions))**

**# In[6]:**

**captions[0]**

**# In[7]:**

**captions[0].split("\t")**

**# In[8]:**

**first,second = captions[0].split('\t')**

**print(first)**

**print(second)**

**# In[9]:**

**first,second = captions[0].split('\t')**

**print(first.split(".")[0])**

**print(second)**

**# In[10]:**

**# Dictionary to Map each Image with the list of captions it has**

**# In[11]:**

**descriptions = {}**

**for x in captions:**

**first,second = x.split('\t')**

**img\_name = first.split(".")[0]**

**#if the image id is already present or not**

**if descriptions.get(img\_name) is None:**

**descriptions[img\_name] = []**

**descriptions[img\_name].append(second)**

**# In[12]:**

**descriptions["1000268201\_693b08cb0e"]**

**# In[13]:**

**get\_ipython().system('pip install opencv-python')**

**# In[14]:**

**##Reading the image**

**# In[15]:**

**import cv2**

**# In[16]:**

**IMG\_PATH = "Flickr\_Data/Flickr\_Data/Images"**

**img = cv2.imread(IMG\_PATH+"/1000268201\_693b08cb0e.jpg")**

**img = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)**

**plt.imshow(img)**

**plt.axis("off")**

**plt.show()**

**# ### Data Cleaning**

**#**

**# In[17]:**

**def clean\_text(sentence):**

**sentence = sentence.lower()**

**sentence = re.sub("[^a-z]+"," ",sentence)#using regular expression to remove the punctuation marks and other numbers**

**sentence = sentence.split()**

**sentence = [s for s in sentence if len(s)>1]**

**sentence = " ".join(sentence)**

**return sentence**

**# In[18]:**

**clean\_text("A cat is sitting over the house # 64")**

**# In[19]:**

**# Clean all Captions**

**for key,caption\_list in descriptions.items():**

**for i in range(len(caption\_list)):**

**caption\_list[i] = clean\_text(caption\_list[i])**

**# In[20]:**

**descriptions["1000268201\_693b08cb0e"]**

**# In[21]:**

**# Write the data to text file to store for the future use**

**with open("descriptions\_1.txt","w") as f:**

**f.write(str(descriptions))**

**# ### Vocabulary**

**# In[22]:**

**descriptions = None**

**with open("descriptions\_1.txt",'r') as f:**

**descriptions= f.read()**

**json\_acceptable\_string = descriptions.replace("'","\"")#making json acceptable string**

**descriptions = json.loads(json\_acceptable\_string)#converting the stirng in dictionary**

**# In[23]:**

**# since there is a very large dataset hence we are storing in a file and it take time in preprocessing so it is better choice**

**# to make a sep file for the descriptions**

**# In[24]:**

**print(type(descriptions))**

**# In[25]:**

**voc = set()**

**voc.update(["apoorv"])**

**voc.update(["excellent","ajay","muna"])**

**voc.update(["apoorv"])**

**print(voc)**

**# In[26]:**

**# Vocab Creation**

**vocab = set() #set basically store all the unique words**

**for key in descriptions.keys():**

**[vocab.update(sentence.split()) for sentence in descriptions[key]]**

**print("Vocab Size : %d"% len(vocab))**

**# In[27]:**

**# Total No of words across all the sentences**

**total\_words = []**

**for key in descriptions.keys():**

**[total\_words.append(i) for des in descriptions[key] for i in des.split()]**

**print("Total Words %d"%len(total\_words))**

**# In[28]:**

**# Filter Words from the Vocab according to certain threshold frequncy**

**# In[29]:**

**import collections#container data types called Collections**

**counter = collections.Counter(total\_words)**

**freq\_cnt = dict(counter)#freq\_cnt give the dictionary of all the words with there occurences**

**print(len(freq\_cnt.keys()))**

**# In[30]:**

**counter**

**# In[32]:**

**type(counter)#we chnange this into type freq\_cnt**

**# In[28]:**

**# Sort this dictionary according to the freq count**

**sorted\_freq\_cnt = sorted(freq\_cnt.items(),reverse=True,key=lambda x:x[1])**

**# Filter**

**threshold = 10**

**sorted\_freq\_cnt = [x for x in sorted\_freq\_cnt if x[1]>threshold]**

**total\_words = [x[0] for x in sorted\_freq\_cnt]**

**# In[33]:**

**print(len(total\_words))#form 3 lakh words finally we have filtered it to 2k words whose frequency is greater than 10**

**# ### Prepare Train/Test Data**

**# In[30]:**

**train\_file\_data = readTextFile("Flickr\_Data/Flickr\_Data/Flickr\_TextData/Flickr\_8k.trainImages.txt")**

**test\_file\_data = readTextFile("Flickr\_Data/Flickr\_Data/Flickr\_TextData/Flickr\_8k.testImages.txt")**

**# In[31]:**

**train = [row.split(".")[0] for row in train\_file\_data.split("\n")[:-1]]**

**test = [row.split(".")[0] for row in test\_file\_data.split("\n")[:-1]]**

**# In[32]:**

**train[:10]**

**# In[33]:**

**# Prepare Description for the Training Data**

**# Tweak - Add <s> and <e> token to our training data**

**train\_descriptions = {}**

**for img\_id in train:**

**train\_descriptions[img\_id] = []**

**for cap in descriptions[img\_id]:**

**cap\_to\_append = "startseq " + cap + " endseq"**

**train\_descriptions[img\_id].append(cap\_to\_append)**

**# In[34]:**

**train\_descriptions["2513260012\_03d33305cf"]**

**# ### Transfer Learning**

**# - Images --> Features**

**# - Text ---> Features**

**# ### Step - 1 Image Feature Extraction**

**# In[35]:**

**model = ResNet50(weights="imagenet",input\_shape=(224,224,3))**

**model.summary()**

**# In[36]:**

**model.layers[-2].output**

**# In[37]:**

**model\_new = Model(model.input,model.layers[-2].output)**

**# In[38]:**

**def preprocess\_img(img):**

**img = image.load\_img(img,target\_size=(224,224))**

**img = image.img\_to\_array(img)**

**img = np.expand\_dims(img,axis=0)**

**# Normalisation**

**img = preprocess\_input(img)**

**return img**

**# In[39]:**

**imgi = preprocess\_img(IMG\_PATH+"/1000268201\_693b08cb0e.jpg")**

**plt.imshow(imgi[0])**

**plt.axis("off")**

**plt.show()**

**##This image is not clearly visble to us but this is how it is fed to resent dont worry for the pixel because while preprocessin**

**#it automatically clips between 0 to 255 here it is visble like this bcoz form each pixel we have subtracted the channel mean**

**##This is how the image look after preprocessing**

**# In[41]:**

**print(imgi)**

**# In[34]:**

**def encode\_image(img):**

**img = preprocess\_img(img)**

**feature\_vector = model\_new.predict(img)**

**feature\_vector = feature\_vector.reshape((-1,))#THis reshape is done because image is feed to the final network in the form**

**#of 4d tensor or batches is also pressent so here bydefalut it will become 1,**

**#print(feature\_vector.shape)**

**return feature\_vector**

**# In[43]:**

**encode\_image(IMG\_PATH+"/1000268201\_693b08cb0e.jpg")**

**# In[44]:**

**len(train)**

**# In[45]:**

**start = time()**

**encoding\_train = {}**

**#image\_id -->feature\_vector extracted from Resnet Image**

**for ix,img\_id in enumerate(train):**

**img\_path = IMG\_PATH+"/"+img\_id+".jpg"**

**encoding\_train[img\_id] = encode\_image(img\_path)**

**if ix%100==0:**

**print("Encoding in Progress Time step %d "%ix)**

**end\_t = time()**

**print("Total Time Taken :",end\_t-start)**

**# In[46]:**

**get\_ipython().system('mkdir saved')**

**# In[47]:**

**# Store everything to the disk**

**with open("saved/encoded\_train\_features.pkl","wb") as f:**

**pickle.dump(encoding\_train,f)**

**# In[48]:**

**start = time()**

**encoding\_test = {}**

**#image\_id -->feature\_vector extracted from Resnet Image**

**for ix,img\_id in enumerate(test):**

**img\_path = IMG\_PATH+"/"+img\_id+".jpg"**

**encoding\_test[img\_id] = encode\_image(img\_path)**

**if ix%100==0:**

**print("Test Encoding in Progress Time step %d "%ix)**

**end\_t = time()**

**print("Total Time Taken(test) :",end\_t-start)**

**# In[49]:**

**with open("saved/encoded\_test\_features.pkl","wb") as f:**

**pickle.dump(encoding\_test,f)**

**# ### Data pre-processing for Captions**

**# In[50]:**

**# Vocab**

**len(total\_words)**

**# In[51]:**

**word\_to\_idx = {}**

**idx\_to\_word = {}**

**for i,word in enumerate(total\_words):**

**word\_to\_idx[word] = i+1#we dont want to map with 0 index**

**idx\_to\_word[i+1] = word**

**# In[52]:**

**#word\_to\_idx["dog"]**

**#idx\_to\_word[1]**

**print(len(idx\_to\_word))**

**# In[53]:**

**# Two special words**

**idx\_to\_word[1846] = 'startseq'**

**word\_to\_idx['startseq'] = 1846**

**idx\_to\_word[1847] = 'endseq'**

**word\_to\_idx['endseq'] = 1847**

**vocab\_size = len(word\_to\_idx) + 1 #0 index is also included here**

**print("Vocab Size",vocab\_size)**

**# In[57]:**

**max\_len = 0**

**for key in train\_descriptions.keys():**

**for cap in train\_descriptions[key]:**

**max\_len = max(max\_len,len(cap.split()))**

**print(max\_len)**

**# ## https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8**

**# ### Data Loader (Generator)**

**# In[58]:**

**def data\_generator(train\_descriptions,encoding\_train,word\_to\_idx,max\_len,batch\_size):**

**X1,X2, y = [],[],[]**

**n =0**

**while True:**

**for key,desc\_list in train\_descriptions.items():**

**n += 1**

**photo = encoding\_train[key+".jpg"]**

**for desc in desc\_list:**

**seq = [word\_to\_idx[word] for word in desc.split() if word in word\_to\_idx]**

**for i in range(1,len(seq)):**

**xi = seq[0:i]**

**yi = seq[i]**

**#0 denote padding word**

**xi = pad\_sequences([xi],maxlen=max\_len,value=0,padding='post')[0]**

**yi = to\_categorcial([yi],num\_classes=vocab\_size)[0]**

**X1.append(photo)**

**X2.append(xi)**

**y.append(yi)**

**if n==batch\_size:**

**yield [[np.array(X1),np.array(X2)],np.array(y)]**

**X1,X2,y = [],[],[]**

**n = 0**

**# ## Word Embeddings**

**# In[59]:**

**f = open("./saved/glove.6B.50d.txt",encoding='utf8')**

**# In[60]:**

**embedding\_index = {}**

**for line in f:**

**values = line.split()**

**word = values[0]**

**word\_embedding = np.array(values[1:],dtype='float')**

**embedding\_index[word] = word\_embedding**

**# In[61]:**

**f.close()**

**# In[62]:**

**embedding\_index['apple']**

**# In[63]:**

**def get\_embedding\_matrix():**

**emb\_dim = 50**

**matrix = np.zeros((vocab\_size,emb\_dim))**

**for word,idx in word\_to\_idx.items():**

**embedding\_vector = embedding\_index.get(word)**

**if embedding\_vector is not None:**

**matrix[idx] = embedding\_vector**

**return matrix**

**# In[64]:**

**embedding\_matrix = get\_embedding\_matrix()**

**embedding\_matrix.shape**

**# In[65]:**

**#embedding\_matrix[1847]**

**# #### Model Architecture**

**# In[66]:**

**input\_img\_features = Input(shape=(2048,))**

**inp\_img1 = Dropout(0.3)(input\_img\_features)**

**inp\_img2 = Dense(256,activation='relu')(inp\_img1)**

**# In[67]:**

**# Captions as Input**

**input\_captions = Input(shape=(max\_len,))**

**inp\_cap1 = Embedding(input\_dim=vocab\_size,output\_dim=50,mask\_zero=True)(input\_captions)**

**inp\_cap2 = Dropout(0.3)(inp\_cap1)**

**inp\_cap3 = LSTM(256)(inp\_cap2)**

**# In[68]:**

**decoder1 = add([inp\_img2,inp\_cap3])**

**decoder2 = Dense(256,activation='relu')(decoder1)**

**outputs = Dense(vocab\_size,activation='softmax')(decoder2)**

**# Combined Model**

**model = Model(inputs=[input\_img\_features,input\_captions],outputs=outputs)**

**# In[69]:**

**model.summary()**

**# In[70]:**

**# Important Thing - Embedding Layer**

**model.layers[2].set\_weights([embedding\_matrix])**

**model.layers[2].trainable = False**

**# In[71]:**

**model.compile(loss='categorical\_crossentropy',optimizer="adam")**

**# ### Training of Model**

**# In[72]:**

**#epochs = 20**

**# batch\_size = 3**

**# steps = len(train\_descriptions)//10**

**epochs = 10**

**number\_pics\_per\_bath = 3**

**steps = len(train\_descriptions)//number\_pics\_per\_bath**

**# # Trained this model on google collab and saved the wights to work in my local pc**

**# In[76]:**

**def train():**

**for i in range(epochs):**

**generator = data\_generator(train\_descriptions,encoding\_train,word\_to\_idx,max\_len,batch\_size)**

**model.fit\_generator(generator,epochs=1,steps\_per\_epoch=steps,verbose=1)**

**model.save('./model\_weights/model\_'+str(i)+'.h5')**

**# In[77]:**

**model = load\_model('./model\_weights/model\_9.h5')**

**# ## Predictions**

**# In[78]:**

**def predict\_caption(photo):**

**in\_text = "startseq"**

**for i in range(max\_len):**

**sequence = [word\_to\_idx[w] for w in in\_text.split() if w in word\_to\_idx]**

**sequence = pad\_sequences([sequence],maxlen=max\_len,padding='post')**

**ypred = model.predict([photo,sequence])**

**ypred = ypred.argmax() #WOrd with max prob always - Greedy Sampling**

**word = idx\_to\_word[ypred]**

**in\_text += (' ' + word)**

**if word == "endseq":**

**break**

**final\_caption = in\_text.split()[1:-1]**

**final\_caption = ' '.join(final\_caption)**

**return final\_caption**

**# In[79]:**

**# Pick Some Random Images and See Results**

**plt.style.use("seaborn")**

**for i in range(15):**

**idx = np.random.randint(0,1000)**

**all\_img\_names = list(encoding\_test.keys())**

**img\_name = all\_img\_names[idx]**

**photo\_2048 = encoding\_test[img\_name].reshape((1,2048))**

**i = plt.imread("Flickr\_Data/Flickr\_Data/Images/"+img\_name+".jpg")**

**caption = predict\_caption(photo\_2048)**

**#print(caption)**

**plt.title(caption)**

**plt.imshow(i)**

**plt.axis("off")**

**plt.show()**