

BUILDING A VOICE BASED IMAGE CAPTION GENERATOR WITH DEEP LEARNING

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in Information Technology

by

MOHANA PRIYA R (Reg No : 37120048)
DIVYA S(Reg No : 37120025)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of MOHANA PRIYA R (37120048) DIVYA S (37120025) who have done the Project work as a team who carried out the project entitled **"BUILDING A VOICE BASED IMAGE CAPTION GENERATOR WITH DEEP LEARNING"** under my supervision from November 2020 to April 2021.

Internal Guide
Dr .V. MARIA ANU M.E., Ph.D.,

Head of the Department
DR.R.SUBHASHINI M.E., PH.D.,

Submitted for Viva Voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, MOHANA PRIYA R (REG NO: 37120048) and DIVYA S (REG NO: 37120025) hereby declare that the Project Report entitled “**BUILDING AN VOICE BASED IMAGE CAPTION GENERATOR WITH DEEP LEARNING**” done by us under the guidance of **Dr.V.MARIA ANU M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr.R.Subhashini M.E., Ph.D. , Head** of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.V.Maria Anu M.E., Ph.D.**, his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

The Image caption generator is a task that involves computer vision and NLP(natural language processing) Concepts to recognize the context of an image and describe them in a natural language like English. And the Convolutional Neural Network(CNN) and Long Short Term Memory(LSTM) can be combined to create an image caption generator and generate captions for your own images. In the field of artificial intelligence, image caption has gradually attracted the attention of many Programmers and developers, And the application of image caption is extensive and significant , It will detect the relationship between people, animals. This project will help us to understand the related methods and attention mechanism, which plays an important role in computer vision and is recently widely used in image caption generation tasks. And the advantages and the shortcomings of these methods are implemented, providing the commonly used datasets and evaluation criteria in this field. and it explains how the present scenarios happening in front of visually impaired people. The advent of machine learning solutions like image captioning is a boon for visually impaired people who are unable to comprehend visuals. With the use of an image caption generator, the image descriptions can be read out to visually impaired people, and our technology will be act as their eyes

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	1
	1.1 General Introduction	1
	1.2 Objective	1
	1.3 Existing system	1
	1.4 Proposed system	2
2	LITERATURE SURVEY	3
3	AIM AND SCOPE OF THE MODEL	7
	3.1 Aim	7
	3.2 Scope of the Project	7
4	METHODOLOGY AND IMPLEMENTATION	8
	4.1 System Requirements	8
	4.2 Packages	9
	4.3 System Architecture	10
	4.4 Implementation	14
5	RESULT AND FUTURE WORKS	22
	5.1 Result Analysis	22
	5.2 Future Work	22
6	CONCLUSION	23
	6.1 Summary	23
	6.2 Appendix	23

A. Sample code	23
B. Output Screenshots	33
C.Publication Report	37
References	38

LIST OF FIGURES

S.NO	NAME	PAGE NO
4.1	System Architecture	10
3.2	CNN Architecture	12
3.3	LSTM Architecture	13
3.4	Dataset Collection	15
3.5	Image with Description	15
3.6	Extraction of feature	16
3.7	Epoch with loss value	20

LIST OF ABBREVIATION

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
API	Application Programming Language

CHAPTER 1

INTRODUCTION

1.1 GENERAL INTRODUCTION:

Nowadays technology will help the human to crack and reach the extreme level of peak to reach particular goals. So researchers where focuses on how to stimulate the system to work like a human brain with the help of concepts like machine learning, deep learning and artificial intelligence. While we deep dive into the concept of caption generator so much of steps involved .when we thought about the humans its very simple and easy method to generate a caption and same for the system is an impossible step. By using the natural language processing toolkit and certain algorithm to train the model to create description for the image with the help of system it will train the system to generate a description for an image. We have to train our model to visualize the image with the help of visualization of objects, actions and relationship between each and everything present in the image, and its a tedious process. To overcome that we are using CNN to extract the features of an image.

1.2 OBJECTIVE:

To learn the concept of CNN and LSTM model and build a working model of image caption generator. Features of the images where extracted by using CNN model trained on large image net dataset. And we are giving the features as an input to the LSTM model and it will generate caption for an image.It requires both computer vision system to localize and describe the features of the image in a single word using natural language.

1.3 EXISTING SYSTEM:

The image caption generator are describing the scenes because images without the captions is meaningless .The process relies on Convolution Neural Network(CNN) that encodes an image into a compact representation by object detection that generates a corresponding sentence. This model is trained to maximize the accuracy and precision

of the caption for the given image. Experiments on several datasets show the robustness of ICG in terms of qualitative results and quantitative evaluations. By imparting various hyper parameter tuning, the model can be made more efficient and productive.

1.4 PROPOSED SYSTEM:

The Proposed methodology for generating captions with the detection and recognition of objects using deep learning It consists of object detection, feature extraction, Convolution Neural Network (CNN) for feature extraction and for scene classification So, to make our image caption generator model, we will be merging these architectures. It is also called a CNN-RNN model. We will use the pre-trained model Exception. And LSTM will use the information from CNN to help generate a description of the image. Once the caption is generated it plays it in a voice

CHAPTER 2

LITERATURE SURVEY

Automatic image annotation is the process by which a computer automatically assign metadata in the form of captioning or keywords to a digital image. This application of computer vision techniques is used in image retrieval system to organize and locate image of interest from a database

TITLE : Combined Hierarchical Model for Automatic Image Annotation and Retrieval.

AUTHOR : T. Sumathi , Dr. M. Hemalatha

YEAR : 2011

DESCRIPTION:

Annotation of images in this work undergoes several stages: first, they extract information from the images by using JEC feature extraction method and form a feature vector, next train several SVM's to create a model for annotation accordingly. Approaches like flat wise, axis-wise and position wise approaches to obtain the strings of annotation. Finally used pair wise fusion between all three methods to sum up strings obtained from all three approaches. Thus the final annotation of an image is obtained at the end of fusion method. Here we treat image annotations a process of transferring keywords from nearest neighbors. The neighborhood structure is constructed using simple low-level image features resulting in a rudimentary model that makes use of JEC distance method to obtain the feature vectors

PROS:

Slick user interface, many difficult functionalities use to encourage and help them to interact with a text, it promotes deeper understanding of passage with a purpose.

CONS:

Complicated installation and only done it with the help of image and no options for drawing any shapes rather than rectangles.

TITLE : Sentence Learning on Deep Convolutional Networks for Image Caption Generation

AUTHOR : Dong-Jin Kim, Donggeun Yoo, Bonggeun Sim, In So Kweon

YEAR : 2016

DESCRIPTION:

Using pre trained CNN has two problems. First, it recognizing unusual objects. Second is the counting problem. Therefore, they proposed a method to fine-tune CNN on sentences Dataset, and employed multi-scale pyramid pooling for aggregating the activation from CNN into Fisher vector. In order to train CNN on sentence data set, a change is needed from sentence to labels. They collected all the words in the data set, and then lemmatize and tagged the words to select only nouns and verbs. The most frequent 100 nouns and verbs are selected to generate classifying label. The network is trained with this annotation. Multi-scale images are fed to this CNN, and dense activation of an image are extracted. In Fisher kernel, all the extracted activation are aggregated to Fisher vectors. Finally, the vectors are pooled to a finale vector with MPP This resulting vector is fed to the sentence decoder. They tested the effect of the input representation that is fed to gLSTM. The results from the proposed method and then original method couldn't predict unusual word but our method correctly caught it. the original method showed correct prediction effectively.

Image captioning has various applications such as recommendations in editing application,usage in virtual assistants for image indexing for visually impaired persons,for social media and several other natural language process we can discuss about it below

TITLE : Deep Learning based Automatic Image Caption Generation

AUTHOR : Varsha Kesavan, Vaidehi Muley, Megha Kolhekar

YEAR : 2019

DESCRIPTION:

In this paper we propose a transfer learning approach to generate automated captions for any given image. In this model the encoder used is pre-trained VGG16 model. This model makes use of a recurrent neural network which encodes the variable length input into a fixed dimensional vector and uses this representation to “decode” it to the desired output sentence. The vector containing the output of the fully connected layer in VGG16 is connected to GRU units is called “thought” vector. In the encoder of the system, pre-trained models VGG 16, RESNET and Inception are used to compare between the results obtained from each of them. The encoder is used to extract thought vector of the image which describes the contents of the image. The size of the output of last fully connected layer is (none,4096). The decoder consists of Tokenizer, embedding layer,GRU layers and dense layer. Each of the captions are first prepended and appended by start and end marker respectively. Tokenizer layer is used to convert the first predefined number of unique words into integer tokens.

PROS:

By using VGG16 neural network we can extract the features of an image it will help them to better understanding of an image.

CONS:

By using this model we cant extract the exact features that are present in the image and it will result in wrong description generation.

Automatic image caption generation using the long short term memory for sequence prediction of caption for an image.so that by using those methods they can implement this caption generation method ,deeply mentioned below.

TITLE : Long Short-Term Memory with Read-only Unit in Neural Image Caption Generator

AUTHOR : Aghasi Poghosyan, Hakob Sarukhanyan.

YEAR : 2017

DESCRIPTION:

Automated caption generation for digital images is one of the fundamental problems in artificial intelligence. Most of the existing works use Long Short-Term Memory as a recurrent neural network cell to solve this task. After training, their deep neural models can generate an image caption. But there is an issue, the next predicted word of the caption depends mainly on the last predicted word, rather than the image content. In this paper we present model that can automatically generate an image description and is based on a recurrent neural network with modified LSTM cell with an additional gate responsible for image features. This modification results in generation of more accurate captions. We have trained and tested our model on MSCOCO image dataset by using only images and their captions.

PROS:

It is a good process for recognition related problems and prediction related problems.

CONS:

They requires a lot of resources and time to get trained and become ready for real world applications.

CHAPTER 3

AIM AND SCOPE OF THE MODEL

3.1 AIM :

Generating a caption for a given image is a challenging problem in the deep learning domain. we will use different techniques of computer vision and NLP to recognize the context of an image and describe them in a natural language like English. we will build a working model of the image caption generator by using CNN (Convolutional Neural Networks) and LSTM (Long short term memory) units. And to judge the quality of the description created during the process.

3.2 SCOPE OF THE PROJECT:

Accurate image captioning with the use of multi modal neural networks has been a interesting topic in the field of deep learning. Its been working with several of these approaches and the algorithms seems to give very promising results. when it comes to using image captioning in real world application most of the time only a few are mentioned such as hearing aid for the visually impaired people and content generation. Main scope is we are adding the additional functionality like audio. once the description is generated it will readout the description of an image.

CHAPTER 4

METHODOLOGY AND IMPLEMENTATION

4.1 SYSTEM REQUIREMENTS:

4.1.1 Software Requirements:

Software requirements deals with defining resources requirements and prerequisites that needs to be installed on a computer to provide functioning of an application the minimal software requirements are as follows,

Front end : Python

Dataset : CSV

IDE : Anaconda Jupyter

OS : Windows 10,7.

4.1.2 Hardware Requirements:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. The minimal hardware requirements are as follows.

Processor : Pentium IV

RAM : 8GB

Processor : 2.4GHz

Main Memory : 8GB RAM

Hard Disk : 1tb

4.2 PACKAGES :

4.2.1 TENSORFLOW:

Tensor flow is an open source library for numerical computation and large-scale machine learning. It bundles together a slew of machine learning and deep learning models and algorithms makes them useful by way of a common metaphor. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensor flow is a symbolic math library based on data flow and differentiable programming.

4.2.2 KERAS:

Keras is a powerful and easy to use free open source python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and Tensor flow and allows you to define and train neural network models in just a few lines of code.

4.2.3 MATPLOT LIB:

Matplotlib is plotting library for the python programming language and its numerical mathematics extension numpy it provides an object oriented API for embedding plots into applications using general purpose GUI toolkit like Tkinter,wxpython,qtetc.pyplot function makes some changes for a created figures, creates a plotting area in figure, decorate the plots with labels. And its multi platform data visualization library built on numpy arrays.

4.2.4 XCEPTION :

It present an interpretation of inception modules in convolutional neural network as being an intermediate step in between regular convolution and the depth wise separable convolution operation. In this light a depth wise separable convolution can

be understood as an inception module with a maximally large number of towers. This architecture dubbed xception slightly outperforms inception v3 on the imagenet dataset and significantly outperforms inception v3 on a large image classification

4.3 SYSTEM ARCHITECTURE:

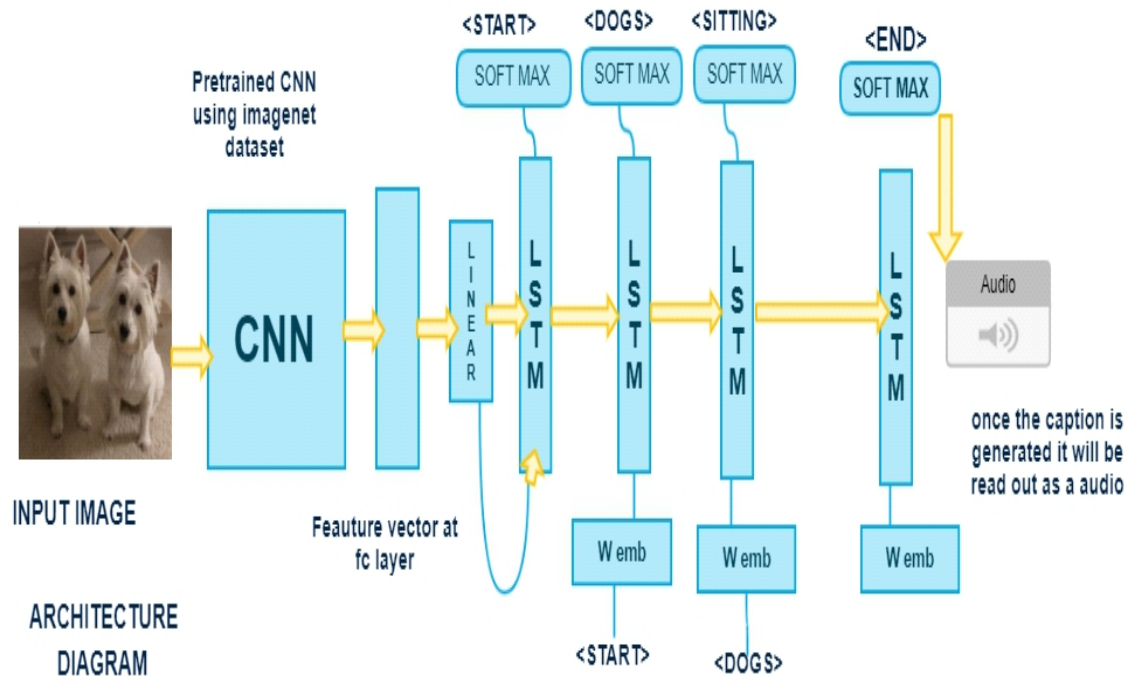


Fig 4.1 system architecture

Input image taken from the user side convolutional neural network identify the objects that present in the image it extract the important features of an image and store those feature vector values ,using pooling functions it will predict the features. Once the process completed it will move on to the long short term memory layer for the sequence sentence prediction based on the previous one, here soft-max function is used to predict the output accurately and for overcoming the over fitting problem ,when we are working with the neural network most of the nodes having the

output that are related to the previous one its results in over fitting,to avoid those problem soft-max layer is used .If the output of this layer is between the range of zero to one ,if the range is greater are lesser it results in the error .and system will not predict the correct description for an image

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

4.3.1 CNN ARCHITECTURE:

CNN can learns the filter automatically without mentioning it explicitly. These filter help in extracting the right and relevant features from the input data.CNN capture the spatial features from an image. Spatial features refers to the arrangement of pixels and the relationship between them in an image. They help us in identifying the object accurately, location of an object as well as its relation with the other objects in an image. If any image is given we can easily identify that it's a human face or animals looking at specific features like eyes, nose ,mouth and so on. We can also see how these specific features are arranged in image. That's what exactly CNN are capable of capturing.

Fully Connected Layer

Lets see the entire process how CNN recognizes a bird

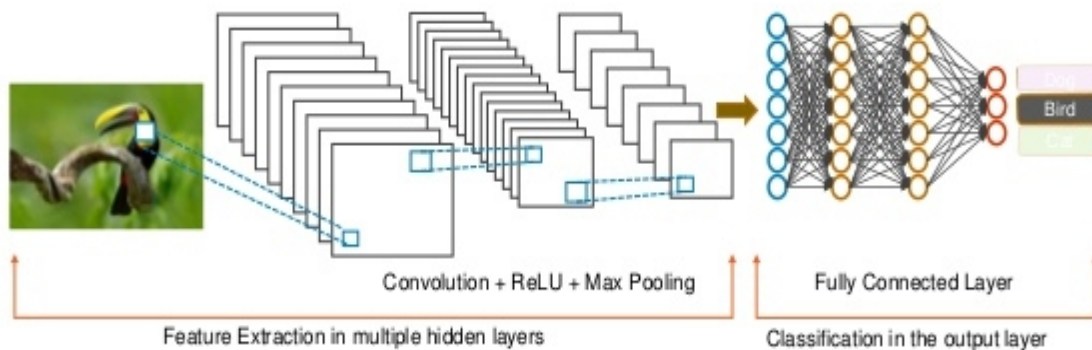


Fig 4.2 CNN Architecture

Convolutional neural networks are specialized deep neural networks which can process the data that has input shape like a two dimensional matrix. Images are easily represented as a two dimensional matrix and CNN is very useful in working with images. It's basically used for image classification and identifying if an image is a bird, plane or superman etc. It scans images from the left to right and top to bottom to pull out important features to classify images. It can handle the images that have been translated, rotated, scaled and changes in perspective. When talking about image processing, extracting the feature of an image manually from an image needs a strong knowledge of the domain. It is an extremely time-consuming process, with the help of deep learning, we can automate the process of feature extraction.

4.3.2 LSTM STRUCTURE:

Long Short Term Memory, they are a type of Recurrent Neural Network which is well suited for sequence prediction problems. Based on the previous text, it can predict what the next word will be; it has proven itself effective from the traditional RNN by overcoming the limitations of RNN which had short term memory. Long short term memory can carry out relevant information throughout the processing of inputs and with a forget

gate, it discards the non-relevant information about the images. we can say that when we move from RNN to LSTM, it has more and more controlling knobs, which control the flow and mixing of inputs as per trained weights. so that long short term memory gives us the most control ability and thus better results. But also comes with the more complexity and operating cost.

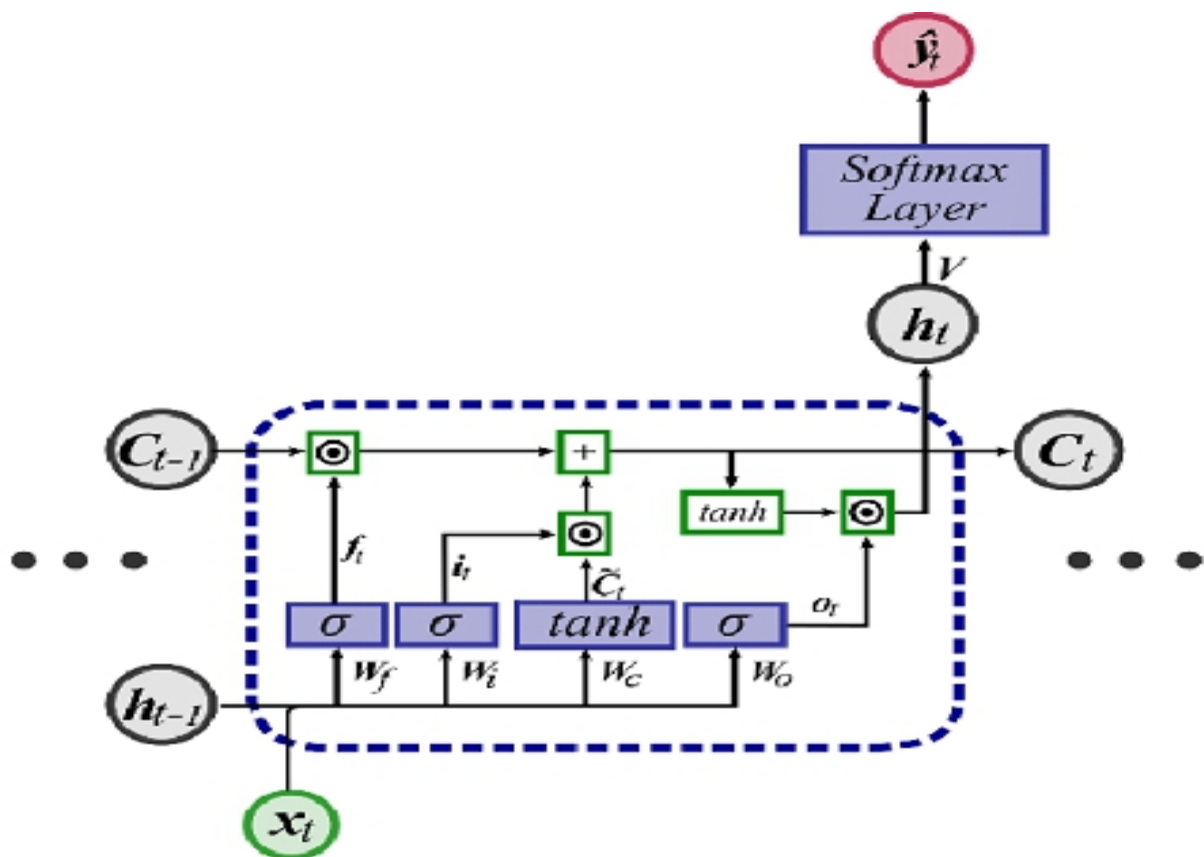


Fig 4.3 Lstm structure

Information is retained by the cells and the memory manipulation are done by the three gates, first is the forget gate in that the information that no longer useful in the cell state is removed with the forget gate, it gets two inputs at a same time and provides one output. if it get the output as zero it forget the information and if the output is one the information is retained for the future use. addition of useful information to the cell state is done by input gate. first the information is regulated using the sigmoid function and filter values to be remembered similar to the forget gate using input. the task of extracting useful information from the current cell state to be presented as an output is done by

output gate.first a vector is generated by applying tanh function on the cell.the information is regulated using the sigmoid function and filter the values to be remembered using the input values. At last the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell.

4.4 IMPLEMENTATION:

1.Data Collection and Data Cleaning.

2.Extracting Feature Vector .

3.Loading Dataset for Training

4.Tokenizing Vocabulary.

5.Creation of Data Generator.

6.Defining Model.

7.Training The Model.

8.Testing

1.DATA COLLECTION AND DATA CLEANING:

For an voice based caption generator, we are using the flicker data set and it contains eight thousand images.for using an large data set is that we can build better models.for this model implementation we require some prerequisite like packages or libraries , tensor flow,keras,pillow,numpy,tqdm.once the data set collection is over we have to perform some cleaning operations.

Data cleaning takes all the description and performs data cleaning .Its an important process when we deal with the textual data,so that we are performing some cleaning operations like removing the vowels ,converting the words to lower case letters,removing digits ,hanging s,and special characters.so a caption like “a dog is running through the grass” will be transformed into “ dog running through the grass”.text_vocabulary function will separate all the unique words and create the

vocabulary from all the descriptions.list of all descriptions have been processed and stored into a file.And it contains the image id with their relevant captions.

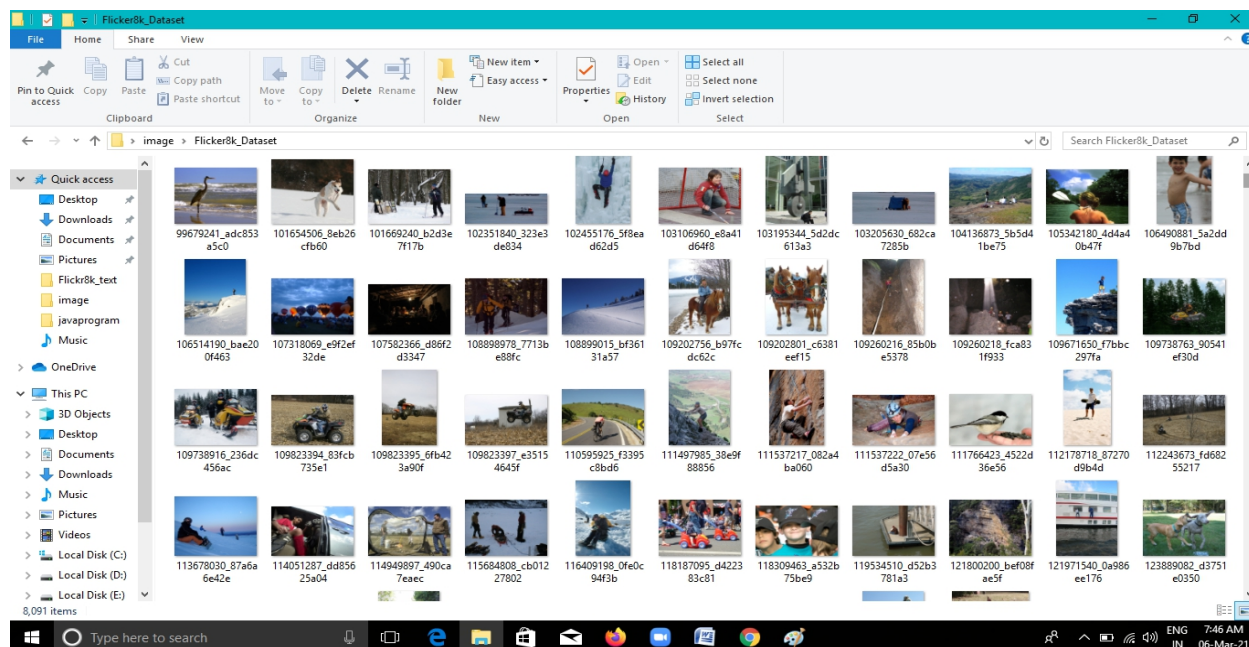


Fig 4.4 Dataset collection

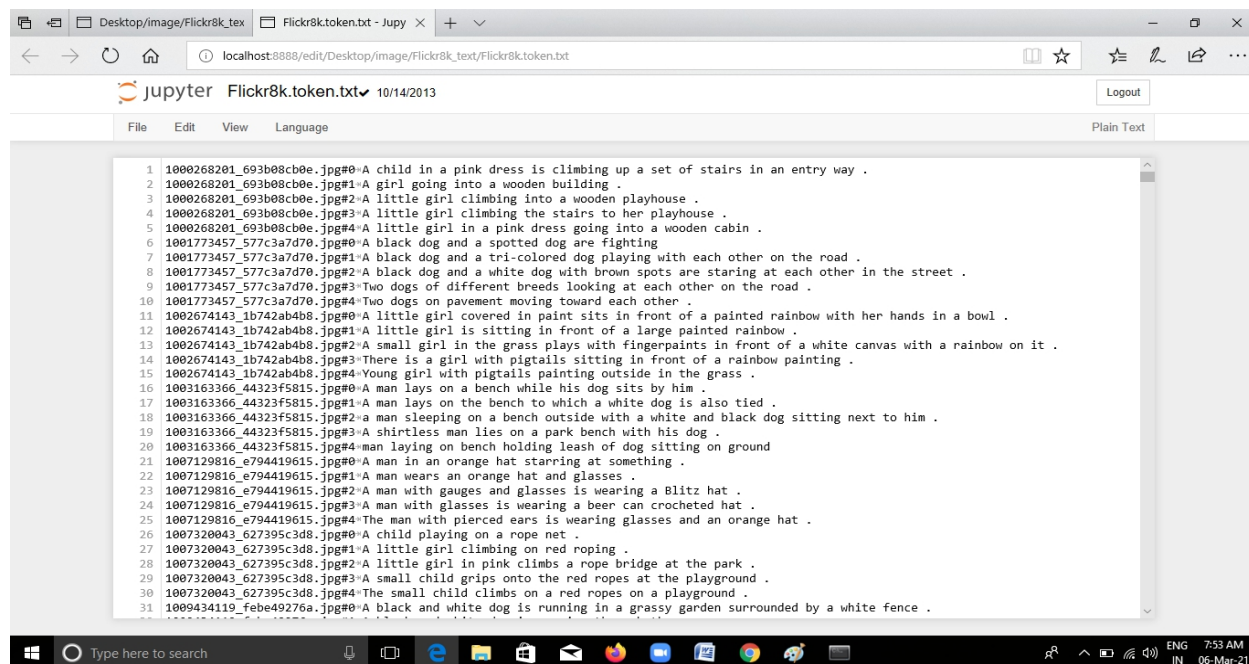
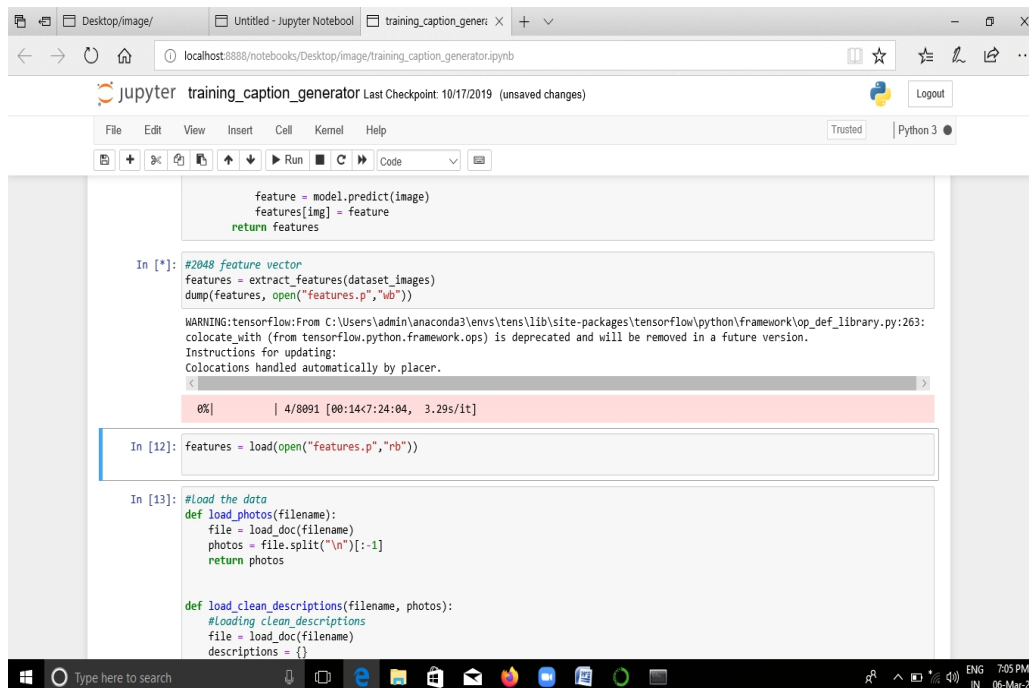


Fig 4.5 image with description

2.EXTRACTING FEATURE VECTOR:

This technique is also called transfer learning, we don't have to do everything on our own, we use the pre-trained model that have been already trained on large data set and extract the features from these models and use them for our tasks. We are using the Xception model which has been trained on image net data set that had some sort of class to classify. We can directly import this model from the keras application. and for this process we have to connect our system with internet and then the weights get automatically downloaded. the xception model is build for image net so that we have to do some changes for integrating with our model, by removing the last classification layer and get the two thousand fourty eight vector. And this module extract features from all the images and map image names with their respective features array. then it will dump the features dictionary into a file. And this process will take lot of time depending on our system. for me it took almost two hours and fifty minutes.



```
feature = model.predict(image)
features[img] = feature
return features

In [*]: #2048 feature vector
features = extract_features(dataset_images)
dump(features, open("features.p", "wb"))

WARNING:tensorflow:From C:\Users\admin\anaconda3\envs\tens\lib\site-packages\tensorflow\python\framework\op_def_library.py:263:
colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.

0% | 4/8091 [00:14<7:24:04, 3.29s/it]

In [12]: features = load(open("features.p", "rb"))

In [13]: #Load the data
def load_photos(filename):
    file = load_doc(filename)
    photos = file.split("\n")[:-1]
    return photos

def load_clean_descriptions(filename, photos):
    #loading clean descriptions
    file = load_doc(filename)
    descriptions = {}
```

Fig 4.6 Extraction of features

3.LOADING DATASET FOR TRAINING :

In this step we are performing some operation for giving those processed data to the training part of the program, first loading the Dataset text files for performing various operation and this function will return the list of images name.Next function will create a dictionary that contains the caption for each photo from the list of photos.After obtaining this we append the start and end keyword for the caption .so that the long short term memory will indentify the initial and end part of the caption.Next function will create a dictionary for images and their feature vector which has been extracted during the extraction process using the xception model.

4. TOKENIZING VOCABULARY :

Contains tokens mapped with an index value and The second phase brings RNN into the picture for 'decoding' the process vector inputs generated by the CNN module. For initiating the task for captions, the RNN model needs to be trained with a relevant data set. It is essential to train the RNN model for predicting the next word in the sentence. However, training the model with strings is ineffective without definite numerical alphas values. Post tokenization, the last phase of the model is triggered using LSTM. This step requires an embedding layer for transforming each word into the desired vector and eventually pushed for decoding. With LSTM, the RNN model must be able to remember spatial information from the input feature vector and predict the next word. Now with LSTM performing its tasks, the final output is generated by calling the data_generator function.

5.CREATION OF DATAGENERATOR:

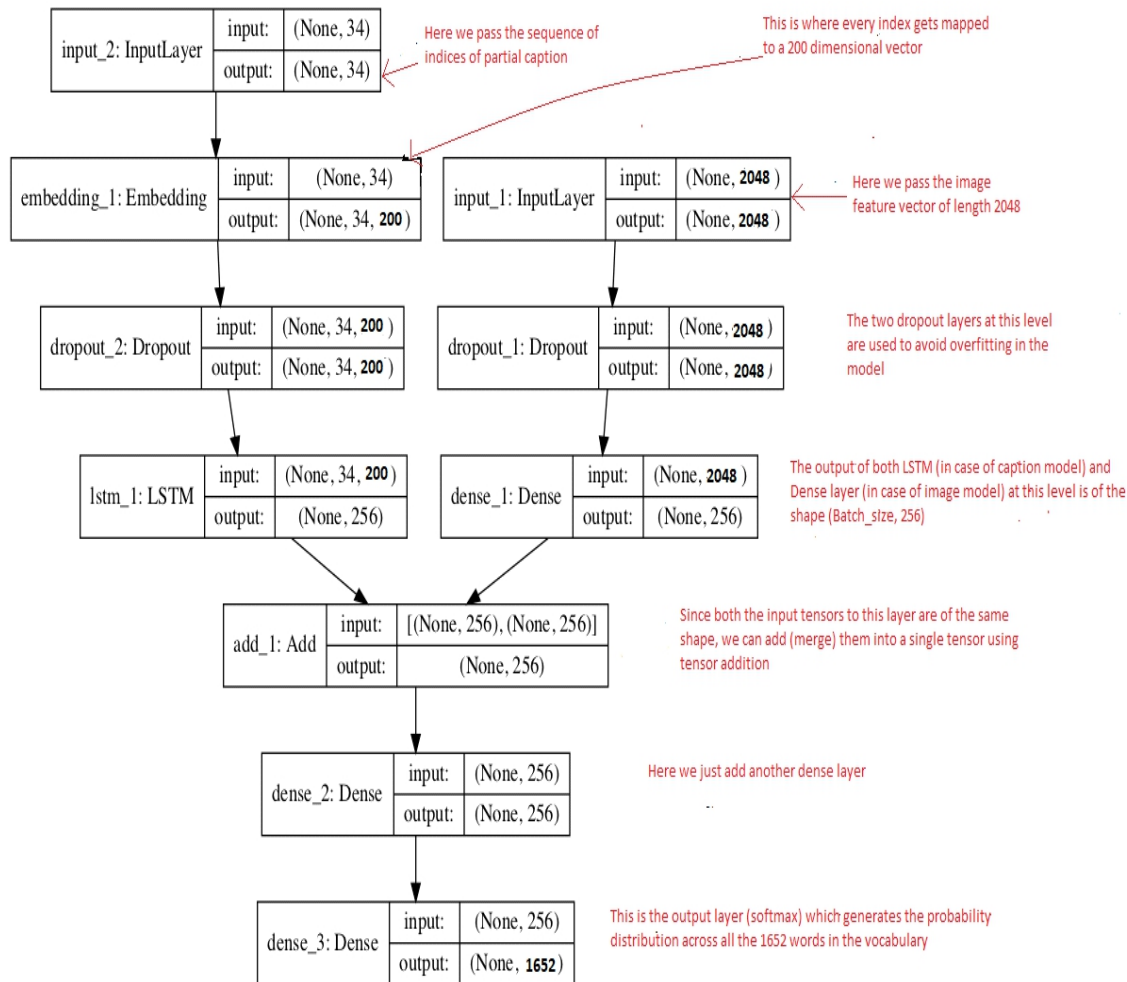
First we have to find what the input and output of the model will resembles. To make this task into a supervised learning task, for this model we have to fed the input and output and then the model will predict action with the help of previous predictions for example, you tube suggestion what the user interested and has seen before based on that the machine will give suggestion or recommended video for entertaining. It

defined by the uses of labeled data set to train algorithm to generate reasonable predictions for the response to new data. And supervised learning uses classification and regression technique to develop predictive model. The input to our model is [k1,k2] and the output will be z,where k1 is the 2048 feature vector of the image,k2 is the input text sequence and z is the output text sequence that the model has to predict.

K1 (feature vector)	K2 (text sequence)	Z (word to predict)
Feature	Start,	Two
Feature	Start, two	Dogs
Feature	Start, two, dogs	Drink
Feature	Start, two, dogs, drink	Water
Feature	Start ,two, dogs, drink, water	End

DEFINING MODEL:

For defining the structure of the model, we are using the keras model from functional API. It will consist of three major parts .Feature extracted from the image has a size of 2048,with a dense layer ,and we are reducing the dimensions to two hundred fifty six nodes. In the sequence processor, an embedding layer will handle the textual input, followed by the long short term memory layer. Then passes through the decoder layer it performs, merging the output from the above two layers we will process it with the help of dense layer to make the final prediction. The ending layer will contains the number of nodes equal to our vocabulary size.



6. TRAINING THE MODEL:

We have trained our model using the method called `fit_generator()`. By using this method training process completed for our project. We trained our model with various epochs value one, two, three, and eight. And after the training process is completed we saved the trained model to folder. By using 1 epoch value we are getting the eighty percentage accuracy of description. While training it will take lots of time depending upon the system compatibility for me it took 16 hours for training model using three epoch value. Gradually the loss value is also decreasing with respect to the each iteration.

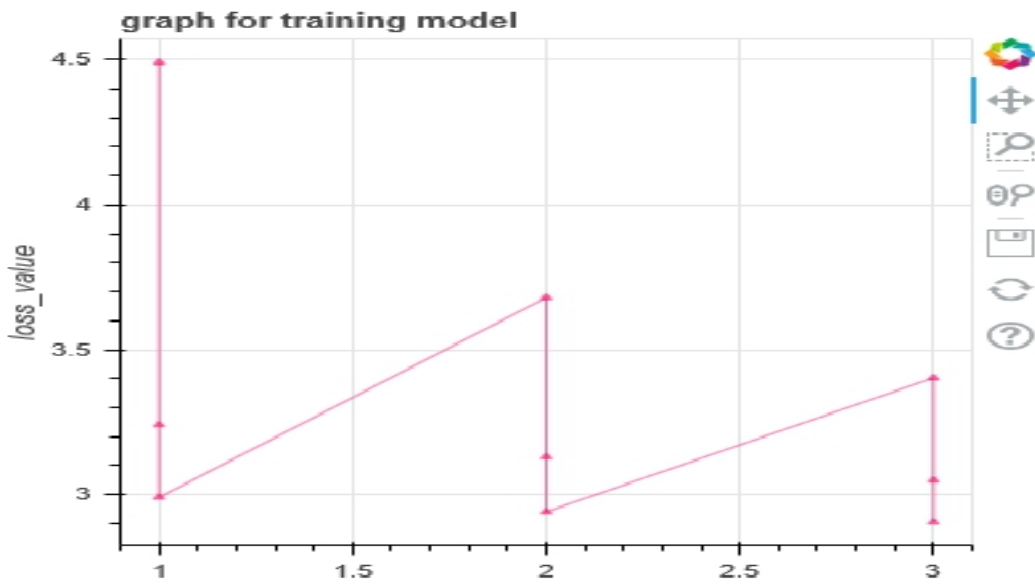


Fig 4.7epoch with loss value

Program :

From bokeh.plotting import figure

From bokeh.io import output_files,show

P=figure(width=500,height=500)

X=[1,1,1,2,2,2,3,3,3]

Y=[4.4915,3.2402,2.9901,3.6769,3.1292,2.9404,3.4024,3.0486,2.9038]

p.title.text="graph for training model"

p.xaxis.axis_label="epoch"

p.yaxis.axis_label="loss value"

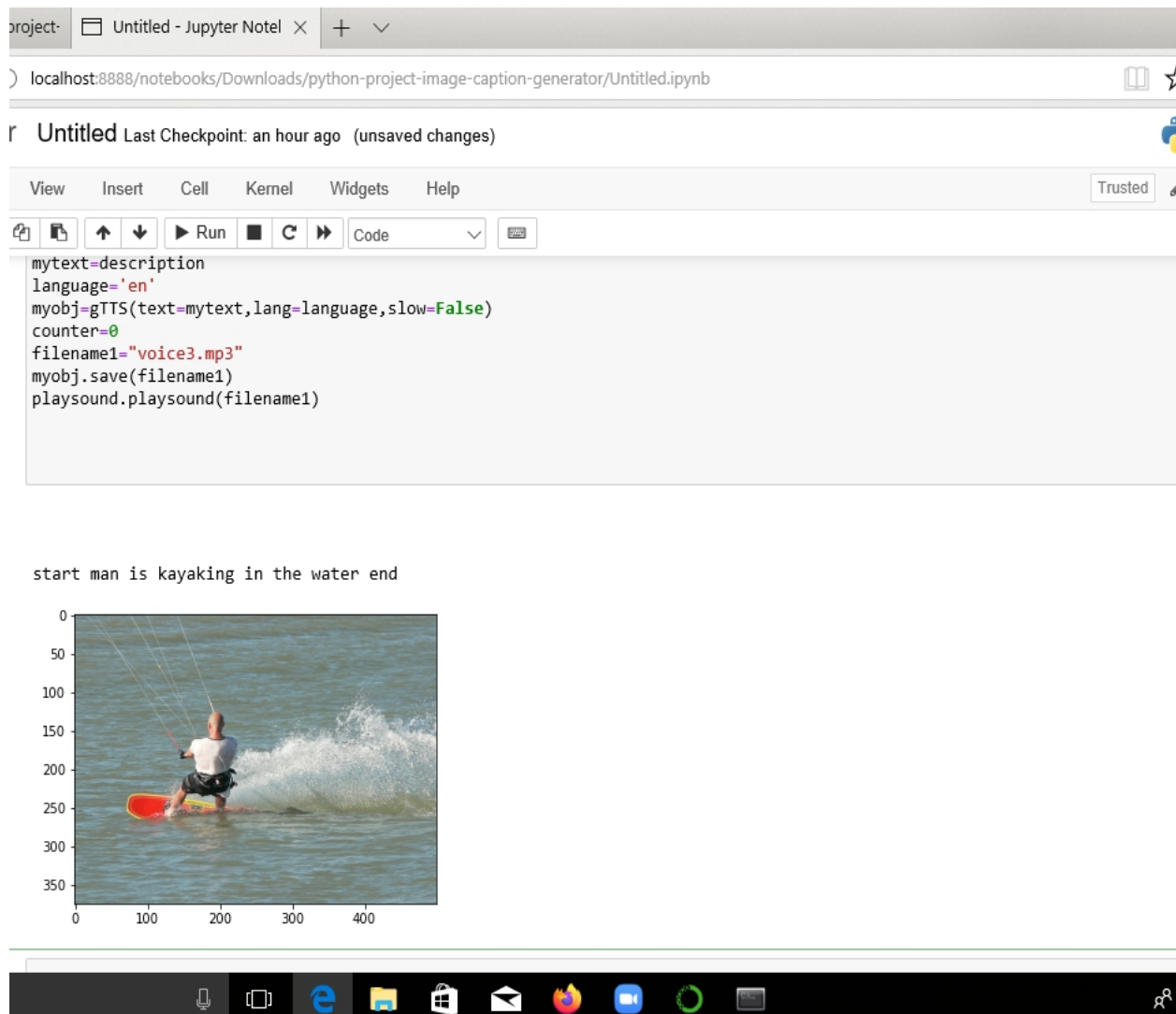
p.line(X,Y,color="pink",alpha=0.5)

output_files ("graph.html")

show(p)

8.TESTING:

For testing our model we are pass the image path and then it pass on to the various function and then it will generate a description for an user given once the description is create it goes to the audio creation function then it will readout the caption that generated during the process.



The screenshot displays a Jupyter Notebook window titled "Untitled - Jupyter Note" with a tab labeled "project-". The address bar shows the local path: "localhost:8888/notebooks/Downloads/python-project-image-caption-generator/Untitled.ipynb". The notebook interface includes a menu bar (View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The code cell contains the following Python code:

```
mytext=description
language='en'
myobj=gTTS(text=mytext,lang=language,slow=False)
counter=0
filename1="voice3.mp3"
myobj.save(filename1)
playsound.playsound(filename1)
```

The output of the code cell shows the text "start man is kayaking in the water end" followed by a plot of an image. The image shows a person kayaking on a body of water, with a vertical axis on the left ranging from 0 to 350 and a horizontal axis at the bottom ranging from 0 to 400.

CHAPTER 5

RESULT AND FUTURE WORK

5.1 RESULT ANALYSIS:

we have compiled some aspects of the image caption generation task, discussed the model framework proposed in recent years to solve the description task. Focused on the algorithmic essence of different attention mechanisms, and summarized how the attention mechanism is applied. We summarize the large datasets and evaluation criteria commonly used in practice. So the obtained captions are readout in the voice format and it help full for the visually unpaired people to recognize. The variety of image captioning system is available today, experimental results shows that this task still has better performance system and improvement. It mainly faces the following challenges, how to generate complete natural language sentences like a human being, and how to make the generated sentences grammatically correct and make the caption semantics as clear as possible and consistent with the given image content.

5.2 FUTURE WORK:

We are planning to implement caption generation for the dynamic video capturing ,it's a tedious and more time consuming process ,to overcome this kinds of stuff we are planning to implement these concepts with the help of deep learning by using the most strong algorithm and planning to converting the caption into the different possible languages. Because live captioning is the required and most useful process for this technical world.

CHAPTER 6

CONCLUSION

6.1.SUMMARY:

The development of the image description system may help the visually impaired people who are unable to comprehend visuals. And also adding a sound to those text, spoken descriptions can be read out enabling them to get a better sense of their surroundings. We have implemented this model using the convolutional neural network and long short term memory to overcome the limitations of traditional recurrent neural network. And this model cannot predict the words that are out of its vocabulary. So we are using small amount of images.

6.2.APPENDIX:

A.SAMPLE CODE:

```
import string

import numpy as np

from PIL import Image

import os

from pickle import dump, load

import numpy as np


from keras.applications.xception import Xception, preprocess_input

from keras.preprocessing.image import load_img, img_to_array

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad_sequences
```

```

from keras.utils import to_categorical

from keras.layers.merge import add

from keras. Models import Model, load_model

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

# small library for seeing the progress of loops.

from tqdm import tqdm as tqdm

tqdm().pandas()

# Loading a text file into memory

def load_doc(filename):

    # Opening the file as read only

    file = open(filename, 'r')

    text = file.read()

    file.close()

    return text

# get all imgs with their captions

def all_img_captions(filename):

    file = load_doc(filename)

    captions = file.split('\n')

    descriptions = {}

    for caption in captions[:-1]:

        img, caption = caption.split('\t')

```

```

if img[:-2] not in descriptions:

    descriptions[img[:-2]] = [caption]

else:

    descriptions[img[:-2]].append(caption)

return descriptions

##Data cleaning- lower casing, removing punctuation and words containing numbers

def cleaning_text(captions):

    table = str.maketrans(",","string. punctuation")

    for img,caps in captions. items():

        for i,img_caption in enumerate(caps):

            img_caption.replace("-", " ")

            desc = img_caption.split()

            #converts to lower case

            desc = [word.lower() for word in desc]

            #remove punctuation from each token

            desc = [word.translate(table) for word in desc]

            #remove hanging 's and a

            desc = [word for word in desc if(len (word)>1)]

            #remove tokens with numbers in them

            desc = [word for word in desc if(word.isalpha())]

            #convert back to string

```

```

        img_caption = ' '.join(desc)

        captions[img][i]= img_caption

    return captions

def text_vocabulary(descriptions):

    # build vocabulary of all unique words

    vocab = set()

    for key in descriptions.keys():

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

    return vocab

#All descriptions in one file

def save_descriptions(descriptions, filename):

    lines = list()

    for key, desc_list in descriptions.items():

        for desc in desc_list:

            lines.append(key + '\t' + desc )

    Data = "\n".join(lines)

    File = open(filename,"w")

    file.write (data)

    file.close ()

dataset_text = r"C:\Users\admin\Desktop\image\Flickr8k_text"

```

```

dataset_images = r"C:\Users\admin\Desktop\image\Flicker8k_Dataset"

#we prepare our text data

filename = dataset_text + "/" + "Flickr8k.token.txt"

#loading the file that contains all data

#mapping them into descriptions dictionary img to 5 captions

descriptions = all_img_captions(filename)

print("Length of descriptions =" ,len(descriptions))

#cleaning the descriptions

clean_descriptions = cleaning_text (descriptions)

#building vocabulary

Vocabulary = text_vocabulary(clean_descriptions)

print("Length of vocabulary = ", len(vocabulary))

#saving each description to file

save_descriptions(clean_descriptions, "descriptions.txt")

def extract_features(directory):

    model = Xception( include_top=False, pooling='avg' )

    features = {}

    for img in tqdm(os.listdir(directory)):

        filename = directory + "/" + img

        image = Image.open(filename)

        image = image.resize((299,299))

```

```

    image = np.expand_dims(image, axis=0)

    #image = preprocess_input(image)

    image = image/127.5

    image = image - 1.0

    feature = model.predict(image)

    features[img] = feature

    return features

filename = dataset_text + "/" + "Flickr_8k.trainImages.txt"

#train = loading_data(filename)

train_imgs = load_photos(filename)

train_descriptions = load_clean_descriptions("descriptions.txt", train_imgs)

train_features = load_features(train_imgs)

def dict_to_list(descriptions):

    all_desc = []

    for key in descriptions.keys():

        [all_desc.append(d) for d in descriptions[key]]

    return all_desc

#creating tokenizer class

#this will vectorise text corpus

```

```

#each integer will represent token in dictionary

from keras.preprocessing.text import Tokenizer

def create_tokenizer(descriptions):

    desc_list = dict_to_list(descriptions)

    tokenizer = Tokenizer()

    tokenizer.fit_on_texts(desc_list)

    return tokenizer

tokenizer = create_tokenizer(train_descriptions)

dump(tokenizer, open('tokenizer.p', 'wb'))

vocab_size = len(tokenizer.word_index) + 1

vocab_size

def data_generator(descriptions, features, tokenizer, max_length):

    while 1:

        for key, description_list in descriptions.items():

            #retrieve photo features

            feature = features[key][0]

            input_image, input_sequence, output_word = create_sequences(tokenizer,
max_length, description_list, feature)

            yield [(input_image, input_sequence), output_word]

def create_sequences(tokenizer, max_length, desc_list, feature):

```

```

X1, X2, y = list(), list(), list()

# walk through each description for the image

for desc in desc_list:

    # encode the sequence

    seq = tokenizer.texts_to_sequences([desc])[0]

    # split one sequence into multiple X,y pairs

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

        # split into input and output pair

        in_seq, out_seq = seq[:i], seq[i]

        # pad input sequence

        in_seq = pad_sequences([in_seq], maxlen=max_length)[0]

        # encode output sequence

        out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]

        # store

        X1.append(feature)

        X2.append(in_seq)

        y.append(out_seq)

    return np.array(X1), np.array(X2), np.array(y)

from keras.utils import plot_model

# define the captioning model

```

```

def define_model(vocab_size, max_length):

    # features from the CNN model squeezed from 2048 to 256 nodes

    inputs1 = Input(shape=(2048,))

    fe1 = Dropout(0.5)(inputs1)

    fe2 = Dense(256, activation='relu')(fe1)

    # LSTM sequence model

    inputs2 = Input(shape=(max_length,))

    se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)

    se2 = Dropout(0.5)(se1)

    se3 = LSTM(256)(se2)

    # Merging both models

    decoder1 = add([fe2, se3])

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

    outputs = Dense(vocab_size, activation='soft-max')(decoder2)

    # tie it together [image, seq] [word]

    model = Model(inputs=[inputs1, inputs2], outputs=outputs)

    model.compile(loss='categorical_crossentropy', optimizer='adam')

    # summarize model

```

```

print(model.summary())

#plot_model(model, to_file='model.png', show_shapes=True)

return model

# train our model

print('Dataset: ', len(train_imgs))

print('Descriptions: train=', len(train_descriptions))

print('Photos: train=', len(train_features))

print('Vocabulary Size:', vocab_size)

print('Description Length: ', max_length)

model = define_model(vocab_size, max_length)

epochs = 3

steps = len(train_descriptions)

# making a directory models to save our models

os.mkdir("models")

for i in range(epochs):

    generator = data_generator(train_descriptions, train_features, tokenizer,
max_length)

    model.fit_generator(generator, epochs=3, steps_per_epoch= steps, verbose=1)

    model.save("models/model_9" + str(i) + ".h5")

```

B. OUTPUT SCREEN SHOTS:

The screenshot displays a Jupyter Notebook environment. The browser address bar shows `localhost:8888/notebooks/Desktop/image/Untitled.ipynb`. The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help), a toolbar with icons for file operations and execution, and a status bar indicating "Not Trusted" and "Python 3".

The code cell contains the following Python code:

```
# here we have marked slow=False, which tells
# the module that the converted audio should
# have a high speed
myobj = gTTS(text=mytext, lang=language, slow=False)

# Saving the converted audio in a mp3 file named
# welcome
counter = 0
myobj.save(r"C:\Users\admin\Desktop\image\welcome23.mp3")
playsound(r"C:\Users\admin\Desktop\image\welcome23.mp3")
# Playing the converted file
# os.system(r'mpg321 r"C:\Users\RETECH\Desktop\image\welcome.mp3welcome.mp3"')
```

Below the code cell, the output shows a video player interface. The title is "boat in the water end". The video frame shows two people in a small boat on a body of water. The video player includes a progress bar and a volume control icon.

Downloads/python-project: Untitled - Jupyter Notebook

localhost:8888/notebooks/Downloads/python-project-image-caption-generator/Untitled.ipynb

jupyter Untitled Last Checkpoint: Last Tuesday at 10:32 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
mytext=description
language='en'
myobj=gTTS(text=mytext,lang=language,slow=False)
counter=0
filename1="voice0001.mp3"
myobj.save(filename1)
playsound.playsound(filename1)
```

the basketball player is wearing red and white uniform and playing basketball end



Windows taskbar: Type here to search, 6:06 AM, 25-Feb-21

Downloads/python-project: Untitled - Jupyter Notebook

localhost:8888/notebooks/Downloads/python-project-image-caption-generator/Untitled.ipynb

jupyter Untitled Last Checkpoint: Last Tuesday at 10:32 PM (unsaved changes)

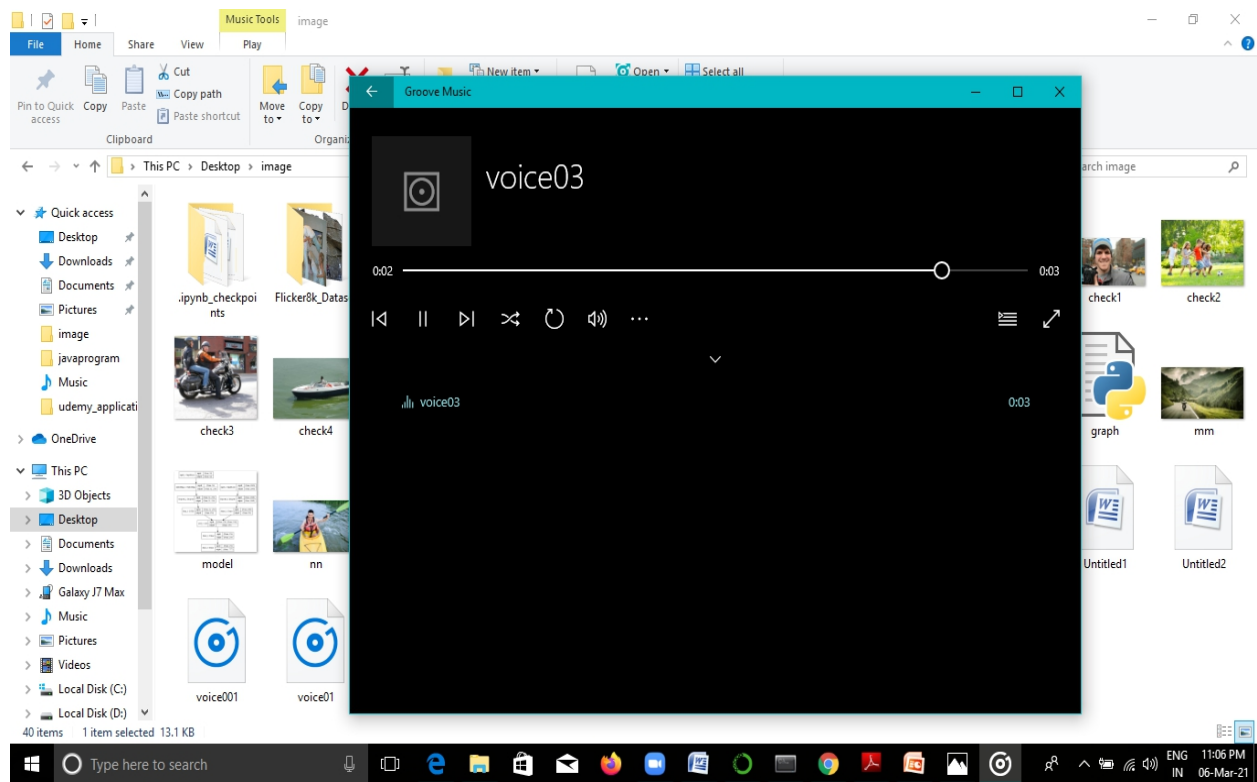
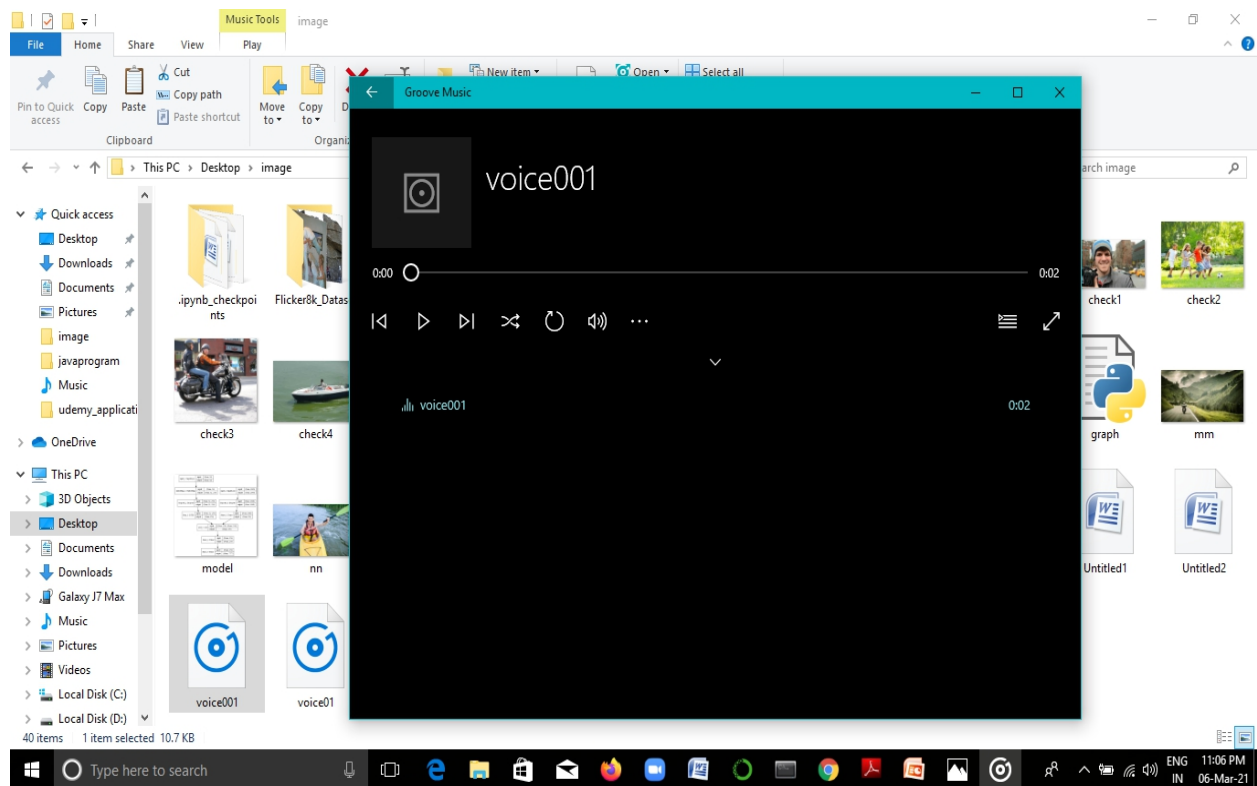
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

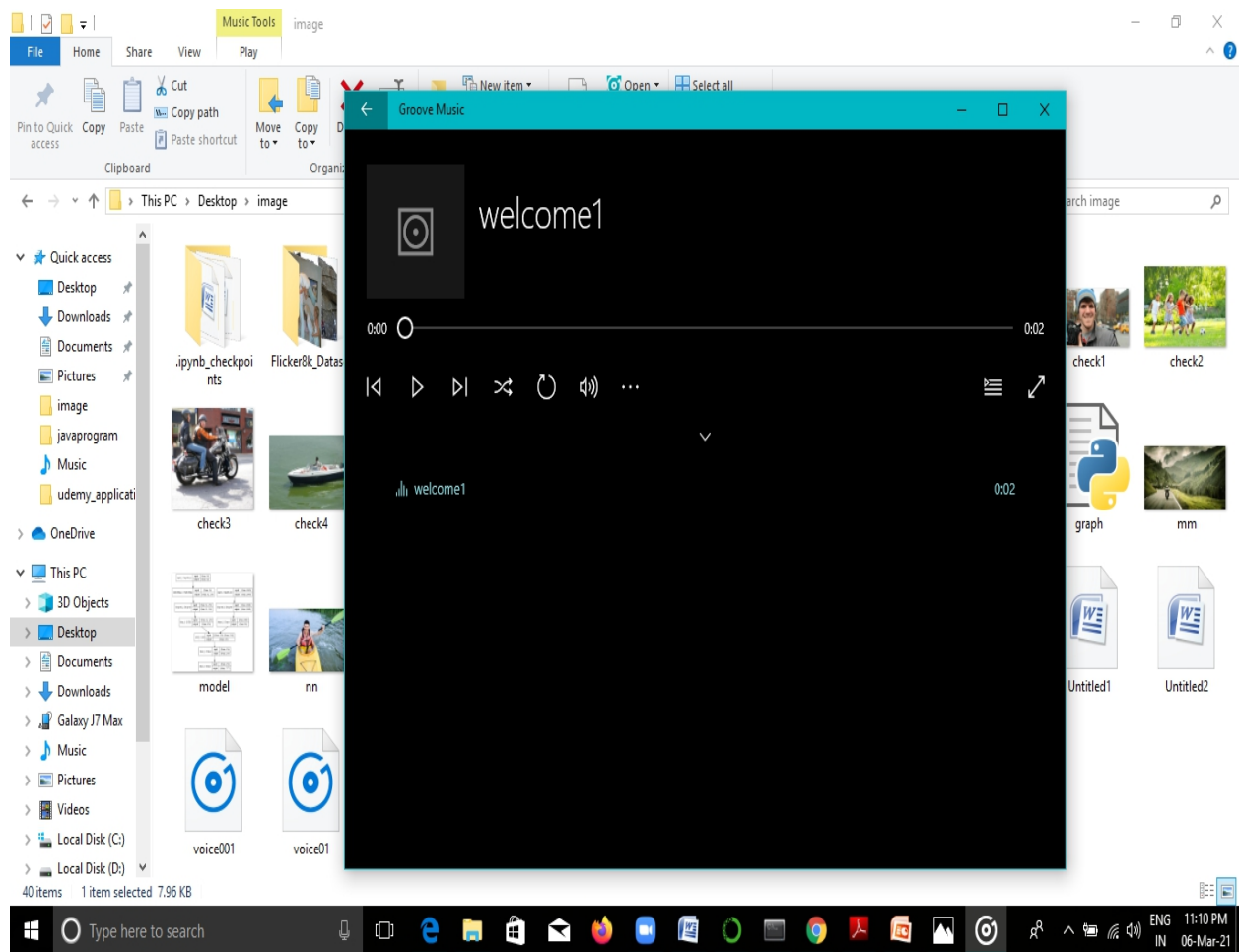
```
mytext=description
language='en'
myobj=gTTS(text=mytext,lang=language,slow=False)
counter=0
filename1="voice0001.mp3"
myobj.save(filename1)
playsound.playsound(filename1)
```

dirt bike rider is riding through the woods end

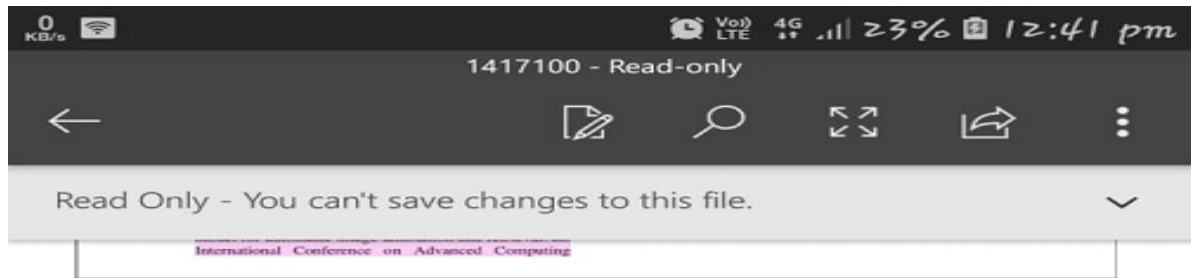


Windows taskbar: Type here to search, 5:59 AM, 25-Feb-21





C.PUBLICATION WITH PLAGIARISM REPORT:



amiraakash99@gmail.com.docx

ORIGINALITY REPORT

4%	4%	4%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Gautham Shekar, S. Revathy, Ediga Karthick Goud. "Malaria Detection using Deep Learning", 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), 2020 Publication	3%
2	Gargi Srivastava, Rajeev Srivastava. "Chapter 8 A Survey on Automatic Image Captioning", Springer Science and Business Media LLC, 2018 Publication	1%

Exclude quotes Off
Exclude bibliography Off

Exclude matches Off

REFERENCES:

1. Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. "Show, attend and tell: Neural image caption generation with visual attention."
2. Chen, Xinlei, and C. Lawrence Zitnick. "Mind's eye: A recurrent visual representation for image caption generation."
3. Bai, Shuang, and Shan An. "A survey on automatic image caption generation." .
4. King horn, Philip, Li Zhang, and Ling Shao. "A region-based image caption generator with refined descriptions."
5. Aghasi Poghosyan, Hakob Sarukhanyan :long short term Memory With Read Only Unit in Neural Image Caption Generator .
6. Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan. "Show and tell: A neural image caption generator."
7. Tan, Ying Hua, and Chee Seng Chan. "Phrase-based image caption generator with hierarchical LSTM network."
8. Wang, Haoran, Yue Zhang, and Xiaosheng Yu. "An Overview of Image Caption Generation Methods."
9. Rahman, Tanzila, Bicheng Xu, and Leonid Sigal. "Watch, listen and tell: Multi-modal weakly supervised dense event captioning."
10. Mathews, Alexander, Lexing Xie, and Xuming He. "Semstyle: Learning to generate stylised image captions using unaligned text."
11. Rawale, Shubham, Megha Ghotkar, Krishna Sonavane, Paras Surve, Shraddha Khonde, and Deepali Patil. "Image Captioning Generator System With Caption to speech conversion mechanism".
12. Wu, Shiqi, Xiangrong Zhang, Xin Wang, Chen Li, and Licheng Jiao. "Scene Attention Mechanism for Remote Sensing Image Caption Generation.
13. Hu, Yu, Yongkang Wong, Wentao Wei, Yu Du, Mohan Kankanhalli, and Weidong Geng. "A novel attention-based hybrid CNN-RNN architecture for sEMG-based gesture recognition."

14.Sindu, S., and R. Kousalya. "Recurrent Neural Network for Content Based Image Retrieval Using Image Captioning Model."

15.Alsharid, Mohammad, Harshita Sharma, Lior Drukker, Pierre Chatelain, Aris T. Papageorghiou, and J. Alison Noble. "Captioning ultrasound images automatically."