



*Project Report on*

## **IMAGE CAPTION GENERATOR**

*Submitted in Partial Fulfilment of the Requirement*

*for the IV Semester MCA Academic Minor Project – I*

**18MCA46**

**MASTER OF COMPUTER APPLICATIONS**

**By**

<b>USN</b>	<b>STUDENT NAME</b>
<b>1RD19MCA09</b>	<b>SNEHAL M HUKKERI</b>
<b>1RD19MCA10</b>	<b>SONU KUMAR GAUTAM</b>

**Under the Guidance of**

**Shaila. H. Koppad**

**Assistant Professor**

Department of Master of Computer Applications

RV College of Engineering<sup>®</sup>, Mysuru Road

RV Vidyanikethan Post, Bengaluru – 560059

June -2021



## DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS



### CERTIFICATE

This is to certify that the project entitled **“Image Caption Generator”** submitted in partial fulfillment of Minor Project-I (18MCA46) of IV Semester MCA is a result of the bonafide work carried out by Snehal M Hukkeri (1RD19MCA09) and Sonu Kumar Gautam(1RD19MCA10), during the Academic year 2020-21.

Shaila. H. Koppad  
Assistant Professor  
Department of MCA,  
RV College of Engineering<sup>®</sup>

Dr. Andhe Dharani  
Professor and Director  
Department of MCA,  
RV College of Engineering<sup>®</sup>



RV Educational Institutions<sup>®</sup>  
RV College of Engineering<sup>®</sup>

Autonomous  
Institution Affiliated  
to Visvesvaraya  
Technological  
University, Belagavi

Approved by AICTE,  
New Delhi

*Go, change the world*

### **UNDERTAKING BY THE STUDENT**

We, Snehal M Hukkeri (1RD19MCA09) and Sonu Kumar Gautam(1RD19MCA10) hereby declare that the Minor project-I “Image Caption Generator” is carried out and completed successfully by us and is our original work.

#### **SIGNATURE**

Snehal M Hukkeri

Sonu Kumar Gautam

## Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned the efforts with success.

I would like to profoundly thank Management of, RV College of Engineering® for providing such a healthy environment for the successful completion of project.

I would like to express my thanks to the Principle **Dr.K.N.Subramanya**, RV College of Engineering® for his encouragement that motivated me for the successful completion of Project.

It gives me immense pleasure to thank **Dr,Andhe Dhrani** Professor and Director ,Department of MCA , RV College of Engineering® for her constant support and encouragement.

I would like to express my deepest sense of gratitude to guide Prof. Shaila.H.Koppad, Assistant Professor, Department of MCA, RV College of Engineering® for her constant support and guidance.

Finally, I would like to express my sincere thanks to all staff members of MCA Department for their valuable guidance and support.

Snehal M Hukkeri and Sonu Kumar Gautam

Master of Computer Applications

RV College of Engineering®

Bengaluru-59

## Abstract

Today lot of images are being clicked because of the digital era. People are clicking pictures and uploading on various social media platforms but they lack when they have to provide relevant caption to their images. It is very difficult for outwardly disabled individuals better comprehend the substance of pictures on the web. Image Caption Generator could help individuals with outwardly hindrance better comprehend visual information sources, along these lines going about as a right hand or a guide, this can help them by providing captions. It will also help people to generate relevant caption for their images that they would be uploading on social media. It could be used in domains such as Hospitals, Educational Institutions to generate caption from the images

Image Caption generator is implemented using CNN (Convolutional Neural Networks) and LSTM (Long short term memory). The image features are extracted from Xception which is a CNN model trained on the image net dataset and then features are feed into the LSTM model which is responsible for generating the image captions. Libraries used are: tensor flow, keras, Pillow, numpy, tqdm and jupyterlab. In the initial phase the required dataset is collected, then preprocessing of the data extracting the feature vector from all images, Loading dataset for Training the model, Tokenizing the vocabulary ,Create Data generator, Defining the CNN-RNN model, Training the model ,Testing of model is done.

Outcome is, for the image provided by the user relevant caption is generated by the model in a natural language like English which can further be used. User will be giving the image to the model, the image features will be extracted using deep learning models and caption is be generated.

## CONTENTS

chapter	Title	Page no
<b>1</b>	<b>Introduction</b>	
	1.1 Project Description	1
<b>2</b>	<b>Literature Review</b>	
	2.1 Literature Survey	
	2.2 Existing and Proposed System	
	2.3 Tools and Technologies used	
	2.4 Hardware and Software Requirements	
<b>3.</b>	<b>Software Requirement Specifications</b>	
	3.1 Introduction	
	3.2 General Description	
	3.3 Functional Requirement	
	3.4 External Interfaces Requirements	
	3.5 Non Functional Requirements	
	3.6 Design Constraints	
<b>4</b>	<b>System Design</b>	
	4.1 System Perspective /Architectural Design	
	4.2 Context Diagram	
<b>5</b>	<b>Detailed Design</b>	
	5.1 System Design	
	5.2 Detailed design	
<b>6</b>	<b>Implementation</b>	
	6.1 Code Snippets / PDL	
	6.2 Implementation	
<b>7</b>	<b>Software Testing</b>	
	7.1 Test cases	
	7.2 Testing and Validations	
<b>8</b>	<b>Conclusion</b>	
<b>9</b>	<b>Future Enhancements</b>	
	<b>Bibliography</b>	

## LIST OF TABLES

Table No	Table Label	Page No
7.1.1	Test cases and Results for Unit Testing	
7.1.2	Test cases and Results for Integration Testing	
5.1.1.1	Activity Diagram	
5.1.1.2	Sequence Diagram	
5.1.2.1	Data Flow Diagram Level 0	
5.1.2.2	Data Flow Diagram Level 1	
5.1.2.3	Data Flow Diagram Level 2	
6.2.1	Result of CNN-RNN model	
6.2.2	Starting server	
6.2.3	User Interface	
6.2.4	Caption of uploaded image	
7.2.1	Importing required libraries	
7.2.2	Data Cleaning	
7.2.3	Output of cleaning descriptions.txt	
7.2.4	Feature Extraction: features stored in features.p	
7.2.5	Loading dataset for training model	
7.2.6	Tokenizing vocabulary	
7.2.7	Data generator	
7.2.8	CNN-RNN Model	
7.2.9	Training model	
7.2.9.1	Training model output	
7.2.9.2	Training output	

7.2.10	Clicking Generate Caption button without uploading image	
7.2.11	User Feedback Form	
7.2.12	Response on Successful feedback	

## LIST OF FIGURES

Figure No	Figure Label	Page No
4.1.2	Block Diagram	
4.2	Context Diagram	



## Chapter 1: Introduction

Today lot of images are being clicked because of the digital era. People are clicking pictures and uploading on various social media platforms but they lack when they have to provide relevant caption to their images. It is very difficult for outwardly disabled individuals better comprehend the substance of pictures on the web. Image Caption Generator could help individuals with outwardly hindrance better comprehend visual information sources, along these lines going about as a right hand or a guide, this can help them by providing captions. It will also help people to generate relevant caption for their images that they would be uploading on social media. It could be used in domains such as Hospitals, Educational Institutions to generate caption from the images

Image Caption generator is implemented using CNN (Convolutional Neural Networks) and LSTM (Long short term memory).The image features are extracted from Xception which is a CNN model trained on the image net dataset and then features are feed into the LSTM model which is responsible for generating the image captions. Libraries used are: tensor flow, keras, Pillow, numpy, tqdm and jupyterlab.In the initial phase the required dataset is collected, then preprocessing of the data extracting the feature vector from all images, Loading dataset for Training the model, Tokenizing the vocabulary ,Create Data generator, Defining the CNN-RNN model, Training the model ,Testing of model is done.

## Chapter 2: Literature Review

### 2.1 Literature Survey

Marc Tanti,et al.,[1], The key property of these models is that the CNN image features are used to predictions of the best caption to describe the image. However, this can be done in different ways and the role of the RNN depends in large measure on the mode in which CNN and RNN are combined.

Lukasz Kaiser,et al.,[2], : Active memory model. The whole memory takes part in the computation at every step. Each element of memory is active and changes in a uniform way, e.g., using a convolution. Active memory is a natural choice for image models as they usually operate on a canvas. And indeed, recent works have shown that actively updating the canvas that will be used to produce the final results can be beneficial.

Vikram Mullachery,et al.,[3],Image captioning is a much more involved task than image recognition or classification, because of the additional challenge of recognizing the interdependence between the objects/concepts in the image and the creation of a succinct sentential narration.

KelvinXn,et al.,[4], use a long short-term memory (LSTM) network that produces a caption by generating one word at every time step conditioned on a context vector, the previous hidden state and the previously generated words.

Oriol Vinyals,et al.,[5], a generative model based on a deep recurrent architecture that combines recent advances in computer vision and machine translation and that can be used to generate natural sentences describing an image. The model is trained to maximize the likelihood of the target description sentence given the training image.

MD.Zakir Hossain,et al.,[6], comprehensive survey of deep learning for image captioning. First, we group the existing image captioning articles into three main categories :( 1) Template-based Image captioning, (2) Retrieval-based image captioning, and (3) Novel image caption generation. The categories are discussed briefly in Section 2. Most deep learning based image captioning methods fall into the category of novel caption generation. Therefore, we focus

only on novel caption generation with deep learning. Second, we group the deep learning-based image captioning methods into different categories namely (1) Visual space-based, (2) Multimodal space-based, (3) Supervised learning, (4) Other deep learning, (5) Dense captioning, (6) Whole scene- based, (7) Encoder-Decoder Architecture-based, (8) Compositional Architecture-based, (9) LSTM (Long Short-Term Memory) language model-based, (10) Others language model-based, (11) Attention-Based, (12) Semantic concept-based, (13) Stylized captions, and (12) Novel object-based image captioning.

Haoran Wang et al.,[7], Attention mechanism, stemming from the study of human vision, is a complex cognitive ability that human beings have in cognitive neurology. When people receive information, they can consciously ignore some of the main information while ignoring other secondary information. This ability of self-selection is called attention. In natural language processing, when people read long texts, human attention is focused on keywords, events, or entities. A large number of experiments have proved that the attention mechanism is applied in text processing. for example, machine translation abstract generation text understanding text classification visual captioning and other issues, the results achieved remarkable, and the following describes the application of different attention mechanism methods in the image description basic framework introduced in the second part, so that its effect is improved. In neural network models, the realization of the attention mechanism is that it allows the neural network to have the ability to focus on its subset of inputs (or features)—to select specific inputs or features. The main part of the attention mechanism is the following two aspects: the decision needs to pay attention to which part of the input; the allocation of limited information processing resources to the important part.

N. Komal Kumar et al.,[8], 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, Recurrent Neural Network (RNN) for human and objects attributes, RNN encoder and a fixed length RNN decoder system. The input image is subject to the feature extraction using the `feature_extraction` command to extract the features in that image and captions are generated using the `generate_desc` command which takes parameters such as model, tokenizer, input image and length as input.

Pranay Mathur et al.,[9], customize and optimize our encoder-decoder model to function in a real-time environment and to work on mobile devices. For this we use Google's numerical computation library, TensorFlow. The main advantage of using TensorFlow for any Deep Learning model is its data-flow graphs based computations, which basically comprise of nodes, representing mathematical operations and edges, representing the multidimensional data arrays (tensors) communicated between nodes. The flexible architecture of TensorFlow enabled us to deploy our model's computation on CPUs and GPUs and thus helped us utilize inherent parallelism in primitive operations and computations.

Longteng Guo et al.,[10], A single-model solution for multi-style image captioning (MSCap) is desired to generate visually grounded and any desired stylized captions for a given image, while multi-style captioning resources including images and multi-style captions are explored jointly for the single-model training. Typically, training such a model requires fully annotated collections of aligned image-stylized-caption pairs (paired data) for each style. However, it is quite expensive to collect such paired multi-style captioning collections, especially when the numbers of images and styles increase. Compared to annotating stylized captions for each image, it is much easier and cheaper to collect a corpus of stylized sentences without aligned images. Therefore, it is challenging but valuable to design a multi-style captioning model by exploring such unpaired multi-stylized data in addition to handily available factual image-caption paired data (*e.g.* MS COCO [22] dataset), which motivates our work.

Lakshminarasimhan Srinivasan et al.,[11], The Bleu metric is an algorithm for evaluating the performance of a machine translation system by grading the quality of text translated from one natural language to another

Sarthak Mehta[12], Picture subtitle generator could be an undertaking that includes PC vision and tongue preparing ideas to recognize the setting of an image and portray them during a tongue like English.

Convolutional Neural systems are particular profound neural systems which may process the data that has input shape kind of a 2D framework. Pictures are effectively spoken to as a 2D grid and CNN is inconceivably valuable in working with pictures

Grishma Sharma et al.,[13], have used VGG-16 (Visual Geometry Group) , which is a 16 convolutional layers model used for object recognition. For the second phase, we need to train our features with captions provided in the dataset. We are using two architectures LSTM (Long Short Term Memory) and GRU (Gated Recurrent Unit) for framing our sentences from the input images given. To get an estimation of which architecture is better we have used the BLEU (Bilingual Evaluation Understudy) score to compare the performances between LSTM and GRU.

Oriol Vinyals et al.,[14], The propose a neural and probabilistic framework to generate descriptions from images. Recent advances in statistical machine translation have shown that, given a powerful sequence model, it is possible to achieve state-of-the-art results by directly maximizing the probability of the correct translation given an input sentence in an “end-to-end” fashion – both for training and inference. These models make 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 de-sired output sentence. Thus, it is natural to use the same approach where, given an image (instead of an input sentence in the source language), one applies the same principle of “translating” it into its description.

Ilay Sutskever et al.,[15], a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector.

Kyunghyun cho et al., [16], propose a novel neural network model called RNN Encoder– Decoder that consists of two recurrent neural networks (RNN). One RNN en- codes a sequence of symbols into a fixed- length vector representation, and the other decodes the representation into another se- quence of symbols. The encoder and de- coder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. The performance of a statistical machine translation system is empirically found to improve by using the conditional probabilities of phrase pairs computed by the RNN Encoder–Decoder as an additional feature in the existing log-linear model. Qualitatively, we show that the proposed model learns a semantically and syntactically meaningful representation of linguistic phrases.

Dzmitry Bahdanau et al., [17], two types of models. The first one is an RNN Encoder–Decoder (RNNencdec, Cho *et al.*, 2014a), and the other is the proposed model, to which we refer as RNNsearch. We train each model twice: first with the sentences of length up to 30 words (RNNencdec-30, RNNsearch-30) and then with the sentences of length up to 50 word (RNNencdec-50, RNNsearch-50). The encoder and decoder of the RNNencdec have 1000 hidden units each.<sup>7</sup> The encoder of the RNNsearch consists of forward and backward recurrent neural networks (RNN) each having 1000 hidden units.

Kyunghyun Cho et al., [18], in the field of statistical machine translation (SMT), deep neural networks have begun to show promising results. (Schwenk, 2012) summarizes a successful usage of feedforward neural networks in the framework of phrase-based SMT system. Along this line of research on using neural networks for SMT, this paper focuses on a novel neural network architecture that can be used as a part of the conventional phrase-based SMT system.

Jia-Yu Pan et al., [19], this paper interested in the following problem: “Given a set of images, where each image is captioned with a set of terms describing the image content, find the association between the image features and the terms”. Furthermore, “with the association found, caption an unseen image”. Previous works caption an image by captioning its constituting regions, by a mapping from image regions to terms.

Alex Krizhevsky, et al., [20], architecture -the net contains eight layers with weights; the first five are convolutional and the remaining three are fully connected. The output of the last fully-connected layer is fed to a 1000-way softmax which produces a distribution over the 1000 class labels. Our network maximizes the multinomial logistic regression objective, which is equivalent to maximizing the average across training cases of the log-probability of the correct label under the prediction distribution.

## 2 Existing and Proposed System

Existing approaches of image caption generator are using a condition random field for linking image and text.

Proposed a method of finding the objects and attributes from an image and using it to generate text with a conditional random field (CRF). The CRF is traditionally used for a structured prediction such as text generation. The use of CRF has limitations in generating text in a coherent manner with proper placement of prepositions. The results have proper predictions of objects and attributes but fail at generating good descriptions. Creating captions using image ranking method to rank the images followed by generating captions. The high-level information extracted from the ranked images can be used to generate the text. More images that are available for ranking, the better the results will be. Retrieving captions can be achieved by connecting encoders of image and text through a latent space. The first model in the image is the full model used for training. Visual features can also be used to generate sentences, or vice-versa.

Image caption generator is a task that involves computer vision and natural language processing (NLP) concepts to recognize the context of an image and describe them in a natural language like English.

For the image provided by the user relevant caption will be generated by the model in a natural language like English which can further be used. User will be giving the image to the model, the image features will be extracted using deep learning models and relevant caption will be generated. Caption Generator could also be used in domains such as Hospitals, Educational Institutions to generate caption from the images.

Methodology adopted in the proposed system is Image Caption generator is implemented using CNN (Convolutional Neural Networks) and LSTM (Long short term memory). The image features are extracted from Xception which is a CNN model trained on the image net dataset and then features are feed into the LSTM model which is responsible for generating the image captions. Libraries used are: tensor flow, keras, Pillow, numpy, tqdm and jupyterlab. In the initial phase the required dataset is collected, then preprocessing of the data extracting the feature vector from all images, Loading dataset for Training the model, Tokenizing the vocabulary, Create Data generator, Defining the CNN-RNN model, Training the model, Testing of model is done.

## **2.2 Tools and Technologies used**

- IDE: Google Colab, Jupyter notebook, Visual Studio
- Web development Framework: Flask
- Deep Learning Framework: Keras, Tensorflow
- Web technologies: Bootstrap, HTML, CSS
- Database: MySQL

## **2.3 Hardware and Software Requirements**

- CPU with at least 8GB RAM
- 1TB hard disk
- 2GB AMD RADEON GRAPHICS(GPU)
- Active internet connection
- Google Colab / Jupyter notebook



## Chapter 3: Software Requirement Specifications

### 3.1 Introduction

Today lot of images are being clicked because of the digital era. People are clicking pictures and uploading on various social media platforms but they lack when they have to provide relevant caption to their images. It is very difficult for outwardly disabled individuals better comprehend the substance of pictures on the web.

Image Caption Generator could help individuals with outwardly hindrance better comprehend visual information sources, along these lines going about as a right hand or a guide, this can help them by providing captions. It will also help people to generate relevant caption for their images that they would be uploading on social media.

Image caption generator is a task that involves computer vision and natural language processing (NLP) concepts to recognize the context of an image and describe them in a natural language like English.

For the image provided by the user relevant caption will be generated by the model in a natural language like English which can further be used. User will be giving the image to the model, the image features will be extracted using deep learning models and relevant caption will be generated.

Image Caption Generator could also be used in domains such as Hospitals, Educational Institutions to generate caption from the images.

### 3.2 General Description

**Product Perspective:** Today lot of images are being clicked because of the digital era. People are clicking pictures and uploading on various social media platforms but they lack when they have to provide relevant caption to their images. It is very difficult for outwardly disabled individuals better comprehend the substance of pictures on the web. Image Caption Generator could help individuals with outwardly hindrance better comprehend visual information sources, along these lines going about as a right hand or a guide, this can help them by providing captions. It will also help people to generate relevant caption for their

images that they would be uploading on social media. It could be used in domains such as Hospitals, Educational Institutions to generate caption from the images

**Product Functions:** Application will take image uploaded by user and provide relevant caption.

**User Characteristics:** User should be able to upload image and get the appropriate caption. User can give feedback also.

**General Constraints:** Need dataset to train the model. The model is trained on dataset that has general images not domain specific.

**Assumptions and Dependencies:** Dataset is taken from Kaggle. This model highly dependent on training data.

### 3.3 Functional Requirement

In this application the user going to give the image to the model, the image features will be extracted using deep learning models and relevant caption will be generated.

#### 3.3.1. Feature Extraction Module

- Input -Image dataset
- Process – CNN (Convolution neural network ) is used for extracting features from the image. Each input image will pass it through a series of convolution layers with filters (Kernels), ReLu activation function and Pooling. Xception is pre trained network ,trained on more than a million images from the Image Net database. Xception is a convolutional neural network that is 71 layers deep. The pre-trained network can classify images into 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 299x299x3.
- Output - Features of image

### 3.3.2. Caption generation Module:

- Input -Feature Vector
- RNN (recurrent neural network) which is well suited for sequence prediction problems.

Process-LSTM will use the feature vector from CNN to generate a description of the image. It has proven itself effective from the traditional RNN by overcoming the limitations of RNN which had short term memory. LSTM can carry out relevant information throughout the processing of inputs and with a forget gate, it discards non-relevant information. The key to LSTMs is the cell state The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a point wise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”

- Output -Caption of the Image

### 3.3.3. Image handling Module

- Pillow is a Python Imaging Library (PIL), which adds support for opening, manipulating, and saving images. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including PNG, JPEG and JPG. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.
- The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

### 3.3.4. Full Stack Module:

Front end: Html, CSS, Bootstrap

Backend: Flask

Database: MySQL

- Input-User uploads and image on the Html page

- Process-That image is fetched from the webpage with the POST request through flask and image is provided into the model. Model processes that image and generates the caption that caption is rendered back with the image with the help of flask framework onto the web page.
- Output - User will get relevant caption for uploaded image.

### 3.4 External Interfaces Requirements

**User Interface:** User shall upload the image for which they want caption .It is done by selecting image from local device and then clicking on upload button. After uploading user shall click on Generate Caption button. Once after clicking the Generate button user will get appropriate caption.

**Software Interface:** Application requires database to store information about user when user gives feedback. This application is using MySQL database.

### 3.5 Non Functional Requirements

**Memory:** CPU with at least 8GB RAM

**Performance requirements:** GPU-(Tesla T4)

**Scalability:** For now the model is trained with 6000 images. The model is capable enough to handle larger dataset.

**Portability and compatibility :** Model is compatible with all the operating systems. Such as Microsoft Windows ,Linux etc.

## Chapter 4: System Design

### 4.1 System Perspective /Architectural Design

**4.1.1 Problem Specification:** Image caption generator is a task that involves computer vision and natural language processing (NLP) concepts to recognize the context of an image and describe them in a natural language like English.

For the image provided by the user relevant caption will be generated by the model in a natural language like English which can further be used. User will be giving the image to the model, the image features will be extracted using deep learning models and relevant caption will be generated.

Image Caption Generator could also be used in domains such as Hospitals, Educational Institutions to generate caption from the images.

**4.1.2 Block Diagram:** A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks.

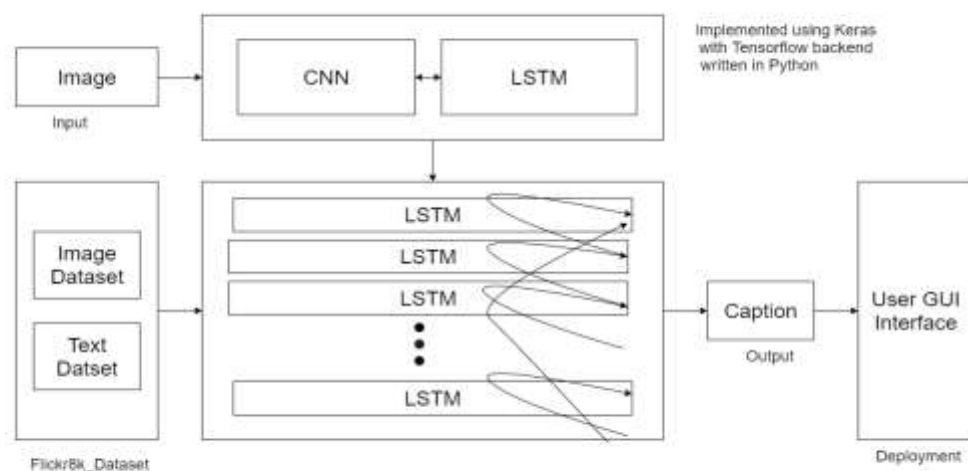


Fig 4.1.2 Block Diagram

1. User will upload the image
2. The CNN-RNN model is implemented using Keras and Tensorflow backend written in python.
3. Data from Flickr8\_k is loaded into Model and model is trained. Once after getting new image it will generate caption based on previous data on which it is trained.
4. Finally the Caption is generated will be viewed to the user on User Interface.

**Data definition/Dictionary:** **CNN**-Convolutional Neural Network, **LSTM**-Long Short Term Memory, Computer Vision, **NLP**-Natural Language Processing, **RNN** -Recurrent Neural Network

**Module specification:** Feature Extraction Module, Caption generation Module, Image handling Module, Full Stack Module.

## 4.2 Context Diagram

The context diagram is used to establish the context and boundaries of the system to be modelled: which things are inside and outside of the system being modelled, and what is the relationship of the system with these external entities.

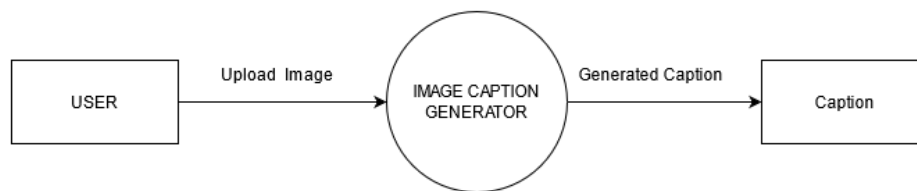


Fig 4.2 Context Diagram

- User will upload image to the system
- System will generate the caption for uploaded image
- Generated caption is shown to the user

## Chapter 5: Detailed Design

### 5.1 System Design

**5.1.1 Dynamic Model:** Dynamic Model involves states, events and state diagram (transition diagram) on the model. Main concepts related with Dynamic Model are states, transition between states and events to trigger the transitions. Predefined relationships in object model are aggregation (concurrency) and generalization.

**5.1.1.1 Activity Diagram:** Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

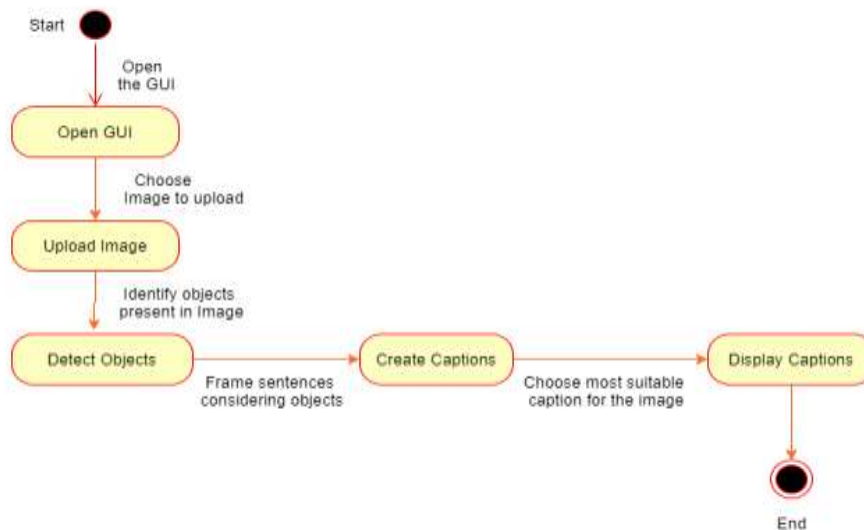


Fig 5.1.1.1 Activity Diagram

Fig 5.1.1.1 represents Activity diagram where various activities of the user is mentioned, the activities are

- User opens GUI
- Uploads image
- System will detect objects
- Based on objects generate caption
- Display caption to user

**5.1.1.2 Sequence Diagram:** A sequence diagram shows object interactions arranged in time sequence

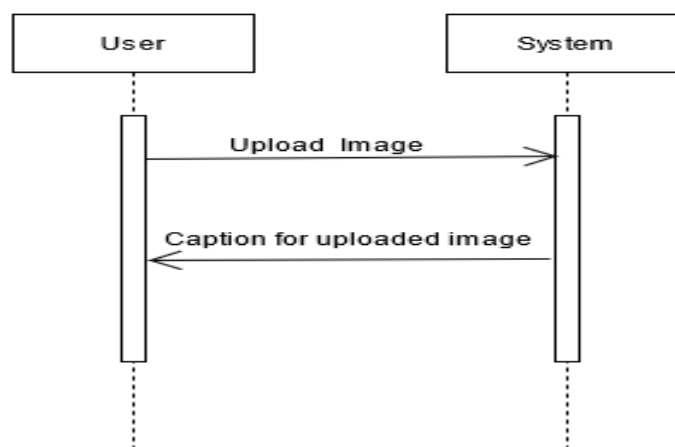


Fig 5.1.1.2 Sequence Diagram

Fig 5.1.1.2 represents sequence diagram where user will upload image to the system and get appropriate caption as the result.

**5.1.2 Functional Model:** Functional Model focuses on the how data is flowing, where data is stored and different processes. Main concepts involved in Functional Model are data, data flow, data store, process and actors. Functional Model in OMT describes the whole processes and actions with the help of data flow diagram (DFD).

**Data Flow Diagrams:** Data flow diagram describes flow of data through a system to perform certain functionality of a business.



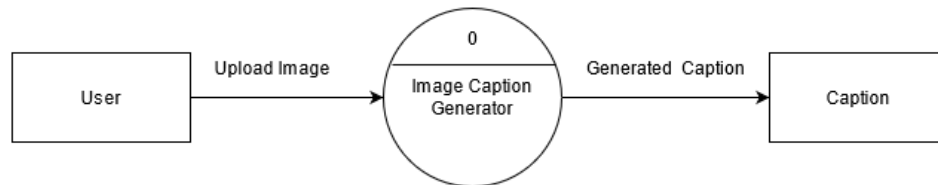
**Level 0:**

Fig 5.1.2.1 Data Flow Diagram Level 0

Fig 5.1.2.1 represents Level 0 dataflow diagram which is very basic level, where user uploads image to the Image Caption Generator system and get appropriate caption as Output.

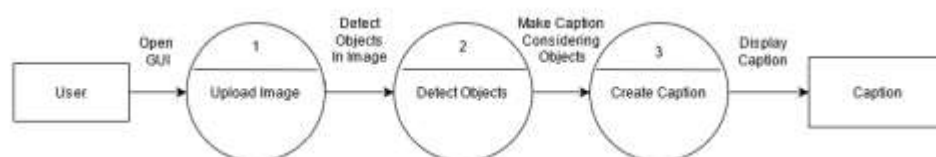
**Level 1:**

Fig 5.1.2.2 Data Flow Diagram Level 1

Fig 5.1.2.2 represents Level 1 dataflow diagram, which represents further details about different processes of the system.

- User will open GUI
- Upload Image process let the user to upload image
- Detect Objects process will detect objects from the uploaded image.
- Create Caption process will create the caption based on the detected objects.
- Finally the created caption will be displayed back to user.

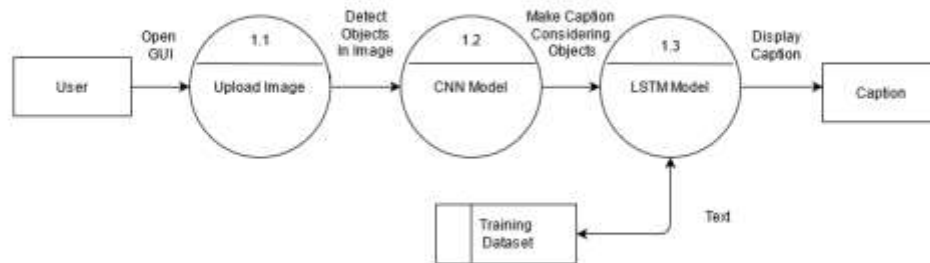
**Level 2:**

Fig 5.1.2.3 Data Flow Diagram Level 2

Fig 5.1.2.3 represents Level 2 dataflow diagram, which represents detailed description of processes involved,

- User opens GUI
- Upload Image process let the user to upload image
- CNN Model helps to detect objects in the image
- LSTM Model helps in generating caption based on detected objects.
- Finally Caption will be displayed to user.

## Chapter 6: Implementation

### 6.1 Code Snippets

#### Index.html

```
<html>
<head>
<link rel="stylesheet" href="static/css/index.css"/>
<link rel="preconnect" href="https://fonts.gstatic.com">
  <link
href="https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;0,700;0,900;1,100;1,300;1,400;1,700;1,900&display=swap" rel="stylesheet">
  <link    href="https://fonts.googleapis.com/css2?family=Hachi+Maru+Pop&display=swap"
rel="stylesheet">

<script type='text/javascript'>
function preview_image(event)
{
  var reader = new FileReader();
  reader.onload = function()
  {
    var output = document.getElementById('output_image');
    output.src = reader.result;
  }
  reader.readAsDataURL(event.target.files[0]);
}
</script>
{% extends "base.html" %}
{% block body %}
<body class="body">
<div class="header-container">
```

```



<h1 style="margin-top:20px;">Caption Generator</h1>
</div>
<form action="/" method="post" enctype="multipart/form-data">
<div id="wrapper"><br>
  <input type="file" name="file1" accept="image/*" onchange="preview_image(event)"
    required><br><br><br>
<div id="output_image"></div>
  <br>
  <input type="submit" class="button-text" value="GENERATE CAPTION">
</div>
</form>
<div class="output">
  
</div>
<div class="caption-text"><h2 style="margin-left:30px;">Generated Caption:
  <i>{{ desc }}</i></h2></div>

{% endblock %}
</body>
</html>

```

**Imagecaptiongenerator.ipynb**

```
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_notebook 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 punctuations 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 lowercase
            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()

# Set these path according to project folder in you system
dataset_text = "Flickr8k_text"
```

```
dataset_images = "Flickr8k_Dataset/Flickr8k_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

#2048 feature vector
features = extract_features(dataset_images)
dump(features, open("features.p", "wb"))

###

features=load(open("features.p", "rb"))

###

#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 = { }
    for line in file.split("\n"):

        words = line.split()
        if len(words)<1 :
            continue
```

```
image, image_caption = words[0], words[1:]

if image in photos:
    if image not in descriptions:
        descriptions[image] = []
        desc = '<start> ' + " ".join(image_caption) + ' <end>'
        descriptions[image].append(desc)

return descriptions


def load_features(photos):
    #loading all features
    all_features = load(open("features.p", "rb"))
    #selecting only needed features
    features = {k:all_features[k] for k in photos}
    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)

###

#converting dictionary to clean list of descriptions
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

# give each word an index, and store that into tokenizer.p pickle file
tokenizer = create_tokenizer(train_descriptions)
dump(tokenizer, open('tokenizer.p', 'wb'))
vocab_size = len(tokenizer.word_index) + 1
vocab_size

###

#calculate maximum length of descriptions
def max_length(descriptions):
    desc_list = dict_to_list(descriptions)
    return max(len(d.split()) for d in desc_list)

max_length = max_length(descriptions)
max_length

###
```

```
#create input-output sequence pairs from the image description.

#data generator, used by model.fit_generator()
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)
```

```
#You can check the shape of the input and output for your model
[a,b],c = next(data_generator(train_descriptions, features, tokenizer, max_length))
a.shape, b.shape, c.shape
#((47, 2048), (47, 32), (47, 7577))

###

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='softmax')(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)
model = define_model(vocab_size, max_length)
epochs = 10
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=1, steps_per_epoch= steps, verbose=1)
    model.save("models/model_" + str(i) + ".h5")
```

**app.py**

```
from flask import Flask, render_template, request
from flask.helpers import flash
from flask_sqlalchemy import SQLAlchemy
#from keras_applications.imagenet_utils import _obtain_input_shape
import numpy as np
from PIL import Image
#import matplotlib.pyplot as plt
#import argparse
from PIL.ImagePalette import load
from keras.models import Model, load_model
from pickle import dump, load
#import string
#import os
#from tqdm import tqdm
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.layers import Input, Dense, LSTM, Embedding, Dropout

def extract_features(filename, model):
    try:
        image = Image.open(filename)

    except:
        print(
            "ERROR: Couldn't open image! Make sure the image path and extension is correct")
```

```
image = image.resize((299, 299))
image = np.array(image)
# for images that has 4 channels, we convert them into 3 channels
if image.shape[2] == 4:
    image = image[..., :3]
image = np.expand_dims(image, axis=0)
image = image/127.5
image = image - 1.0
feature = model.predict(image)
return feature
```

```
def word_for_id(integer, tokenizer):
    for word, index in tokenizer.word_index.items():
        if index == integer:
            return word
    return None
```

```
def generate_desc(model, tokenizer, photo, max_length):
    in_text = ""
    for _ in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence], maxlen=max_length)
        pred = model.predict([photo, sequence], verbose=0)
        pred = np.argmax(pred)
        word = word_for_id(pred, tokenizer)
        if word is None:
            break
        in_text += ' ' + word
        if word == 'end':
            break
```



```
    return in_text

max_length = 32


app = Flask(__name__)
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 1
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql+pymysql://root:Skg@123!@localhost/flask_learning'
db = SQLAlchemy(app)


class Contact(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    fname = db.Column(db.String(50), unique=False, nullable=False)
    lname = db.Column(db.String(50), unique=False, nullable=False)
    contact = db.Column(db.String(12), unique=True, nullable=False)
    email = db.Column(db.String(255), unique=True, nullable=False)
    msg = db.Column(db.String(100), unique=False, nullable=False)


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/aboutus')
def aboutus():
    return render_template('aboutus.html')
```

```
@app.route('/contactus', methods=['GET', 'POST'])
def contactus():

    if(request.method == 'POST'):
        f_name = request.form.get('firstname')
        l_name = request.form.get('lastname')
        contactus = request.form.get('telnum')
        emailid = request.form.get('emailid')
        feedback = request.form.get('feedback')

        entry = Contact(fname=f_name, lname=l_name,
                        contact=contactus, email=emailid, msg=feedback)
        db.session.add(entry)
        db.session.commit()
        flash("Feedback Received..We will get back to you very soon!!!", "success")

    return render_template('contactus.html')


@app.route('/', methods=['GET', 'POST'])
def prediction():
    if request.method == 'POST':
        file = request.files['file1']
        img_path = "./static/files.jpg"
        file.save(img_path)
        tokenizer = load(open("tokenizer.p", "rb"))
        model = load_model('models/model_9.h5')
        xception_model = Xception(include_top=False, pooling="avg")
        photo = extract_features(img_path, xception_model)
        img = Image.open(img_path)
```

```

description = generate_desc(model, tokenizer, photo, max_length)

"""result_dic = {
    'img_path': img_path,
    'caption': description
}"""

return render_template('index.html', desc=description)

app.run(debug=True)

```

## 6.2 Implementation

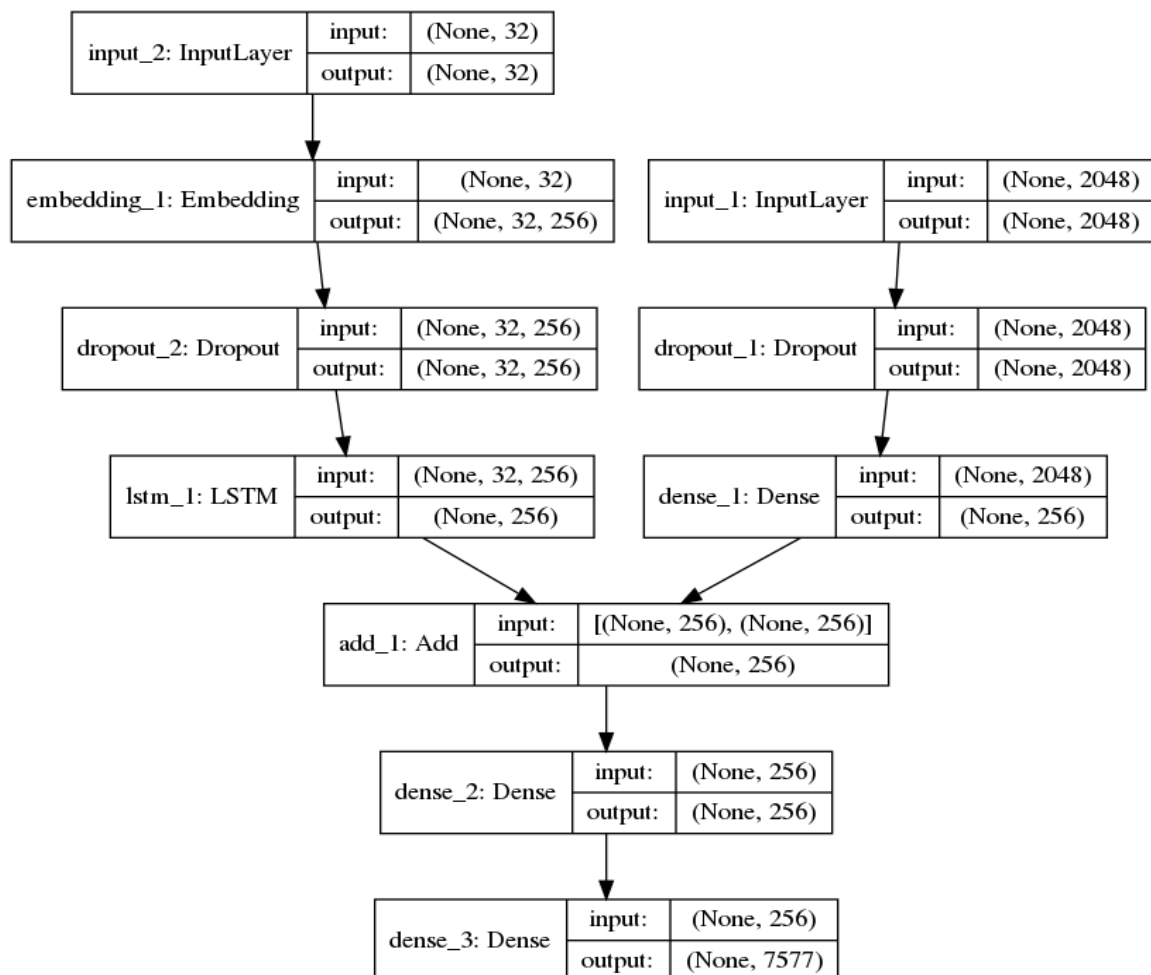


Fig 6.2.1 Result of CNN-RNN model

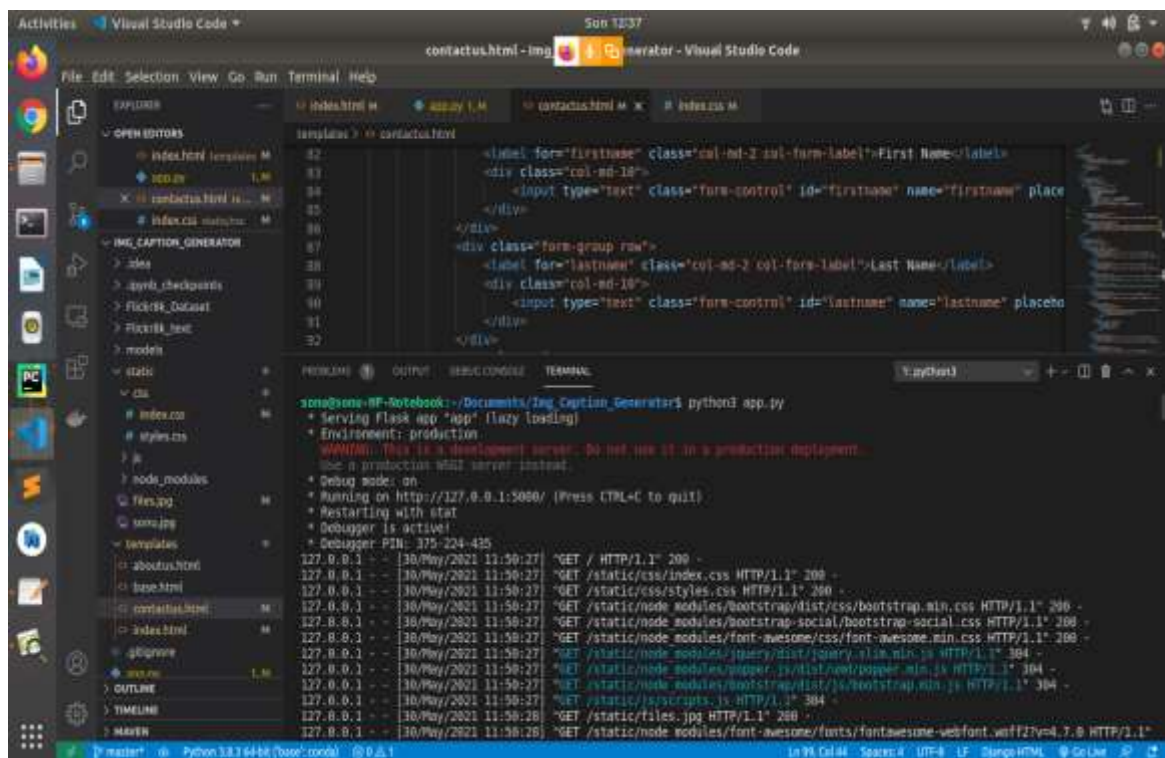


Fig 6.2.2 Starting server

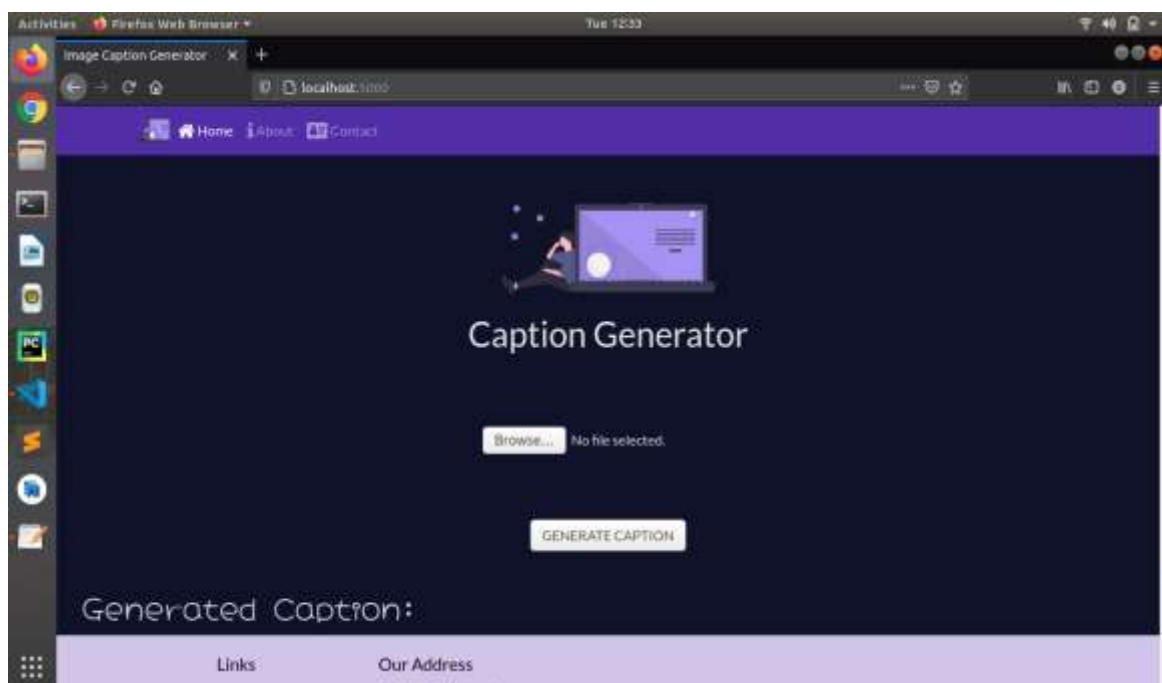


Fig 6.2.3 User Interface

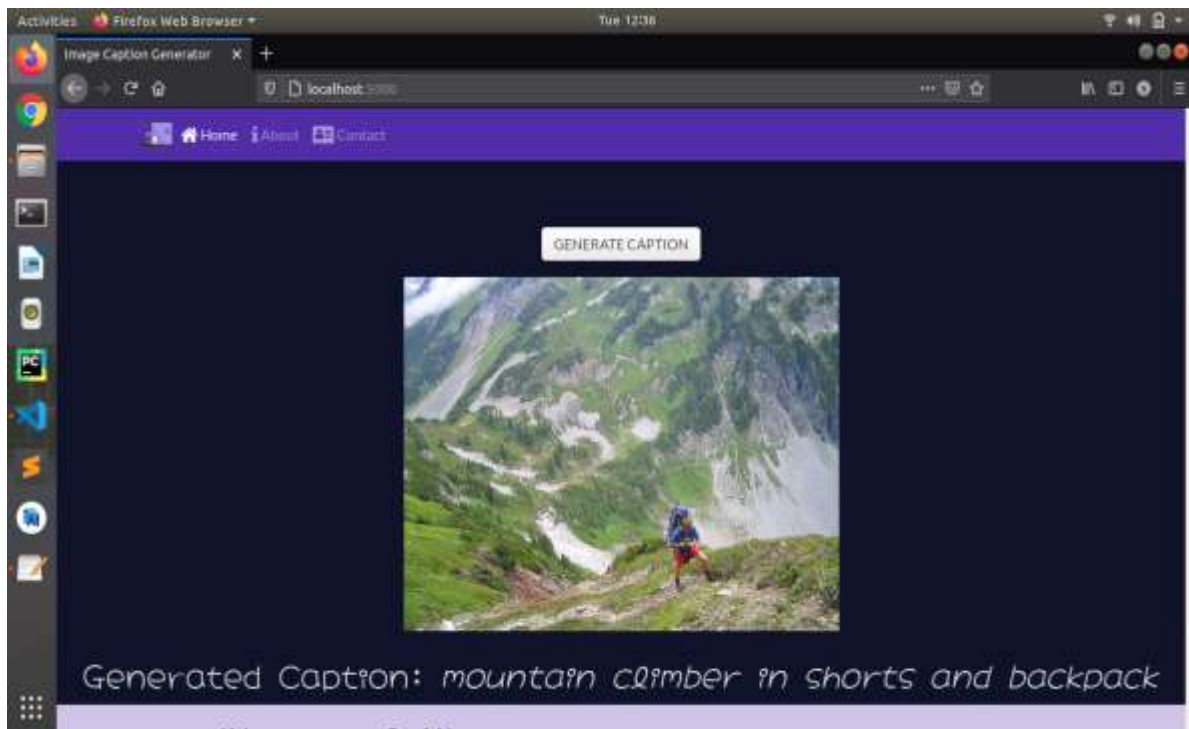


Fig 6.2.4 Caption of uploaded image

## Chapter 7: Software Testing

### White Box Testing:

Here we are testing internal operations of system, it involves testing of software code for flow of specific inputs through the code.

**Unit Testing:** Here individual units or components of software are tested the purpose is to validate that each unit of software code performs as expected .It is done in development phase of an application by developers.

### Integration Testing:

Here individual units or components of the application's source codes are combined and tested as a group. The purpose is to expose errors in the interactions of different interfaces with one another. It is done after unit testing.

## 7.1 Test cases

### 7.1.1 Unit Testing:

**Table 7.1.1 Test cases and Results for Unit Testing**

Test Case Id	Description	Input	Expected Output	Actual Output	Remark
1	Able to import required libraries	Import statements	Able to import	Able to import	Pass
2	Check whether correct path is given for text and image files	Path	Correct path	Correct path	Pass
3	Check whether able to load and read text file	Load document function with file name	Able to load and read text file	Able to load and read text file	Pass
4	Check whether able to get all images	Get all images function with	Able to get images	Able to get images	Pass

	with their captions	file name			
5	Check Data cleaning	Function for removing punctuations, converting all text to lowercase and removing words that contain numbers.	Punctuations are removed, converted text to lowercase and removed words that contain numbers.	Punctuations are removed, converted text to lowercase and removed words that contain numbers.	Pass
6.	Check for Vocabulary building	Function that will separate all unique words and create vocabulary.	Able to build vocabulary	Able to build vocabulary	Pass
7.	Check for descriptions.txt file	Function that will create a list of all the descriptions that have been preprocessed and store them into a file	Descriptions.txt file will be created to store all the captions	Descriptions.txt file will be created	Pass
8	Check for Extraction of feature vector from all images	function extract_features() will extract features for all images and we will map image names with their respective feature array. Then we will dump the features dictionary into a "features.p"	features.p file	features.p file	Pass

		pickle file.			
9.	Check for loading the data for training model	Load Flickr_8k.trainImages.txt, descriptions.txt and features.p	Data loaded	Data loaded	Pass
10.	Check for Tokenizing the vocabulary	Keras library provides us with the tokenizer function that we will use to create tokens from our vocabulary and save them to a "tokenizer.p" pickle file.	tokenizer.txt	Tokenizer.txt	Pass
11	Check data generator	create all of the input-output sequence pairs from the image's description	input-output sequence pairs	input-output sequence pairs	Pass
12	Check for CNN-RNN model	feature extractor model, LSTM sequence model, Merging both models	Model	Model	Pass
13	Check Training of model	define model function	Models contain trained model	Models contain trained model	Pass
14	Testing the model	Testing_caption_generator.py which will load the model and generate predictions	Caption	Caption	Pass



### 7.1.2 Integration Testing

**Table 7.1.2 Test cases and Results for Integration Testing**

<b>Test Case Id</b>	<b>Description</b>	<b>Input</b>	<b>Expected Output</b>	<b>Actual Output</b>	<b>Remark</b>
15	Integrate tested model and GUI and test as whole	Integrate model and GUI	Integrated system	Integrated system	Pass
16	Check whether user able to select Image	Select image from the local system	Image selected successfully	Image selected successfully	Pass
17	Check whether user able to upload Image	After selecting image click on Upload Image button	Image uploaded successfully	Image uploaded successfully	Pass
18	Check whether user able to get caption	Click on Generate Caption button	Appropriate caption	Appropriate caption	Pass
19	Check whether user will be able to give feedback	Add information and press submit button	Feedback Successful	Feedback Successful	Pass

## 7.2 Testing and Validations

For validating accuracy the concept of BLEU is used, BLEU stands for Bilingual Evaluation Understudy.

It is an algorithm, which has been used for evaluating the quality of machine translated text. We can use BLEU to check the quality of our generated caption.

BLEU compares the n-gram of the candidate translation with n-gram of the reference translation to count the number of matches. These matches are independent of the positions where they occur.

The more the number of matches between candidate and reference translation, the better is the caption.

### **Precision:**

In terms generated caption, we define Precision as ‘the count of the number of candidate translation words which occur in any reference translation’ divided by the ‘total number of words in the candidate translation.’

$$\text{Precision} = \frac{\text{No. of candidate translation words occurring in any reference translation}}{\text{Total no. of words in the candidate translation}}$$

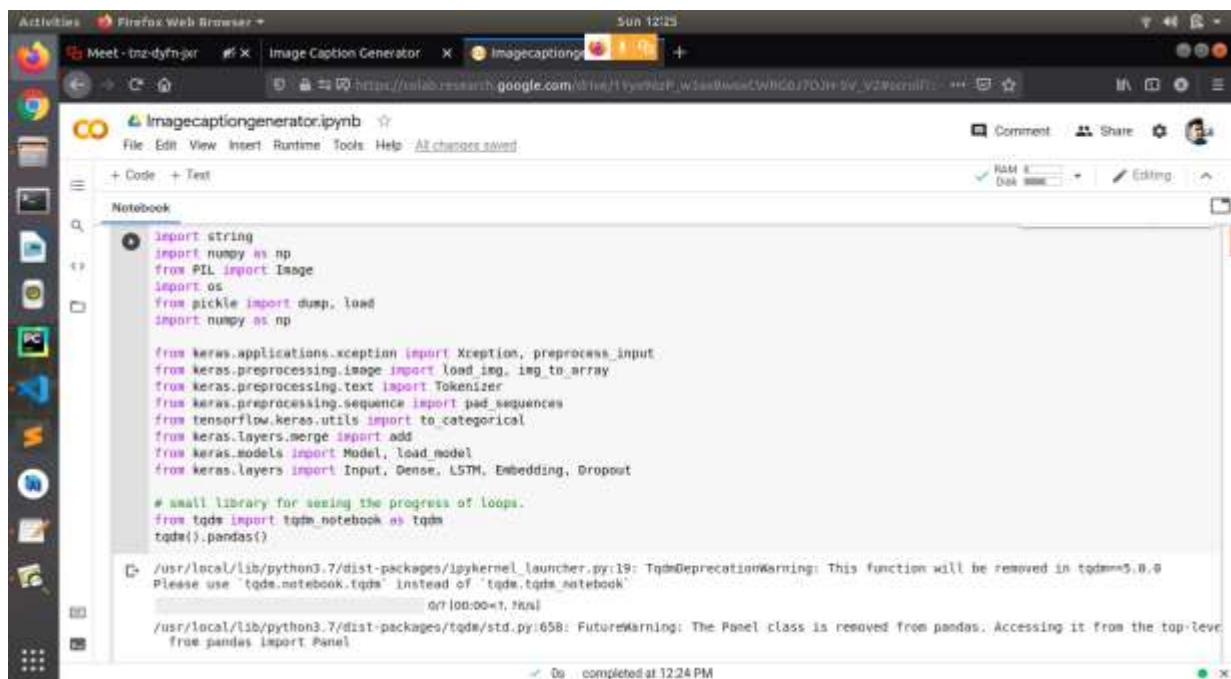
Example:

Candidate 1: the the the the the the the

Candidate 2: the cat is on the mat

Reference: The cat is on the mat

- The precision for candidate 1 is 2/7 (28.5%)
- The Precision for candidate 2 is 1(100%).



```

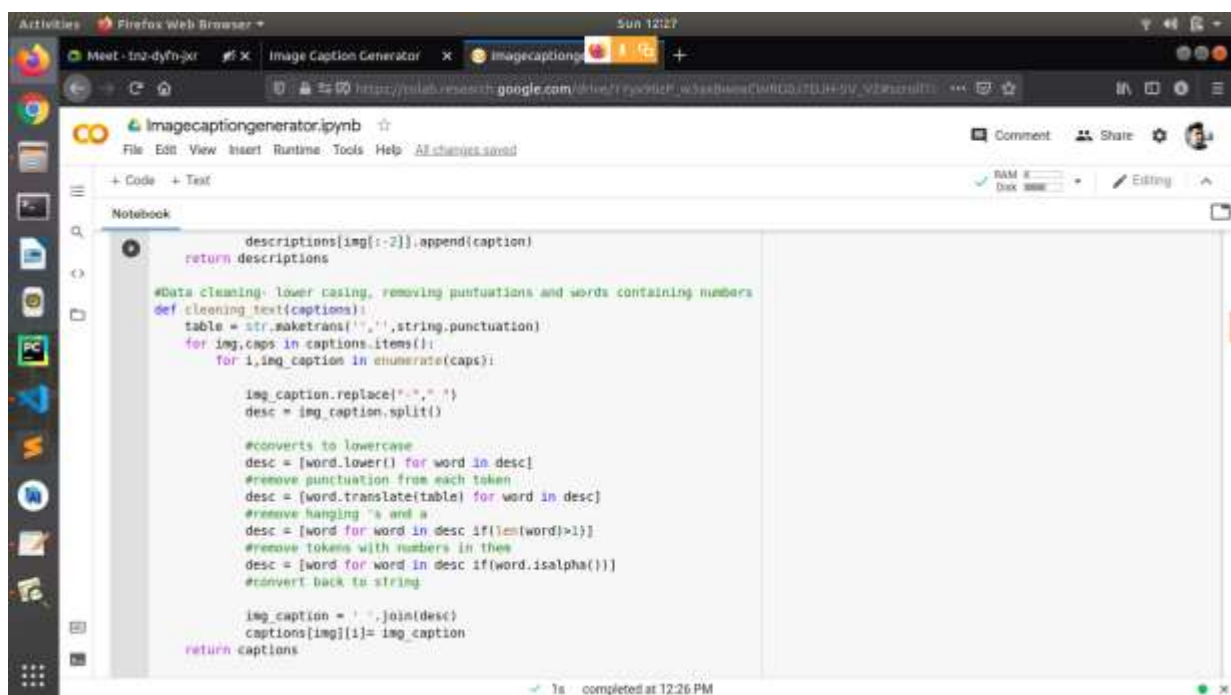
import string
import numpy as np
from PIL import Image
import os
from pickle import dump, load
import numpy as np

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 tensorflow.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_notebook as tqdm
tqdm().pandas()

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:19: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use 'tqdm.notebook.tqdm' instead of 'tqdm.tqdm_notebook'
0/7 [00:00<, 100s]
/usr/local/lib/python3.7/dist-packages/tqdm/std.py:658: FutureWarning: The Panel class is removed from pandas. Accessing it from the top-level
true pandas import Panel
  
```

Fig 7.2.1 Importing required libraries



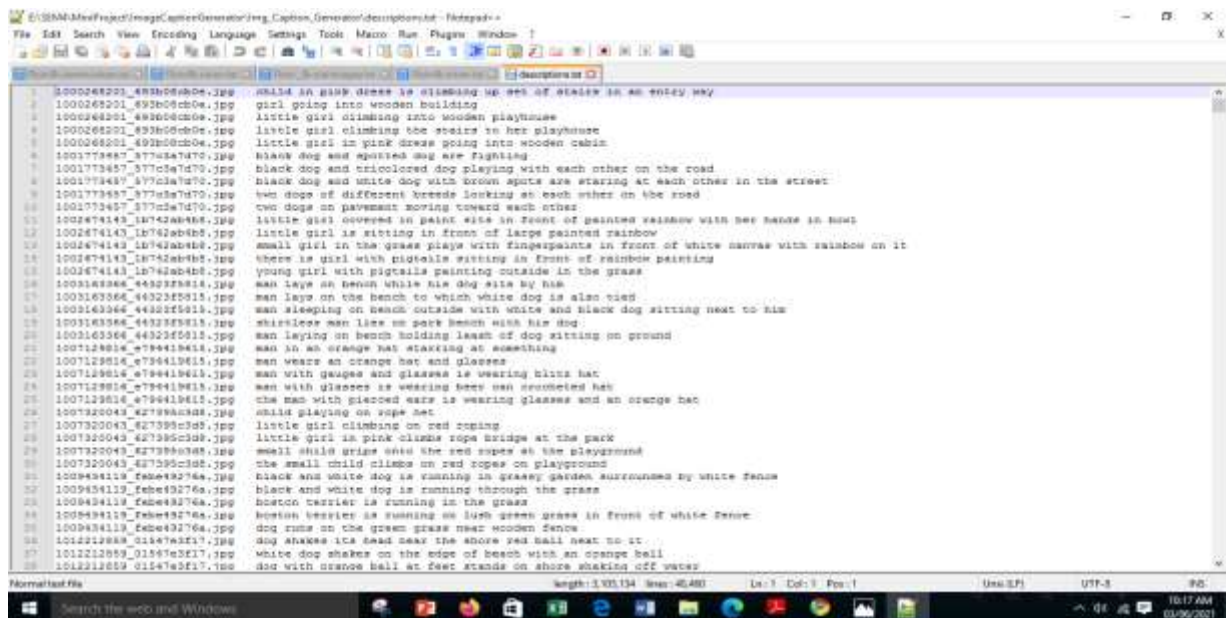
```

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

#Data cleaning: lower casing, removing punctuations 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 lowercase
            desc = [word.lower() for word in desc]
            #remove punctuation from each token
            desc = [word.translate(table) for word in desc]
            #remove hanging 'a 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
  
```

Fig 7.2.2 Data Cleaning



### 7.2.3 Output of cleaning descriptions.txt

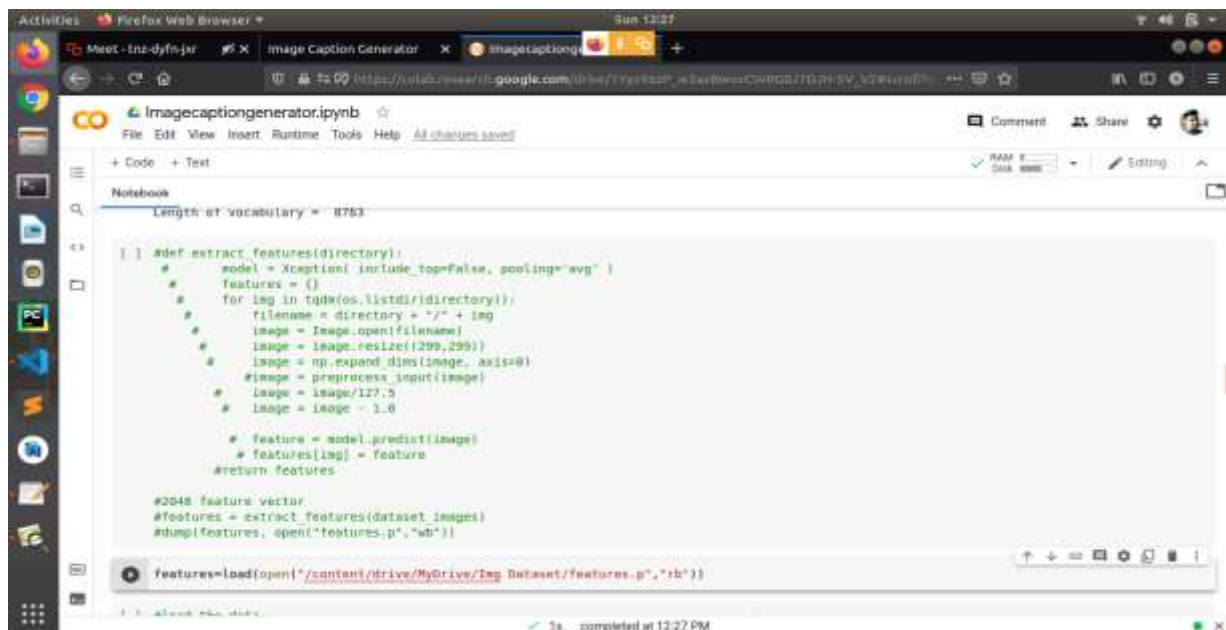


Fig 7.2.4 Feature Extraction: features stored in features.p

```

#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 = {}
    for line in file.split("\n"):
        words = line.split()
        if len(words) < 2:
            continue
        image, image_caption = words[0], words[1:]

        if image in photos:
            if image not in descriptions:
                descriptions[image] = {}
            desc = '<start>' + ' '.join(image_caption) + '<end>'
            descriptions[image].append(desc)

```

Fig 7.2.5.Loading dataset for training model

```

#converting dictionary to clean list of descriptions
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

# give each word an index, and store that into tokenizer.p pickle file
tokenizer = create_tokenizer(train_descriptions)
dump(tokenizer, open('/content/drive/MyDrive/Img_Dataset/tokenizer.p', 'wb'))
vocab_size = len(tokenizer.word_index) + 1
vocab_size

```

Fig 7.2.6.Tokenizing vocabulary



```

# Create input-output sequence pairs from the image descriptions.
#data generator, used by model.fit_generator()
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)

```

Fig 7.2.7 Data generator

```

from tensorflow.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)

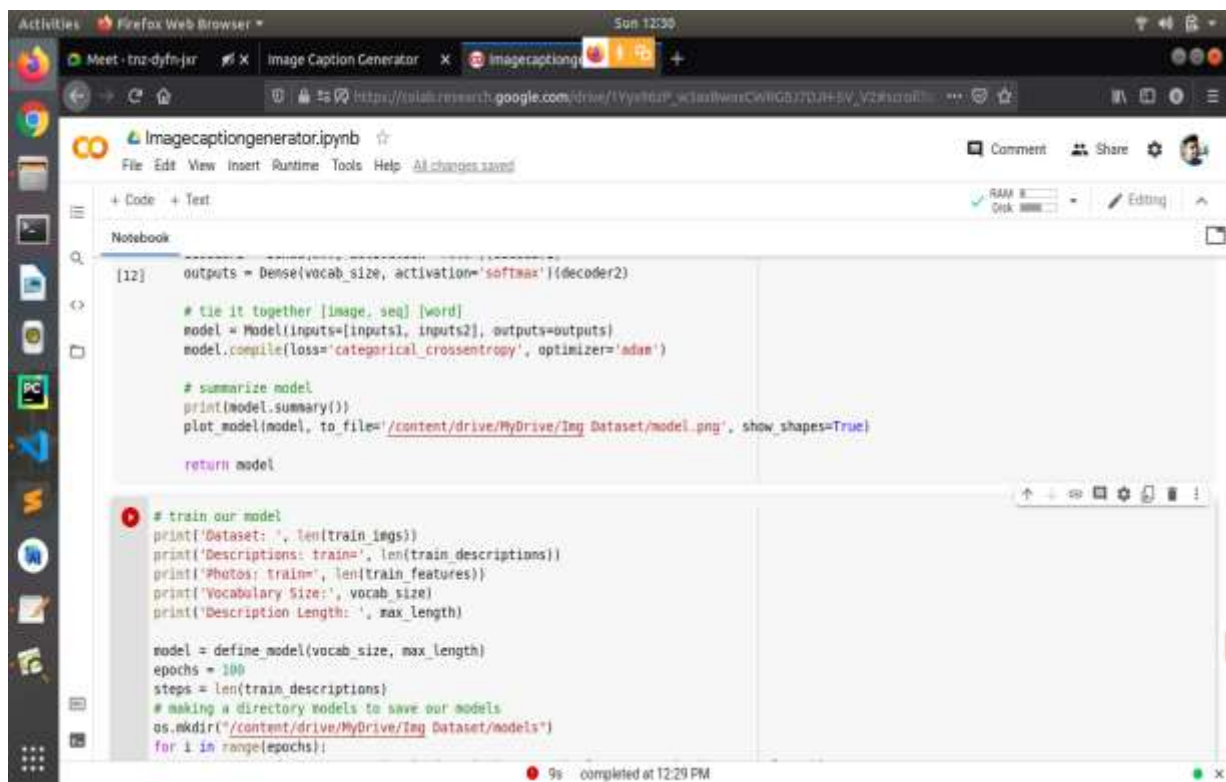
    # LSTM sequence model
    inputs2 = Input(shape=(max_length,))
    se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
    se2 = Dropout(0.5)(se1)
    se3 = LSTM(256)(se2)

    # Merging both models
    decoder1 = add([fe2, se3])
    decoder2 = Dense(256, activation='relu')(decoder1)
    outputs = Dense(vocab_size, activation='softmax')(decoder2)

    # tie it together [image, seq] [word]
    model = Model(inputs=[inputs1, inputs2], outputs=outputs)
    model.compile(loss='categorical_crossentropy', optimizer='adam')

```

Fig 7.2.8 CNN-RNN Model



```
[12] outputs = Dense(vocab_size, activation='softmax')(decoder2)

# tie it together [image, seq] [word]
model = Model([inputs1, inputs2], outputs=outputs)
model.compile(loss='categorical_crossentropy', optimizer='adam')

# summarize model
print(model.summary())
plot_model(model, to_file='/content/drive/MyDrive/Img_Dataset/model.png', show_shapes=True)

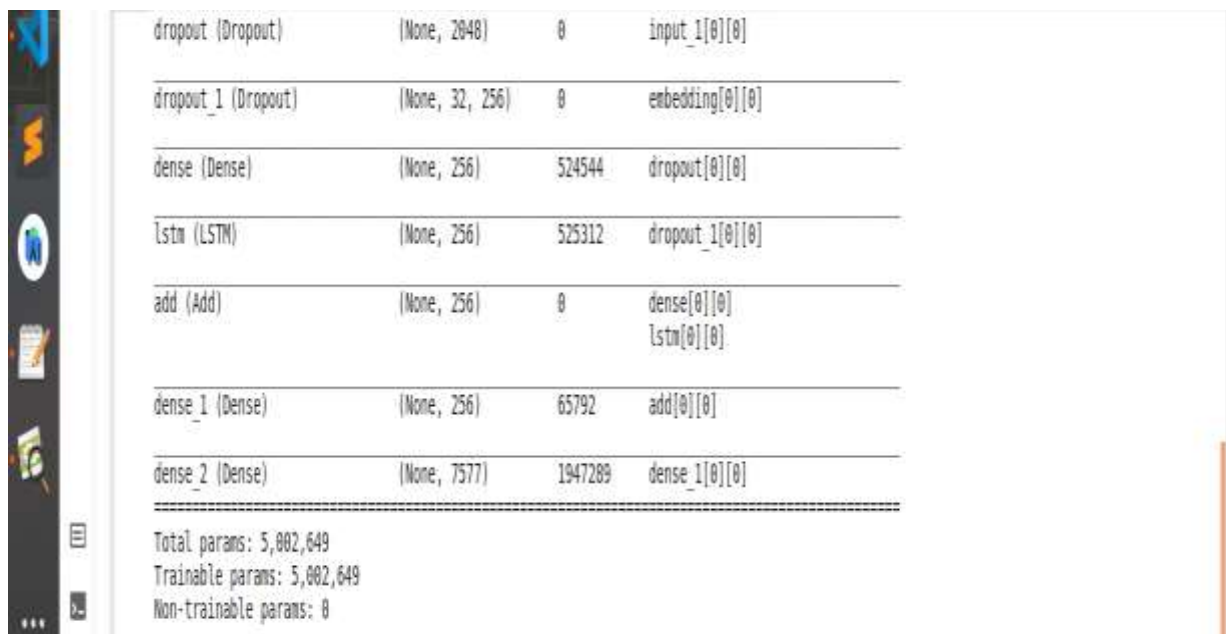
return model

# 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 = 100
steps = len(train_descriptions)
# making a directory models to save our models
os.mkdir('/content/drive/MyDrive/Img_Dataset/models')
for i in range(epochs):
```

9s completed at 12:29 PM

Fig 7.2.9 Training model



dropout (Dropout)	(None, 2048)	0	input_1[0][0]
dropout_1 (Dropout)	(None, 32, 256)	0	embedding[0][0]
dense (Dense)	(None, 256)	524544	dropout[0][0]
lstm (LSTM)	(None, 256)	525312	dropout_1[0][0]
add (Add)	(None, 256)	0	dense[0][0] lstm[0][0]
dense_1 (Dense)	(None, 256)	65792	add[0][0]
dense_2 (Dense)	(None, 7577)	1947289	dense_1[0][0]
Total params: 5,002,649			
Trainable params: 5,002,649			
Non-trainable params: 0			

Fig 7.2.9.1 Training model output

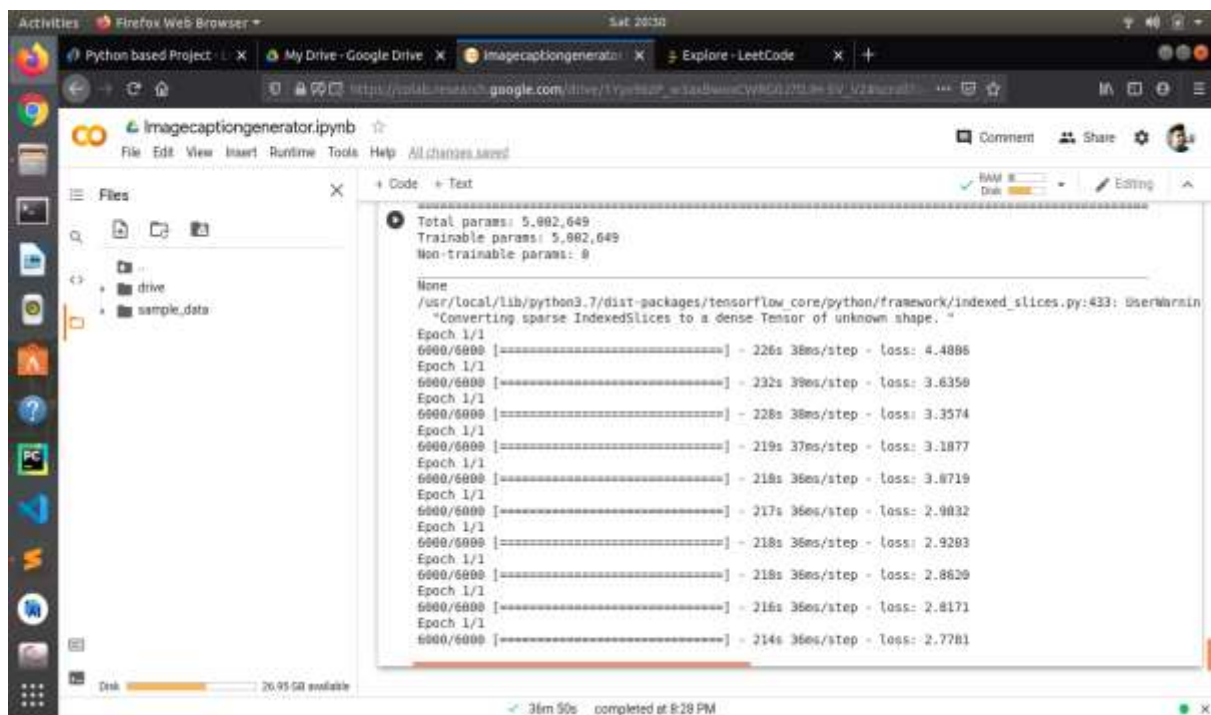


Fig 7.2.9.2 Training output

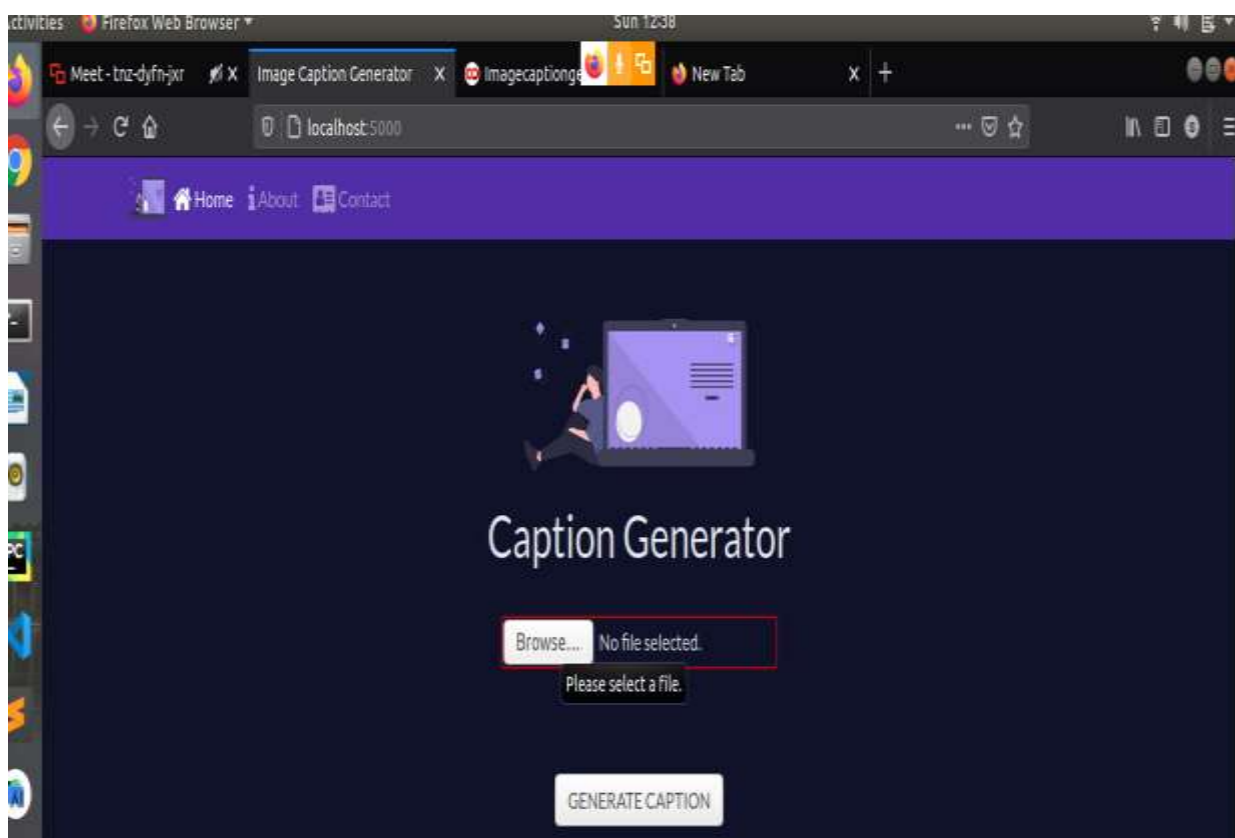
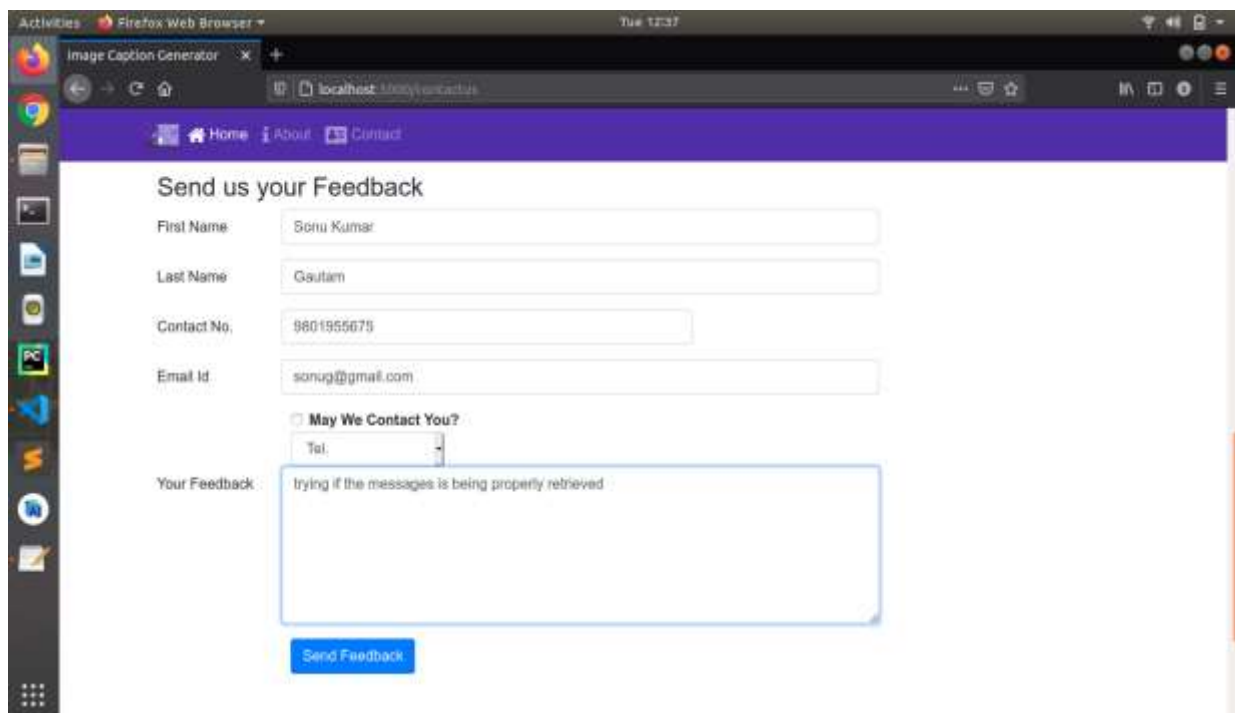


Fig 7.2.10 Clicking Generate Caption button without uploading image





The screenshot shows a web browser window with the title "Image Caption Generator". The address bar shows "localhost:5000/contactus". The page has a purple header with navigation links: Home, About, and Contact. The main content area is titled "Send us your Feedback". It contains a form with the following fields: First Name (filled with "Sonu Kumar"), Last Name (filled with "Gautam"), Contact No. (filled with "9801955679"), and Email Id (filled with "sonug@gmail.com"). Below these fields is a checkbox labeled "May We Contact You?" which is checked, and a dropdown menu labeled "Tel." with "Tel." selected. A text area labeled "Your Feedback:" contains the text "trying if the messages is being properly retrieved". At the bottom of the form is a blue button labeled "Send Feedback".

Fig 7.2.11 User Feedback Form

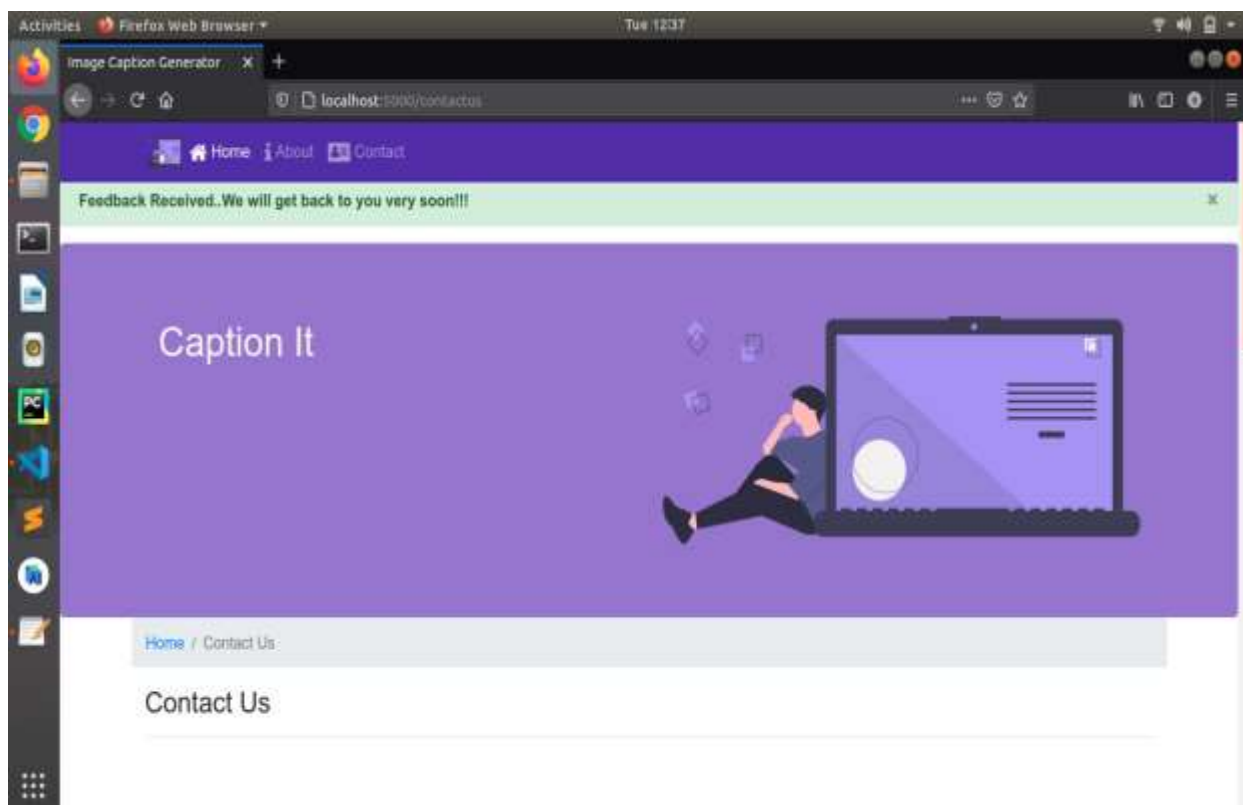


Fig 7.2.12 Response on Successful feedback

## Chapter 8: Conclusion

Today lot of images are being clicked because of the digital era. People are clicking pictures and uploading on various social media platforms but they lack when they have to provide relevant caption to their images. It is very difficult for outwardly disabled individuals better comprehend the substance of pictures on the web. Image Caption Generator could help individuals with outwardly hindrance better comprehend visual information sources, along these lines going about as a right hand or a guide, this will help them by providing captions. It will also help people to generate relevant caption for their images that they would be uploading on social media. It could be used in domains such as Hospitals, Educational Institutions to generate caption from the images

The idea behind choosing the project was to learn the technologies which is being used in day to day life and is in very demand in the IT industries these days.

This project helps in understanding the deep learning aspects through which are able to demonstrate how computers learn to process visual data. Through this project have implemented deep learning concepts such as CNN-RNN model by building Image Caption Generator.

This project demonstrates that how process of generating caption for images works automatically at the fundamental level. Here basically user will upload image for which user wants to generate caption. The uploaded image is given to trained CNN-RNN model. Then the model will generate relevant caption and display it to the user. The main focus here is the foundation of computer vision which further provides the support of solving more real world problems.

## Chapter 9: Future Enhancement

- The model can be trained with much larger dataset so that it provides more accurate results.
- Image is often rich in content .the model should be able to generate description corresponding to multiple main objects.
- This model can also be integrated with other domains such as hospital education and mainly in the field of social media. For an example when the user uploads an image on social media platform user is asked to provide captions, rather than user giving a caption this model is integrated with platform that suggests relevant caption according to image to the user.

## Bibliography

- [1] Marc Tanti, Albert Gatt, Kenneth P. Camilleri, "What is the Role of Recurrent Natural Networks (RNNs) in an Image Caption Generator?", Endeavour Scholarship Scheme (Melta), part-financed by the European Social Fund (ESF), 25<sup>th</sup> Aug 2017, arXiv:1708.02043v2 [cs.CL]
- [2] Lukasz Kaiser, Samy Bengio, Google Brain, "Can Active Memory Replace Attention", 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 7<sup>th</sup> Mar 2017, arXiv:1610.08613v2 [cs.LG]
- [3] Vikram Mullachery, Vishal Motwani, "Image Captioning", 13 May 2018, arXiv:1805.09137v1 [cs.CV].
- [4] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, Yoshua Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", 19<sup>th</sup> Apr 2016, arXiv:1502.03044v3 [cs.LG].
- [5] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan, "Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge", IEEE TRANSACTION ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. XX, NO. XX, MONTH 2016, 21<sup>st</sup> Sep 2016, arXiv:1609.06647v1 [cs.CV]
- [6] MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, Hamid Laga, "A Comprehensive Survey of Deep Learning for Image Captioning", ACM Computing Surveys, Vol. 0, No. 0, Article 0. Acceptance Date: October 2018, 14<sup>th</sup> Oct 2018, arXiv:1810.04020v2 [cs.CV]
- [7] Haoran Wang, Yue Zhang, Xiaosheng Yu, "An Overview of Image Caption Generator Methods", Computational Intelligence and Neuroscience Volume 2020, Article ID 3062706, 13 pages, Received 5 October 2018; Revised 10 December 2019; Accepted 11 December 2019; Published 8 January 2020
- [8] N. Komal Kumar, D. Vigneswari, A. Mohan, K. Laxman, J. Yuvaraj, "Detection and Recognition of Objects in Image Caption Generator System: A Deep learning Approach", 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)
- [9] Pranay Mathur, Aman Gill, Aayush Yadav, Anurag Mishra and Nand Kumar Bansode, "Camera2Camera: A Real Time Image Caption Generator", 2017 International Conference

on Computational Intelligence in Data Science (ICCIDS), Conference Paper · June 2017 DOI: 10.1109/ICCIDS.2017.8272660

[10] Longteng Guo, Jing Liu, Peng Yao, Jiangwei Li, Hanging Lu, "MSCap: Multi-Style Image Captioning with Unpaired stylized Text", Open Access version, provided by the Computer Vision Foundation, IEEE Xplore.

[11] Lakshminarasimhan Srinivasan, Dinesh Sreekanthan, Amutha A.L., "Image Captioning-A Deep Learning Approach", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 9 (2018) pp. 7239-7242, © Research India Publications.

[12] Sarthak Mehta, "An Optimized Image Caption Generator", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 07 Issue: 07 | July 2020

[13] Grishma Sharma, Priyanka Kalena, Nishi Malde, Aromal Nair, Saurabh Parkar, "Visual Image Caption Generator Using Deep Learning", 2nd International Conference on Advances in Science & Technology (ICAST-2019) K. J. Somaiya Institute of Engineering & Information Technology, University of Mumbai, Maharashtra, India.

[14] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan, "Show and Tell : A Neural Image Caption Generator", CVPR2015 paper is the open Access version, provided by the Computer Vision Foundation. The authoritative version of this paper is available in IEEE Xplore.

[15] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, "Sequence to Sequence Learning with Neural Networks", arXiv:1409.3215v3 [cs.CL] 14 Dec 2014

[16] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, Yoshua Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches", arXiv:1409.1259v2 [cs.CL] 7 Oct 2014.

[17] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", Published as a conference paper at ICLR 2015, arXiv:1409.0473v7 [cs.CL] 19 May 2016

[18] Kyunghyun Cho, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", arXiv:1406.1078v3 [cs.CL] 3 Sep 2014

- [19] Jia-Yu Pan, Hyung-Jeong Yang, Pinar Duygulu and Christos Faloutsos, "Automatic Image Captioning", Conference Paper · July 2004, DOI: 10.1109/ICME.2004.1394652 · Source: IEEE Xplore.
- [20] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Network", : <https://papers.nips.cc/paper/4824-imagenetclassificationwith-deep-convolutional-neural-networks.pdf>