Image Caption Generator

A Project Report Submitted
In Partial Fulfilment of the Requirements
for the Degree of

Bachelor of Technology
in
Information Technology

by

Gaurav Pratap Singh (2000520139002)
Apoorv Jha (2000520139001)
Mradul Dixit (1900520130031)

Under the Guidance of
Prof. Maheshwari Tripathi
Dr. Pawan Kumar Tiwari

Department of Computer Science and Engineering
Institute of Engineering & Technology
Dr. APJ Abdul Kalam Technical University, Uttar Pradesh

June, 2023

Declaration

We affirm that this project report is the result of our own independent work and that, to the best of our knowledge and belief, it does not include any material that has been previously published or written by another person, nor does it contain any material that has been substantially accepted for the award of any other degree from the university or any other institute of higher learning. We have duly acknowledged all sources used in the text.

Additionally, this project report has not been submitted by us to any other institute to fulfill the requirements of any other degree.

Signature of the Students

Gaurav Pratap Singh (2000520139002)

Apoorv Jha (2000520139001)

Mradul Dixit (1900520130031)

Certificate

We hereby certify that the project report titled **"Image Caption Generator"** submitted by **Gaurav Pratap Singh, Apoorv Jha, Mradul Dixit** in partial fulfillment for the degree of Bachelor of Technology in Computer Science Engineering, is an authentic and genuine work conducted by them under our supervision and guidance at the Department of Computer Science Engineering, Institute of Engineering Technology Lucknow.

We further confirm that this work has not been submitted anywhere else for the purpose of obtaining any other degree, to the best of our knowledge.

(Dr. Pawan Kumar Tiwari)
Department of Computer Science and Engineering,
Institute of Engineering & Technology, Lucknow

(Prof. Maheshwari Tripathi)
Department Of Computer Application,
Institute of Engineering & Technology, Lucknow

Acknowledgement

We would like to extend our heartfelt appreciation to our Supervisor, **Prof. Maheshwari Tripathi, and Co-supervisor, Dr. Pawan Kumar Tiwari,** for their unwavering support and valuable suggestions throughout the course of this project.We would also like to extend my heartfelt thanks to **Dr. Tulika Narang** for her unwavering support and guidance throughout the development of our final year project during seven semester. Their guidance has been instrumental in shaping this thesis work, and their constructive feedback has greatly facilitated its development. Their continuous encouragement and kindness have served as a driving force, motivating us to excel in various aspects of our research.

We would also like to express our gratitude to the **Project Evaluation Committee Members** for their consistent efforts in evaluating our project at different stages and for providing valuable insights to enhance its quality.

We are grateful to Project Coordinator **Dr. Pawan Kumar Tiwari** for providing me with the opportunity to conduct my thesis project and all of the resources and support they provided.

We are also indebted to **Prof. Yogendra Narain Singh,** the Head of the Department of Computer Science and Engineering at IET, Lucknow, for his support and encouragement.

Furthermore, we would like to acknowledge and thank our classmates and teammates for their motivation and assistance during critical junctures, which played a crucial role in successfully completing this project.

Signature of the Students

Gaurav Pratap Singh (2000520139002)

Apoorv Jha (2000520139001)

Mradul Dixit (1900520130031)

Abstract

Picture subtitling is a course of consequently depicting a picture with at least one regular language sentences. As of late, picture inscribing has seen fast advancement, from introductory layout based models to the ongoing ones, in view of profound brain organizations. This paper gives an outline of issues and late picture subtitling research, with a specific accentuation on models that utilization the profound encoder-decoder architecture.Recent propels in profound learning techniques on perceptual undertakings, for example, picture order and protest discovery have urged specialists to handle considerably more troublesome issues for which acknowledgment is only a stage towards to more mind boggling thinking about our visual world. Picture inscribing is one of such undertakings. The point of picture inscribing is to consequently depict a picture with at least one normal language sentences. This is an issue that coordinates PC vision and regular language handling, so its principal challenges emerge from the need of deciphering between two particular, yet generally matched, modalities. In the first place, it is important to recognize objects on the scene and decide the connections among them and afterward, express the picture content accurately with appropriately framed sentences. The produced portrayal is still very different from the manner in which individuals depict pictures since individuals depend on sound judgment and experience, call attention to significant subtleties and overlook items and connections that they suggest. In addition, they frequently use creative mind to make portrayals distinctive and fascinating.

Contents

Chapter 1

Introduction

1.1 Background Information

Image caption generator is a process of recognizing the context of an image and annotating it with relevant captions using deep learning, and computer vision. It includes the labeling of an image with English keywords with the help of datasets provided during model training. It is the system of producing a textual description for given images.It has been an extremely important and basic endeavor in the Deep Learning space. Picture subtitling has a major amount of use.

Picture inscribing can be considered as a start to finish Sequence to Sequence issue, as it changes over pictures, which is considered as a grouping of pixels to a succession of words. NVIDIA is the usage of picture captioning applied sciences to create an software to assist human beings who have low or no eyesight.
Digital images are of 4 types:

- Highly contrasting Images, pixel have esteems high or low.
- Grayscale Images, pixel esteem is range limited showing dull or light.
- RGB Images, every pixel power is addressed in Red, Blue, Green tone.

- RGBA pictures, every pixel alonwith RGB part has an extra alpha part.

1.2 Motivation

As per WHO, near about a billion people have some disability and some 280 million have visual impairments. many of them use some devices like screen readers which help them to understand digital text.

There are many motivations behind image captioning systems. Some of the most common motivations include:

- **To help people with visual impairments-** Image captioning systems can help people with visual impairments to understand the content of images. This can be done by generating text descriptions of images that can be read aloud by a screen reader.
- **To improve search engine results-** Image captioning systems can be used to improve the quality of search engine results. This is because image captions can provide additional information about the content of images, which can help users to find the images they are looking for more easily.
- **To create new forms of media-** Image captioning systems can be used to create new forms of media, such as virtual reality experiences and interactive stories. This is because image captions can provide a way for users to interact with images in a more meaningful way.
- **To advance research in artificial intelligence-** Image captioning systems are a challenging problem in artificial intelligence. By developing better image captioning systems, we can learn more about how humans perceive and understand images.

1.3 Project Objective

We are going to develop a web application that helps us in identifying the action of an image. We aim at creating a Neural Network Model to analyze the images and create captions using transformer models.

1.4 Report Layout

- Literature Review describes all the previous works done in this field.
- Methodology describes the architecture of the project.
- Implementation Details describes the tools that are used and the process that needs to be followed in each mode.
- Results shows the final outputs that are done in the course of this project.
- End indicates the impediments and future extent of this undertaking.

Chapter 2

Literature Review

The ImageNet project, where they openly supported a large number of labelled photographs and created models over the past ten years to perceive items in the picture, is one of the most outstanding examples. Starting in 2010, the annual ImageNet Large Scale Visual Recognition Challenge (ILSCRC) organises a competition to compete for the highest level of precision on various visual recognition projects. Right present, persons in recognition are outnumbered by precise CNN[2] networks. Anyway, captioning photographs can be a challenging undertaking because it involves identifying objects and finding relationships between them. This was unimaginable not so long ago,due to the enormous increase in computer power[13]. Despite the fact that other scientists are working on a related problem, two groups stood out with their estimates. one from Stanford University and the other from Google. A work titled "Sharing time: A Neural Image Caption Generator" was presented by Google in 2014 [6]. Given the image, their algorithm is ready to increase the likelihood of the objective portrayal sentence. The model was developed using a variety of datasets, including Flickr30K, SBU, and MSCOCO, and it successfully produced subtitles at a human level. When Google first presented a paper in 2014, the system made use of the "Commencement V1" photo characterisation model, which achieved 89.6 percentage accuracy. The "Commencement V3" model, which achieves 93.9 percent precision[3], was used for the most recent delivery in 2016. Prior to Google, the DistBelief software system allowed for the possibility of visual subtitling. Then Google released TensorFlow execution, which makes use of GPU power and reduces preparation time by a factor of 4 when compared to earlier executions. Fei Li and Andrej Karpathy, who are from Stanford University, are the other team that did well in handling the problem. They used images and depictions in their 2015 study "Profound Visual-Semantic Alignments for Generating Image Descriptions" [7] to investigate the multi-modular correspondences between language and visual information. They carried out the task using CNN and RNN. Their execution occurs in groups. It makes use of the GPU-powered Torch library, which supports CNN fine-tuning.

Chapter 3

Methodology

The methodology is a relevant structure for research. We have multiple sections that cover the Architecture, Working, and Tools Used in the project.

System Architecture

The accompany addresses the system architecture and the essential working of Web Application of Image caption system.

The framework is intended to give subtitles of a picture.

3.1 RNN

Recurrent neural networks (RNNs) are a type of neural network in which the results of one phase are used as inputs for the next. Traditional neural networks have inputs and outputs that are independent of one another, but when predicting the next word in a phrase, it is necessary to remember the previous words because they are needed. As a result, RNN was developed, which utilised a Hidden Layer to resolve this problem. The Hidden state of RNN, which retains some information about a sequence, is its primary and most significant characteristic. The state, which recalls the previous input to the network, is also known as memory state.

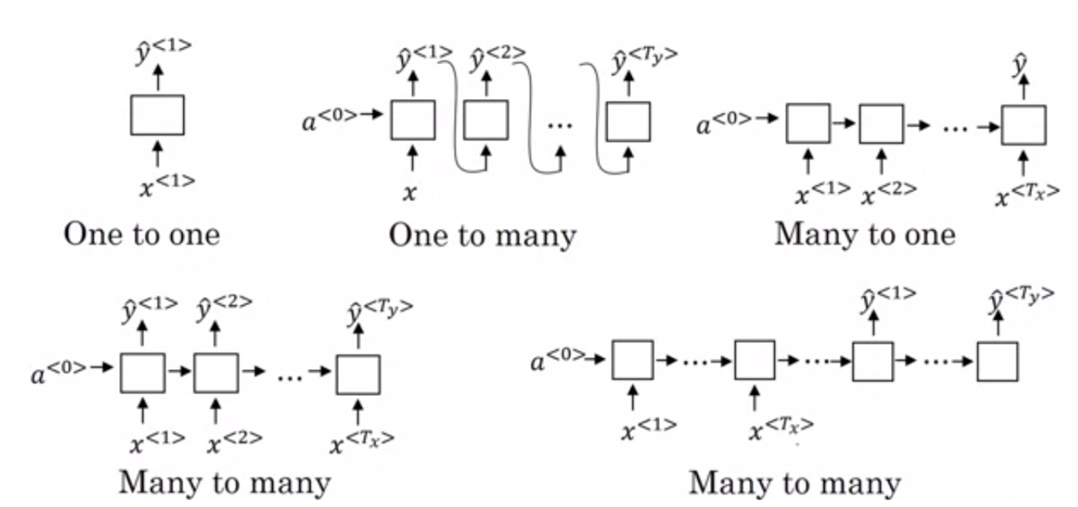


Figure 3.1: Types of RNN(Image from opengenus)

3.2 LSTM

LSTMs are designed to prevent the problem with extended dependency. At this point, learning to recall information for a long period of time is not something kids struggle with. A fictitious repeated brain local area (RNN) structure called long non super durable memory (LSTM) is used in the circle of deep learning. LSTM has comments associations, which is not at all like fashionable feedforward brain organisations. Because there may be gaps of hazy time between crucial events in a measurement, LSTM networks are suitable for organising, managing, and making predictions based on supported measurement measurements.

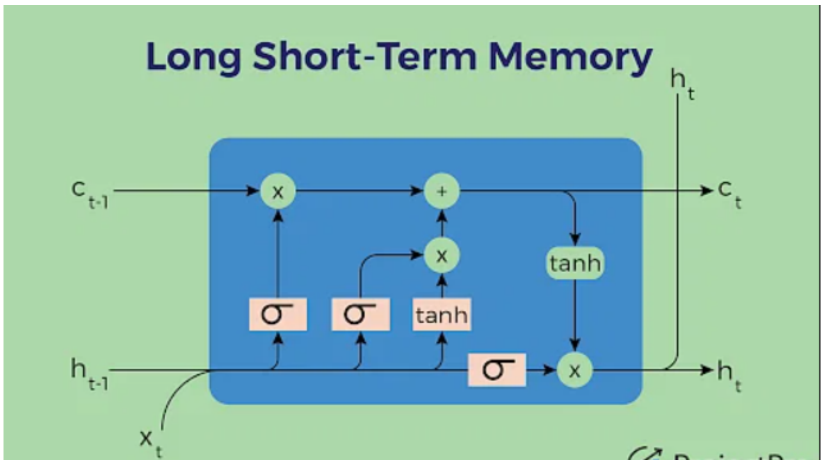


FIGURE 3.2: LSTM Architecture(Image from Stack Overflow)

3.3 ResNet-50

ResNet-50 is a deep convolutional neural network architecture that was introduced in 2015 by researchers from Microsoft Research.ResNet-50 is a deep convolutional neural network architecture that addresses the vanishing gradient problem in deep networks. It consists of 50 layers, including residual connections that allow information to bypass layers. This enables the network to learn residual mappings and train deeper networks. ResNet-50 follows a building block structure with residual blocks containing convolutional layers and shortcut connections. It extracts features from input images, gradually increases complexity, and produces classification probabilities. ResNet-50 is widely used in computer vision tasks and has achieved significant success in image classification, object detection, and image segmentation.

3.4 Transformers

Transformers are shown in the picture below, which is also known as a sequence-to-sequence architecture. A neural network called sequence-to-sequence architecture[4] transforms one grouping of components, such as the words in a sentence, into another. These models are acceptable for interpretation, in which a collection of words from one language are replaced with a collection of various terms from a different dialect.

Above figure proposes the System Architecture of our task that arrangements with the text and pictures.

The walkthrough of the Architecture is as follows:

- The Text Data is cleaned and pre-processed. This Bag Of Words or Embedding Matrix is created to passed through RNN architecture which converts text into its feature vector.
- The image data is pre-processed and size is adjusted to 248*248 to be input into Resnet50 model. The Resnet50 model outputs a feature vector describing the image mathematically.
- Feature vectors created in above steps are concatenated and passed through LSTM model, which train on data and for a particular image learns suitable text.

Chapter 4

Proposed Model

The system created is one for creating captions. It includes pre-processing, caption production, and caption decoding components. Images are read from the disc throughout the training and testing phases. A JPEG/PNG image has been transformed into an array representation. The image is scaled down to a shape array of 224X224X3. The 224x224 array of the image is normalised by the image pre-processor. A Renet50 model processes the image to produce a compressed summary of the information it contains. Text inputs are also preprocessed, but solely for the training phase. Each sentence has'startofseq' and 'endofseq' appended to it, and the text has been turned to lower case. Following this, each word is divided into smaller units, and the frequency of each unit is temporarily kept in the form of a dictionary. A new dictionary is created and used to replace words in a sentence with their corresponding indexes. The statements that correspond to each image are then maintained in a mapping.

LSTM architecture is used to create the text caption for the image using the vectors obtained for the image (in testing) or the image and text (in training).

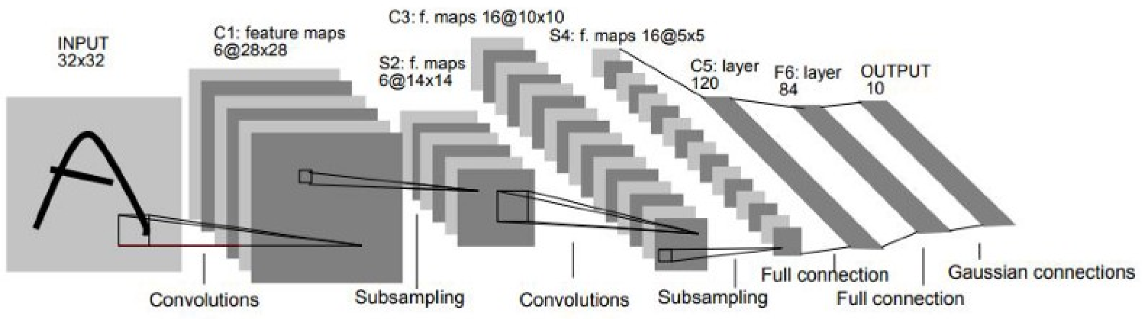


FIGURE 4.1:
An example of CNN architecture (Image from datasceincecentral)

4.1 Architecture

RNN and CNN are the two main types of brain organisations used in the model's engineering. CNN stands for Convolutional Neural Network, and RNN stands for Recurrent Brain Network. They are studied in light of the following observations:

Convolutional Neural Network:

Image recognition is accomplished using neural networks. The issue with basic neural networks is that the number of parameters for a neural group increases when the image has a lot of pixels. This advances neural organisations and uses a lot of computer resources.

CNN[8] are utilised to solve this issue. A unique type of feed-forward neural network is the convolutional neural network. Tensors are matrices of integers with additional dimensions, and convolutional neural networks input and process images as tensors.

There are three primary kinds of layers to fabricate CNN structures:

- Convolutional layer
- Pooling layer
- Fully-connected layer

The entirely associated layer resembles the typical brain organisations quite much. It is possible to think of the convolutional layer as performing the convolution activity frequently found on the previous layer. The limit of each block from the previous layer can be used to downsample the pooling layer. These three layers are stacked together to create the complete CNN engineering.

Convolutional Neural Networks have 2 main components.

- **Feature learning:** We have convolution, ReLU,Pooling layer stages here. In this feature learning process, edges, shades, lines, and curves are extracted.
- **Classification:** In this stage, there is a Fully Connected (FC) layer [9]. They will provide a probability that the object in the image is what the computation indicates it is.

For our use-case, we are only interested in Feature learning component of CNN.

Feature learning:

Has the following components:

The input image can be thought of as a network of pixel values. Consider a 5 x 5 image with only pixel values of 0 and 1. Filter: The input image is extended under the guidance of a channel to produce the convolved layer. To achieve extraordinary focuses like edges, bends, and lines, these channels vary in shapes and values. This channel has occasionally been called Kernel, Feature seeker. Convolved layer is created when each value in every pixel of the information image is duplicated with a different value and pixel of channel. This Convlayer is frequently referred to as a Convolutional Feature, Feature map, or Filter map.

ReLU(Rectified Linear Unit):

Following each Convolution activity, a different one called ReLU[13] has been used. The activity of relu is indirect. ReLU replaces all undesirable pixel values in the limit map with zero as part of an issue-aware activity (applied per pixel). Given that we would prefer our ConvNet to look at measurements that are not straight for the majority of this current reality, ReLU is justified in adding non-linearity to our ConvNet.

Pooling layer:

The dimensionality of the trademark map or convlayer is reduced in this section to protect the important data. This spatial pooling is occasionally also referred to as downsampling or subsampling. Max pooling, average pooling, and total pooling are other possible pooling layers[14]. Max pooling is commonly observed to be used most.

Recurrent Neural Network:

Recurrent Neural Network is represented by RNN[1]. It is a type of brain region that communicates memories and provides good approval for subsequent information. Google Voice Search and Apple's Siri are both used to implement RNN. We should talk about a few key RNN norms.

It is hypothesised that the feed-forward brain local area contains an internal memory. RNNs are repetitive in that they play out the same component for each data entry while the output of the current entry depends on the calculation of the previous entry. After the result is delivered, it is replicated and sent lower into the repetitious organisation once more[5]. It takes into account both the current day entry and the conclusion drawn from the prior data when making a decision.

RNN[10] and feed forward brain networks differ in that RNN can use internal memory to manage the organisation of data sources. It makes sense for subsequent information when the outcome of one information is dependent on earlier informational circumstances and outcomes.

RNN can keep track of previous data. It considers what it has learned from previous informational and resulting conditions as it works through the current information. It uses a combination of current information and its previous state to determine its current status in this way. In this manner, a circle is burned through by the data.

There are different types of RNN:

- one-to-one RNN
- one-to-many RNN
- many-to-many RNN
- many-to-one RNN

RNNs are effective but challenging to train. The "vanishing gradient problem" is the main contributor. Although in theory RNNs might use data in arbitrary long sequences, in practise they are limited to going back only a few stages. With this ability in use, the range of contextual data that popular RNNs can access is restricted[3].In order to solve this problem, different types of structures were developed, the LSTM being one of them.

LSTM (Long Short Term Memory):

A RNN structure called LSTM[1,5] accepts note values at any point in time. It is employed to categorise, process, and forecast the timing of events when given temporal lags with uncertain lengths. LSTM has an advantage over choice RNNs, hidden Markov models, and other sequence learning techniques due to its relative insensitivity to hole size.

The telephone state is a common name for the prolonged memory. Each phone has a recursive nature that enables measurements from stretching out to come before to be stored in the LSTM cell[10]. Cell country may be changed by using the ignore door underneath the telephone area and can also be changed through the enter regulation door.

The ignore entrance is a common name for the consider vector. By copying 0 to a position in the system, the disregard entryway's output instructs the mobile world which measurements to ignore. If the ignore door yields a result of 1, the measurements are saved in the current state of the phone. The enter doorway is frequently used to refer to the retail vector. These entrances specify which information must be stored in the flexible country/long-term memory. The highlights of each door's enactment are what matter. The entranceway is a sigmoid trademark with a [0,1] shift. The term "result entryway" is frequently used to refer to the focus point of the consideration vector. The hidden state is the common name for the working memory.

4.2 Hardware Requirements

- 16 GB RAM
- NVIDIA GPU
- M1 core
- 50 GB physical storage

4.3 Software Requirements

- Windows 11
- Python3
- CUDA 9
- Tensorflow
- Keras
- Numpy
- Matplotlib

4.4 Dataset Sources

Our dataset was chosen from Kaggle. It is an online community foundation for AI enthusiasts and information researchers. It enables users to collaborate on projects, locate publicly available datasets, access GPU notepads, and other things.

Our data collection consists of roughly 8000 photos from the Flickr image database. There are roughly five descriptions for each photograph, each of which offers a distinct angle on the picture. Many researchers and programmers utilise this common dataset while creating image captioning models.

Chapter 5

Implementation Details

5.1 Languages And Libraries

- **HTML:**HyperText Markup Language, also known as HTML, is a markup language that enables online users to create and customise various page elements, such as headers, tables, and links, using elements, tags, and attributes. Innovations like Cascading Style Sheets (CSS) and pre-arranging languages like JavaScript tend to aid in this[14]. This would be based on the language I would use to deploy my three models of sentiment analysis on the website I would build.

- **NumPy[9]:** This Python library is available for free. Multi-dimensional arrays are Python objects that are used by NumPy. In essence, arrays are collections of values with one or more dimensions. Ndarray, which stands for n-dimensional array, is another name for the NumPy array data structure. In contrast to working with lists, opening matrices, which are how datasets are typically constructed, is significantly simpler when using NumPy. Numerous processes in this context use Numpy. All of my project's numerical calculations are performed using this library.

- **Tensorflow[2]:**An open source framework is Tensorflow. It was primarily intended to be a neural network library, but as technology has advanced, it is now capable of much more. It is a library for machine learning. This is the Keras cover that helped with model designing and value fitting; it is the base library that we utilised to create our model.

5.2 Setup Used

- **Flask[16]:** It is a small Python-coded net design. It has the moniker microframework. It doesn't require specific tools or libraries as a result. Information abstraction layer, type validation, and other components where pre-existing third-party libraries provide common functionality are not present. It has been utilised to establish a connection between the models and the website and is also in charge of returning the HTML pages in accordance with the output.

- **Kaggle Notebooks[17]:**A completely cloud-based Jupyter notebook environment is available for free at Kaggle Notebbok. In particular, it doesn't require a plan, and the notebooks you create can be simultaneously changed by your colleagues - similar to how you can change reports in Google Docs. In order to provide quick results, this project uses the free GPU of Kaggle for training the Video and Audio modes.

- **Spyder[18]:** It is a free and open-source logical environment designed by and for researchers, professionals, and information examiners. It was created in Python. It has a wonderful combination of high-level modifying, testing, troubleshooting, and profiling utility from a comprehensive development tool. All of the mathematical work is completed over here, and I've utilised this to work on all of my Python files.

- **VS Code[19]:** One of the greatest text editors for coding in almost all languages is VS Code, a lightweight programme. You can use it to programme in a variety of languages, including Python, Java, C++, JavaScript, and more. A source code editor called Visual Studio Code assists companies in creating and troubleshooting online apps that operate on Windows, Linux, and macOS. It is a text editor programme for source code.

5.3 Model Training Implementations

The steps that we went through to train our model:

- We started with the picture dataset since our dataset is split into 2 files with corresponding text and photos.
- We handled photos carefully and read the accompanying illustrations.
- Pictures are downsized to an appropriate element of 224x224 and reshaped to 224x224x3 during the picture pre-handling process. Third consideration: RGB component.
- Pre-handled images are run through the Renet50 model to produce element vectors with a 2048-bit length.
- These vectors are stored against their image.
- After that, we prepare our text dataset..
  Captions are initially made lowercase before being read.
- In this, feature vectors created during pre-processing are first translated into captions for an image.
- Captions are initially made lowercase before being read.
- Then, at the beginning and end of the captions, respectively, strings like "startofseq" and "endofseq" are appended.

- The addition of "startofseq" gives us a place to start when using the deployed model.
- 'endofseq' is inserted to signal the end of the prediction.
- The words in these captions are then separated into frequency groups and counted.
- A dictionary is then built by indexing these words and tokens.
- In captions, the tokens are swapped out for their indices.
- Then, the tokenized captions and image feature vectors are mapped together.
- The architecture for our model, which is composed of an image-model and a language-model, is then created.
- These models are combined, and the output layer of the aforementioned architecture is a Dense LSTM layer with a softmax activation layer.
- In this, we measure misfortune using categorical$_{c}rossentropy, RMSpropasanenhancer, andexactn$ $processeddatawasthensenttoourarchitecture, wherethemodelwastrained.$

5.4 Web Site Implementation

The procedures we followed to create the website were as follows:

We developed our website using the lightweight flask framework. When a user accesses our endpoint at http://127.0.0.1:5000/, they are forwarded to the home page.

A user is prompted to post any photograph they wish a caption for on the website.After uploading an image, the user clicks the predict button, and a caption is generated for the picture.Under the hood, when the predict button is clicked, the user's request for our server (localhost) includes an image parameter head, which we utilise to pre-process the user's image.

Similar to how it happened during model training, image pre-processing takes place.The image is put through a similar model architecture to that used in mode training after it has been preprocessed.The only input that differs is textual information. 'Startofseq' and 'Endofseq' were supplied as the prefix and suffix for complete captions while the model was being trained. 'startofseq' is the sole text input we pass during user testing.The word "startofseq" is used to make a list of text predictions.Once the full caption has been predicted, the user is given the result.We employed trained model$_vocabularyandmodel_weightsinourdesignforthepredictionmodel.$

5.5 Deployments

The main goal of the project is to provide a location where we may test all of its possibilities. The final step in assisting us in accomplishing this mission is this deployment.

On the local server, we built a website that will handle the entire project. Flask will help us execute our project on the local network so that all of the dependencies can be utilised at once by utilising the model we developed throughout the training period.

The results of the implementation of all the stages covered here are shown below together with the finished local web server.

Chapter 6

Result And Analysis

The Web App has deployed the models on a local server and ran it using Flask , results of models is shown in below :-

6.1 Landing Page

Fig 6.1 is Home Page The generated sentence are shown in Fig 6.2 Generated sentences are “ a crowd watching a crowd of people . at night . it ...“ , Fig 6.2 shows the image captions generated for the image.

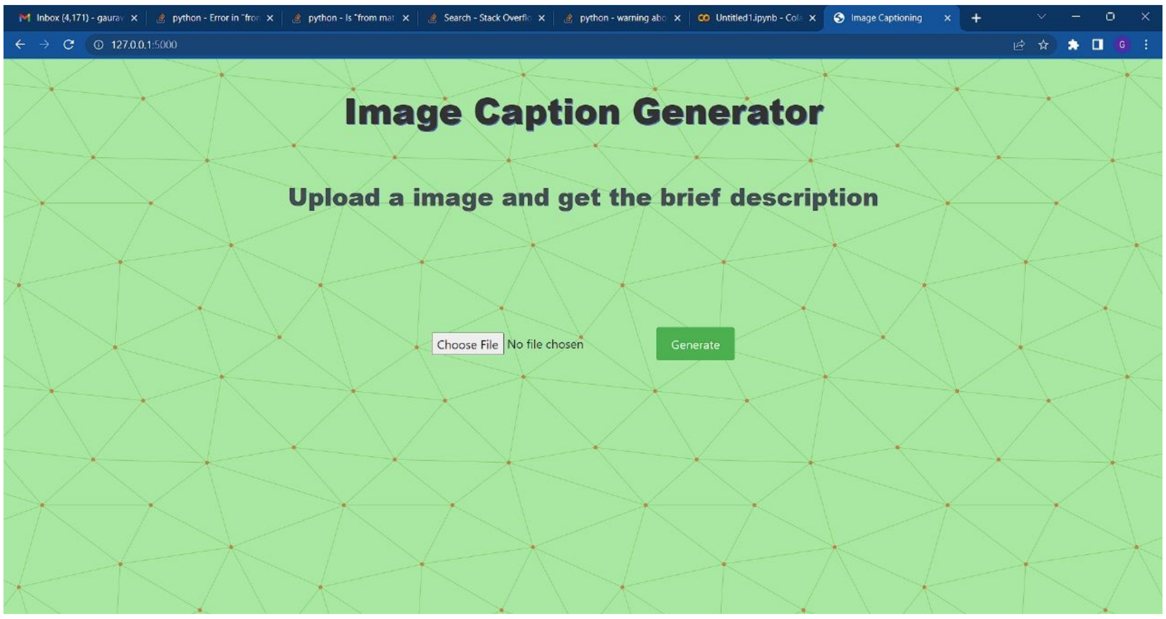


FIGURE 6.1: Home Page

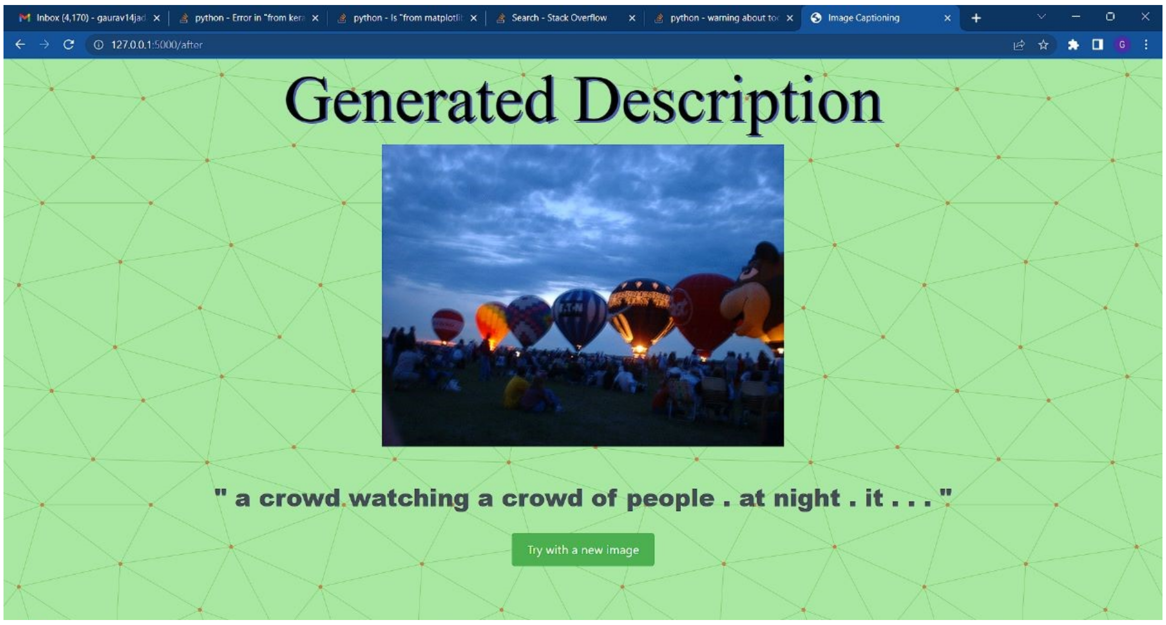

Generated Description
" a crowd watching a crowd of people . at night . it . . . "
Try with a new image

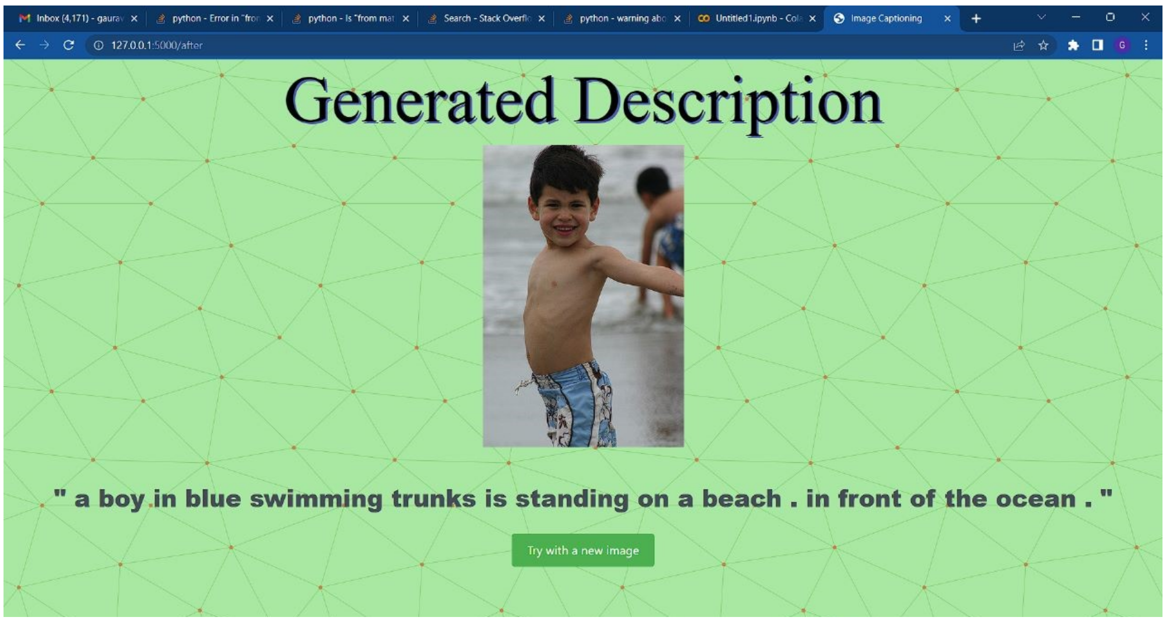
FIGURE 6.2
shows image of captions generated for the image

6.2 Caption Accuracy

We are dividing our captions into 3 categories of High, Medium and Low accuracy. Following are examples of each category:

- High Accuracy Captions
- Medium Accuracy Captions
- Low Accuracy Captions

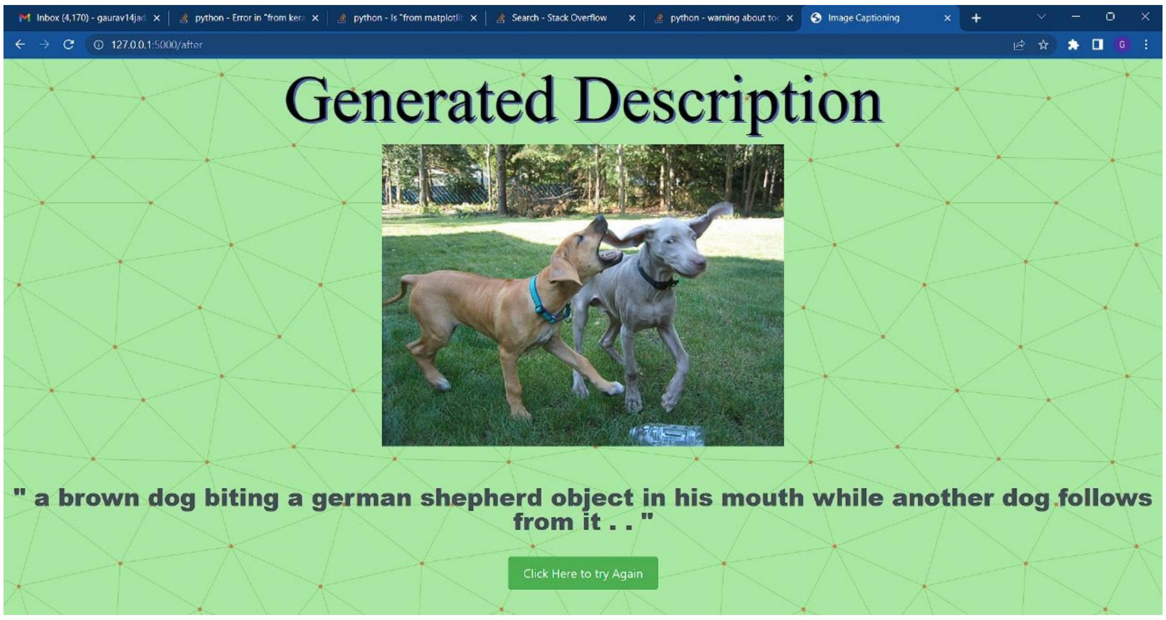
High Accuracy Captions



" a boy in blue swimming trunks is standing on a beach . in front of the ocean . "

FIGURE 6.3

In this image shown in fig 6.3, model generated caption "a boy in blue swimming trunks is standing on a beach.in front of the ocean" . In this, we can see that model captured the image scenario accurately and gave an appropriate sense of the image. The only weak point is syntax of a sentence.

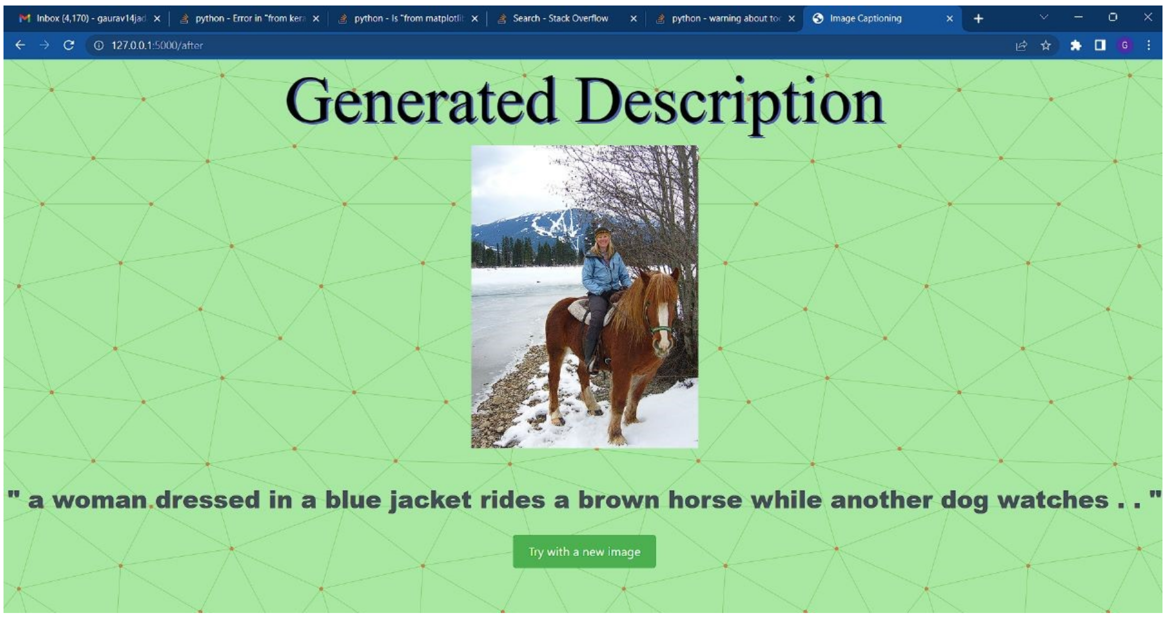

Inbox (4,170) - gaurav14jad
python - Error in "from ker
python - Is "from matplotl
Search - Stack Overflow
python - warning about to
Image Captioning
127.0.0.1:5000/after
Generated Description
Click Here to try Again

" a brown dog biting a german shepherd object in his mouth while another dog follows from it . . "

FIGURE 6.4

In this image shown in fig 6.4, model generated caption "a brown dog biting a german shepherd object in his mouth while another dog follows from it..". In this image, the model correctly senses two dogs with color and their actions also.

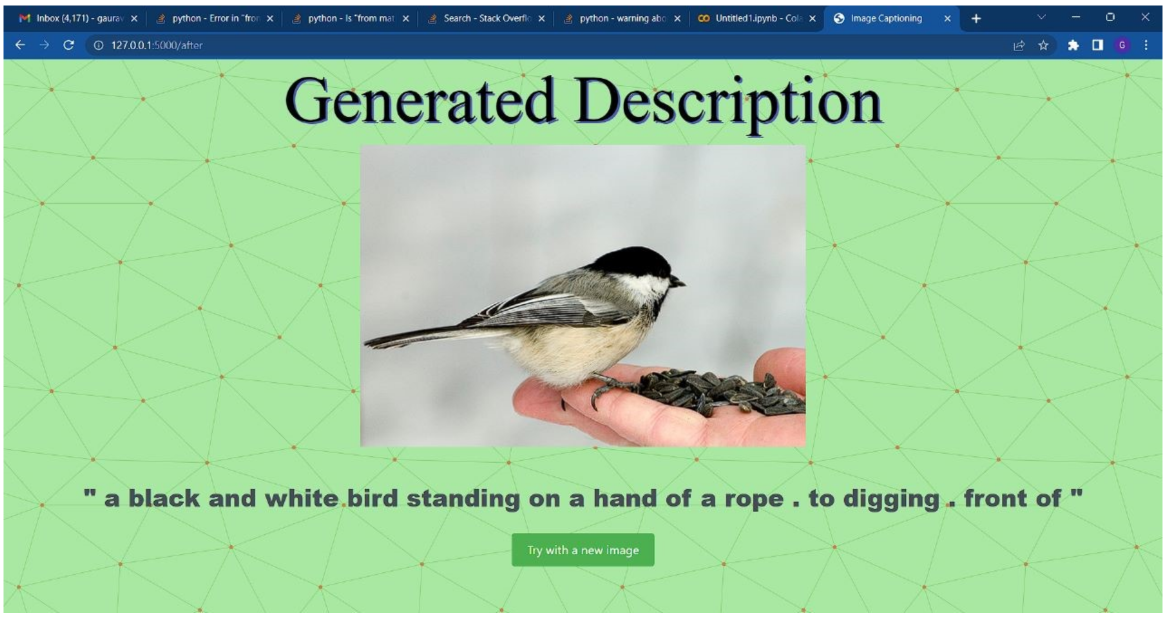
Medium Accuracy Captions



" a woman dressed in a blue jacket rides a brown horse while another dog watches . . "

FIGURE 6.5

In this image shown in fig 6.5, model generated caption "a woman dressed in a blue jacket rides a brown horse while another dog watches.."

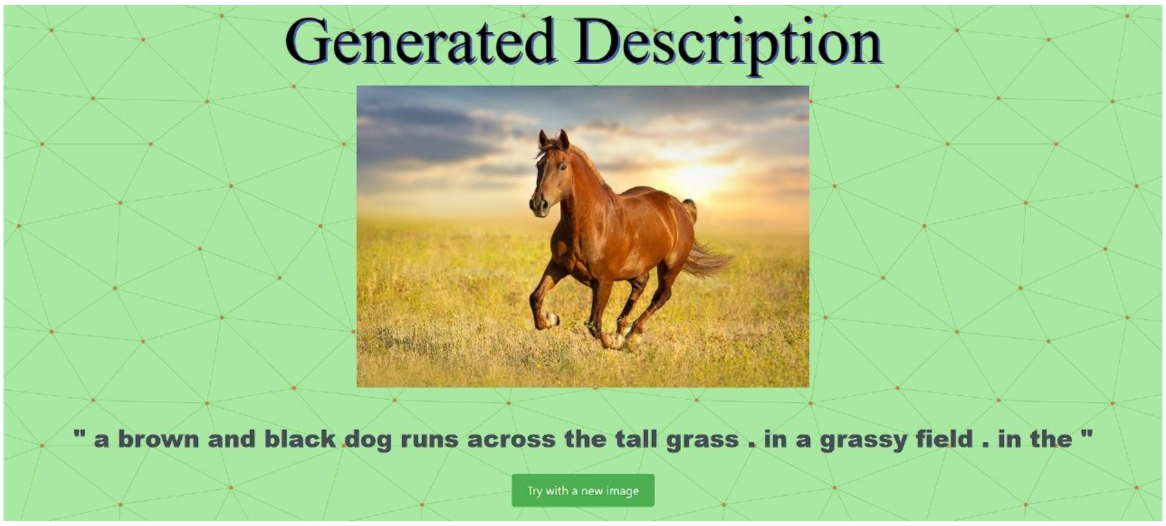


" a black and white bird standing on a hand of a rope . to digging . front of "

FIGURE 6.6

In this image shown in fig 6.6, model generated caption "a black and white bird standing on a hand of a rope. to digging. front of". In this image, model is successfully captured bird with color but added wrong caption like rope

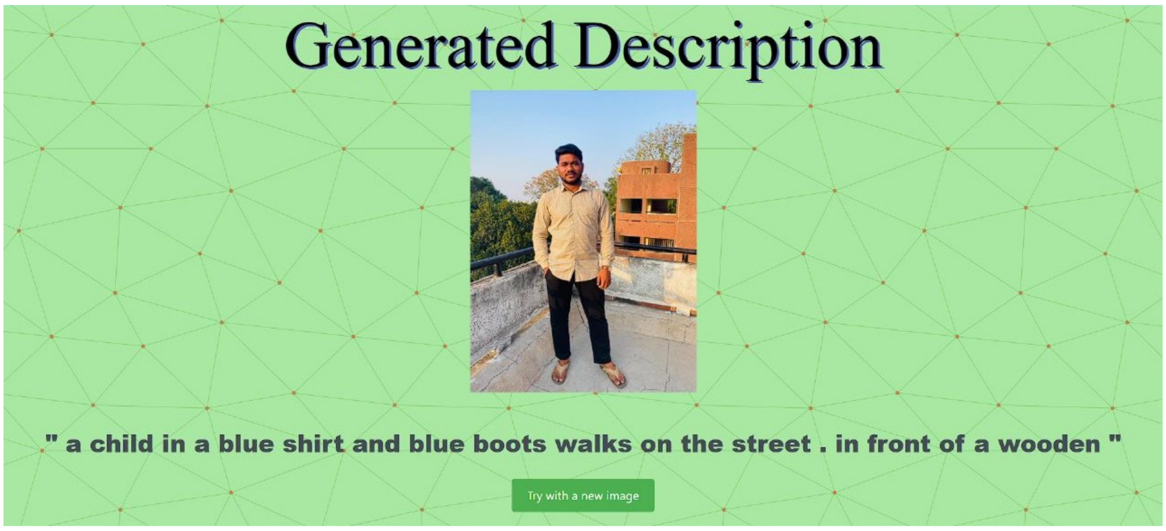
Low Accuracy Captions



" a brown and black dog runs across the tall grass . in a grassy field . in the "

Figure 6.7

In this image shown in fig 6.7, model generated caption "a brown and black dog runs across the tall grass. in a grassy field. in the". In this we see that model only able to some aspect of images such as "tall grass" but has misjudged horse as a dog. In this image shown in fig 6.8,



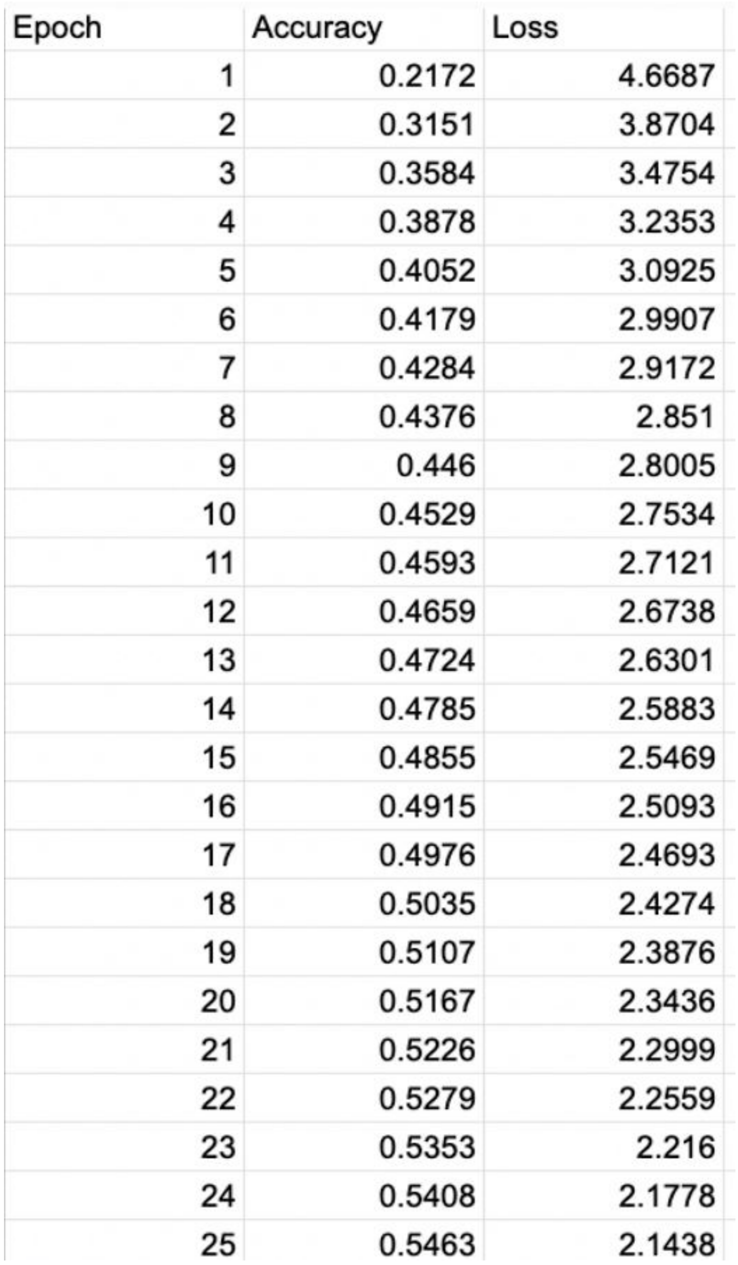
" a child in a blue shirt and blue boots walks on the street . in front of a wooden "

FIGURE 6.8

model generated caption "a child in a blue shirt and blue boots walks on the street . in front of a wooden". In this we can see that model gave completely wrong captions

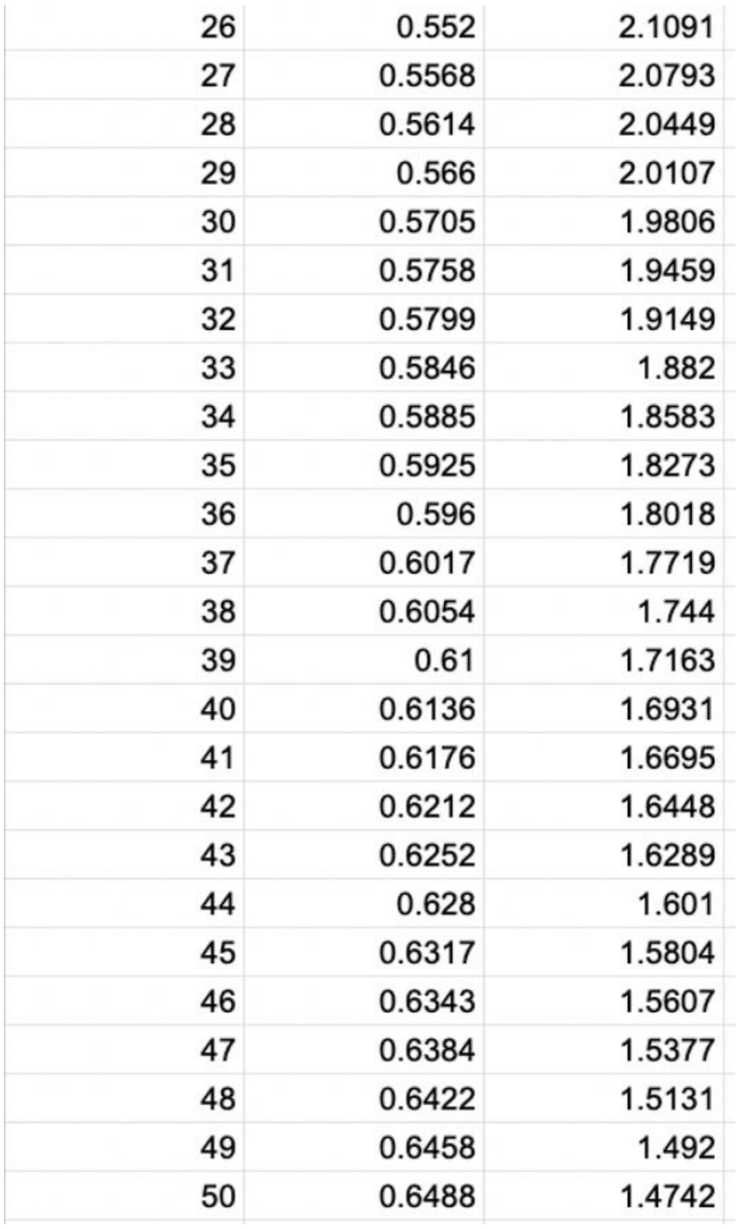
6.3 Analysis

We trained our model on a dataset of around 8000 flickr images. Model training took about 19 hours. We were able to obtain an accuracy of about 65 percent for our model on training data. **fig 6.9 , 6.10 shows the data show result of each Epoch Training**



Epoch	Accuracy	Loss
1	0.2172	4.6687
2	0.3151	3.8704
3	0.3584	3.4754
4	0.3878	3.2353
5	0.4052	3.0925
6	0.4179	2.9907
7	0.4284	2.9172
8	0.4376	2.851
9	0.446	2.8005
10	0.4529	2.7534
11	0.4593	2.7121
12	0.4659	2.6738
13	0.4724	2.6301
14	0.4785	2.5883
15	0.4855	2.5469
16	0.4915	2.5093
17	0.4976	2.4693
18	0.5035	2.4274
19	0.5107	2.3876
20	0.5167	2.3436
21	0.5226	2.2999
22	0.5279	2.2559
23	0.5353	2.216
24	0.5408	2.1778
25	0.5463	2.1438

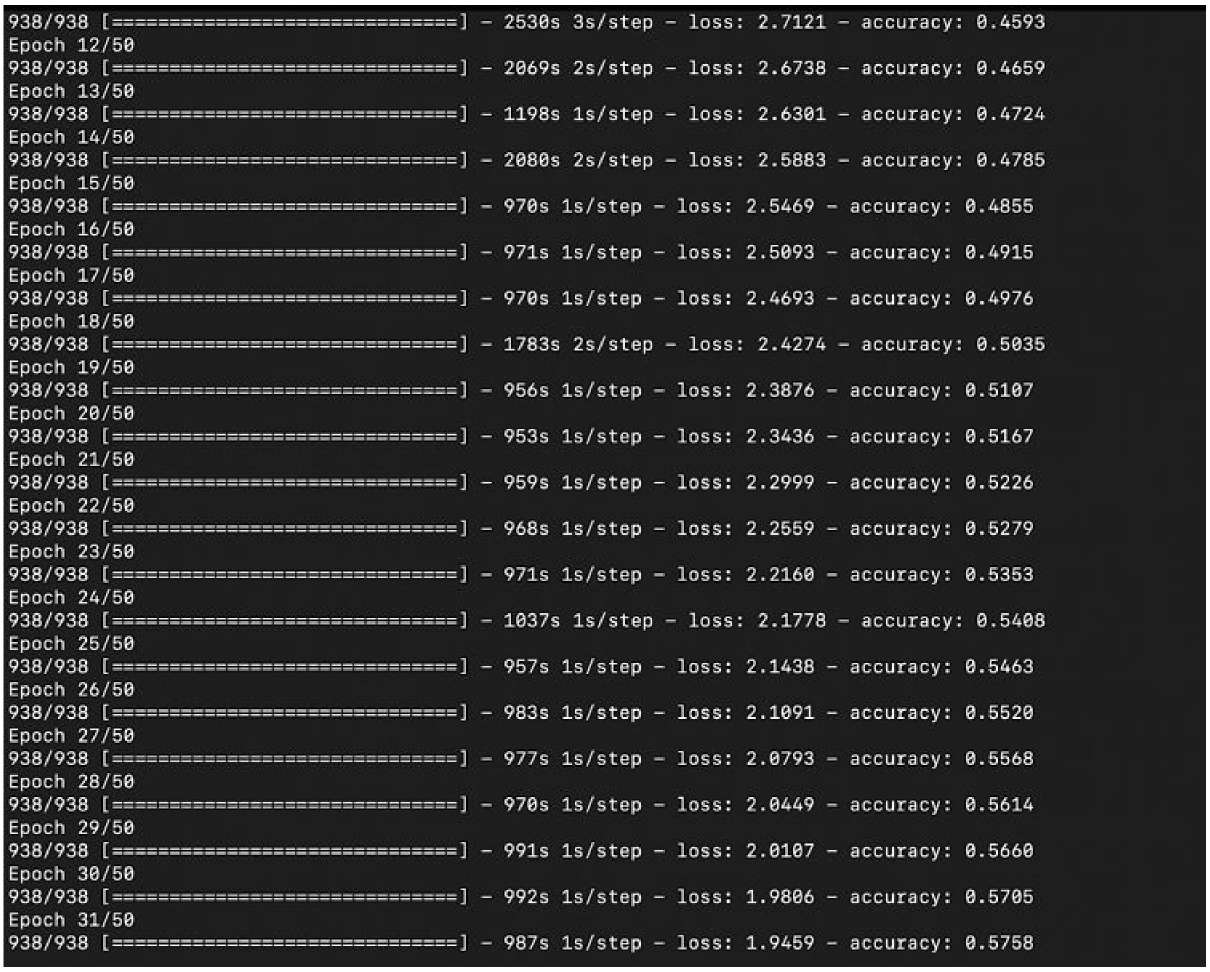
Figure 6.9



26	0.552	2.1091
27	0.5568	2.0793
28	0.5614	2.0449
29	0.566	2.0107
30	0.5705	1.9806
31	0.5758	1.9459
32	0.5799	1.9149
33	0.5846	1.882
34	0.5885	1.8583
35	0.5925	1.8273
36	0.596	1.8018
37	0.6017	1.7719
38	0.6054	1.744
39	0.61	1.7163
40	0.6136	1.6931
41	0.6176	1.6695
42	0.6212	1.6448
43	0.6252	1.6289
44	0.628	1.601
45	0.6317	1.5804
46	0.6343	1.5607
47	0.6384	1.5377
48	0.6422	1.5131
49	0.6458	1.492
50	0.6488	1.4742

Figure 6.10

Model training output using the Keras deep learning library and the Python programming language from fig:6.11 to fig:6.13



```
938/938 [==============================] - 2530s 3s/step - loss: 2.7121 - accuracy: 0.4593
Epoch 12/50
938/938 [==============================] - 2069s 2s/step - loss: 2.6738 - accuracy: 0.4659
Epoch 13/50
938/938 [==============================] - 1198s 1s/step - loss: 2.6301 - accuracy: 0.4724
Epoch 14/50
938/938 [==============================] - 2080s 2s/step - loss: 2.5883 - accuracy: 0.4785
Epoch 15/50
938/938 [==============================] - 970s 1s/step - loss: 2.5469 - accuracy: 0.4855
Epoch 16/50
938/938 [==============================] - 971s 1s/step - loss: 2.5093 - accuracy: 0.4915
Epoch 17/50
938/938 [==============================] - 970s 1s/step - loss: 2.4693 - accuracy: 0.4976
Epoch 18/50
938/938 [==============================] - 1783s 2s/step - loss: 2.4274 - accuracy: 0.5035
Epoch 19/50
938/938 [==============================] - 956s 1s/step - loss: 2.3876 - accuracy: 0.5107
Epoch 20/50
938/938 [==============================] - 953s 1s/step - loss: 2.3436 - accuracy: 0.5167
Epoch 21/50
938/938 [==============================] - 959s 1s/step - loss: 2.2999 - accuracy: 0.5226
Epoch 22/50
938/938 [==============================] - 968s 1s/step - loss: 2.2559 - accuracy: 0.5279
Epoch 23/50
938/938 [==============================] - 971s 1s/step - loss: 2.2160 - accuracy: 0.5353
Epoch 24/50
938/938 [==============================] - 1037s 1s/step - loss: 2.1778 - accuracy: 0.5408
Epoch 25/50
938/938 [==============================] - 957s 1s/step - loss: 2.1438 - accuracy: 0.5463
Epoch 26/50
938/938 [==============================] - 983s 1s/step - loss: 2.1091 - accuracy: 0.5520
Epoch 27/50
938/938 [==============================] - 977s 1s/step - loss: 2.0793 - accuracy: 0.5568
Epoch 28/50
938/938 [==============================] - 970s 1s/step - loss: 2.0449 - accuracy: 0.5614
Epoch 29/50
938/938 [==============================] - 991s 1s/step - loss: 2.0107 - accuracy: 0.5660
Epoch 30/50
938/938 [==============================] - 992s 1s/step - loss: 1.9806 - accuracy: 0.5705
Epoch 31/50
938/938 [==============================] - 987s 1s/step - loss: 1.9459 - accuracy: 0.5758
```

Figure 6.11

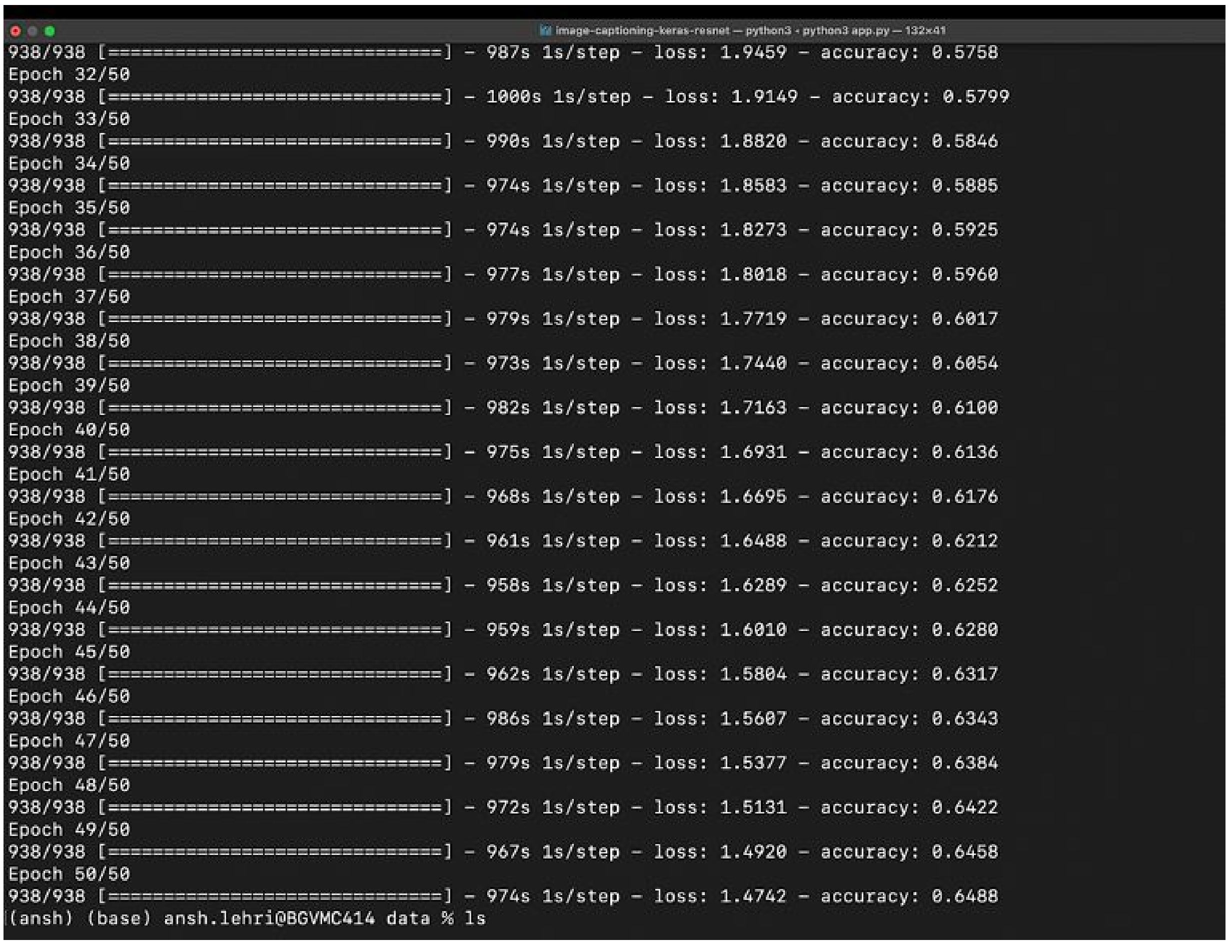
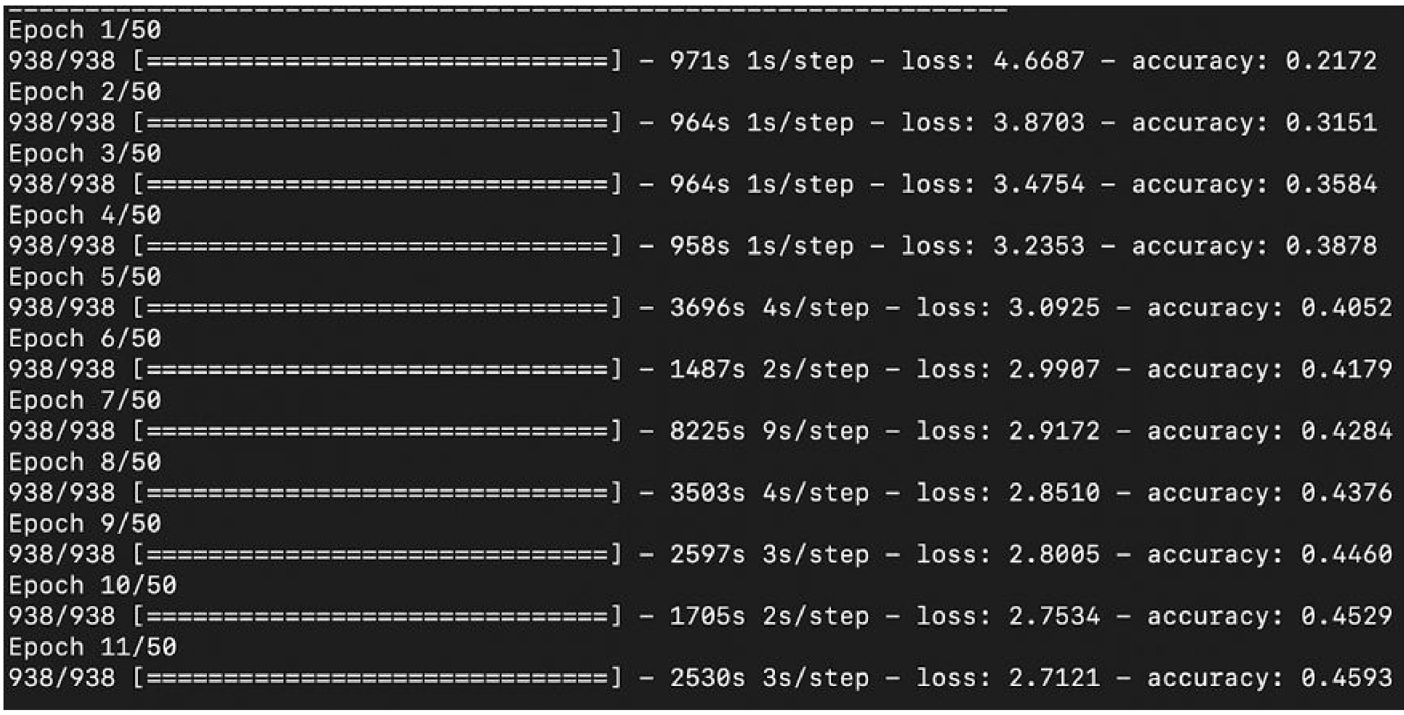


image-captioning-keras-resnet — python3 • python3 app.py — 132×41

```
938/938 [==============================] - 987s 1s/step - loss: 1.9459 - accuracy: 0.5758
Epoch 32/50
938/938 [==============================] - 1000s 1s/step - loss: 1.9149 - accuracy: 0.5799
Epoch 33/50
938/938 [==============================] - 990s 1s/step - loss: 1.8820 - accuracy: 0.5846
Epoch 34/50
938/938 [==============================] - 974s 1s/step - loss: 1.8583 - accuracy: 0.5885
Epoch 35/50
938/938 [==============================] - 974s 1s/step - loss: 1.8273 - accuracy: 0.5925
Epoch 36/50
938/938 [==============================] - 977s 1s/step - loss: 1.8018 - accuracy: 0.5960
Epoch 37/50
938/938 [==============================] - 979s 1s/step - loss: 1.7719 - accuracy: 0.6017
Epoch 38/50
938/938 [==============================] - 973s 1s/step - loss: 1.7440 - accuracy: 0.6054
Epoch 39/50
938/938 [==============================] - 982s 1s/step - loss: 1.7163 - accuracy: 0.6100
Epoch 40/50
938/938 [==============================] - 975s 1s/step - loss: 1.6931 - accuracy: 0.6136
Epoch 41/50
938/938 [==============================] - 968s 1s/step - loss: 1.6695 - accuracy: 0.6176
Epoch 42/50
938/938 [==============================] - 961s 1s/step - loss: 1.6488 - accuracy: 0.6212
Epoch 43/50
938/938 [==============================] - 958s 1s/step - loss: 1.6289 - accuracy: 0.6252
Epoch 44/50
938/938 [==============================] - 959s 1s/step - loss: 1.6010 - accuracy: 0.6280
Epoch 45/50
938/938 [==============================] - 962s 1s/step - loss: 1.5804 - accuracy: 0.6317
Epoch 46/50
938/938 [==============================] - 986s 1s/step - loss: 1.5607 - accuracy: 0.6343
Epoch 47/50
938/938 [==============================] - 979s 1s/step - loss: 1.5377 - accuracy: 0.6384
Epoch 48/50
938/938 [==============================] - 972s 1s/step - loss: 1.5131 - accuracy: 0.6422
Epoch 49/50
938/938 [==============================] - 967s 1s/step - loss: 1.4920 - accuracy: 0.6458
Epoch 50/50
938/938 [==============================] - 974s 1s/step - loss: 1.4742 - accuracy: 0.6488
(ansh) (base) ansh.lehri@BGVMC414 data % ls
```

FIGURE 6.12



```
=================================================================
Epoch 1/50
938/938 [==============================] - 971s 1s/step - loss: 4.6687 - accuracy: 0.2172
Epoch 2/50
938/938 [==============================] - 964s 1s/step - loss: 3.8703 - accuracy: 0.3151
Epoch 3/50
938/938 [==============================] - 964s 1s/step - loss: 3.4754 - accuracy: 0.3584
Epoch 4/50
938/938 [==============================] - 958s 1s/step - loss: 3.2353 - accuracy: 0.3878
Epoch 5/50
938/938 [==============================] - 3696s 4s/step - loss: 3.0925 - accuracy: 0.4052
Epoch 6/50
938/938 [==============================] - 1487s 2s/step - loss: 2.9907 - accuracy: 0.4179
Epoch 7/50
938/938 [==============================] - 8225s 9s/step - loss: 2.9172 - accuracy: 0.4284
Epoch 8/50
938/938 [==============================] - 3503s 4s/step - loss: 2.8510 - accuracy: 0.4376
Epoch 9/50
938/938 [==============================] - 2597s 3s/step - loss: 2.8005 - accuracy: 0.4460
Epoch 10/50
938/938 [==============================] - 1705s 2s/step - loss: 2.7534 - accuracy: 0.4529
Epoch 11/50
938/938 [==============================] - 2530s 3s/step - loss: 2.7121 - accuracy: 0.4593
```

Figure 6.13

6.4 Limitations

Hardware Limitations:

- Machine used has limited GPUs.
- Machine used to train the model has 16 GB RAM. To train bigger and better datasets, a machine with higher RAM is required with hisg computing capacity.

Dataset Limitations:

- The only dataset available was of Flickr.
- Flickr dataset had less variety in terms of nature of images. Most of images were of animals, scenic beauty, sports etc.
- Images present are only high resolution.

Implementation Limitations:

- Due to machine constraints, we could not try hyper-parameter tuning to a large extent.
- Different architectures could not run on the machine.
- BLEU to gauge exactness couldn't be carried out.
- Because of the limited dataset, vocabulary generated for the model was very constrained. As it had not many words to choose from, this resulted in many bad captions.

Because of the limited dataset, vocabulary generated for the model was very constrained. As it had not many words to choose from, this resulted in many bad captions.

These limitations can be overcome by following measures:

- Utilizing strong machines with high RAM and registering limit.
- Using variety of datasets which can have images with high to low resolution, nature of images can be different as well.
- Applying hyper-parameter tuning and using different architectures.
- Using word embeddings to increase our dictionary.
- Using large datasets to generate bigger vocabulary dictionary.

Chapter 7

Conclusion And Future Scope

The growing topic of image captioning is still the subject of many investigations. By reducing complex models to simple structures, recent work using deep learning approaches has increased the accuracy of photograph captioning. It is possible to broaden the potential uses of visual comprehension in the realms of medicine, security, the military, and other disciplines by improving the efficiency of content-based picture retrieval. The fake system and inquiry techniques of picture inscription may additionally advance the concept and usefulness of photo comment and apparent question addressing (VQA), go media recovery, video subtitling, and video sharing, which has important educational and rational benefit.

7.1 Use case and future Scopes

- **Medical Application :** This type can be used by someone who is blind to learn about their environment along with some sunglasses, cameras, and hearing aids. One illustration of this application is Horus Technology, which is still in the development phase and is now working on a comparable project with NVIDIA.

- **Intelligent monitoring :** Intelligent monitoring allows the computer to identify and analyse how people and cars are behaving in the captured scene. Under the correct circumstances, the computer can emit alarms to prompt a human to act in an emergency and avert tragic accidents.

  One of the cutting-edge technologies in development attaches a palm-sized device to a camera to give the driver a view of the road while alerting them to potential hazards and lowering accident rates. This technology is being developed by the businesses CPRI and Intel IIIT Hyderabad.

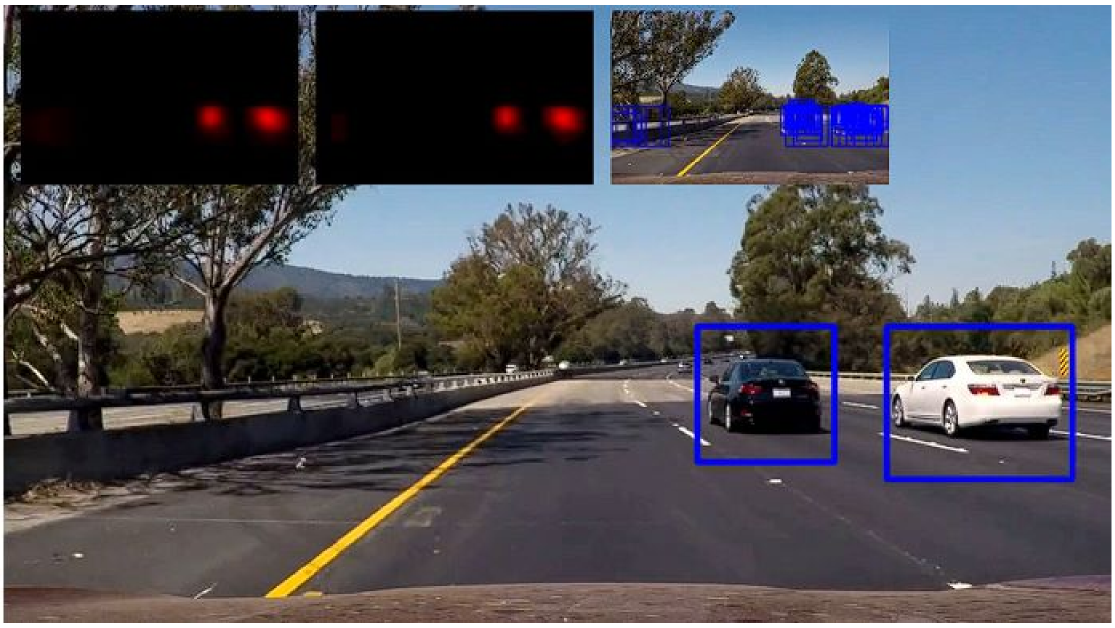


FIGURE 7.1

- **Campus Level Implementationing :**

  To detect any ludicrous activities, this model can be combined with cameras and alarm systems like SMS, sirens, etc. The aforementioned robbery is being detected by image cameras.

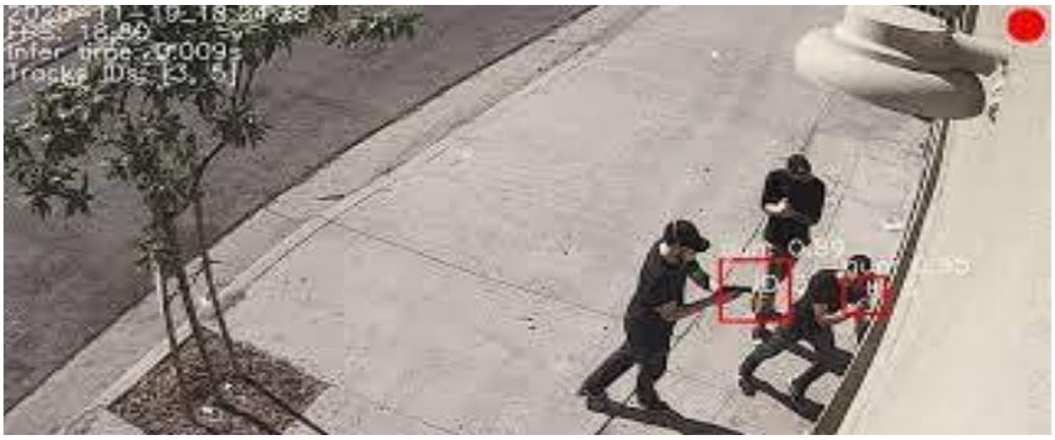
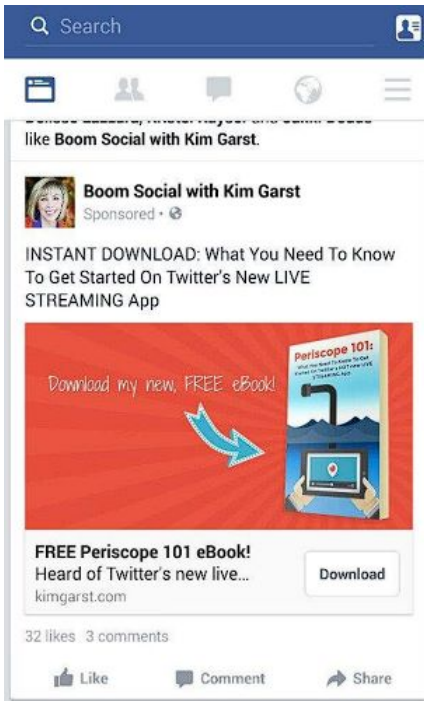


FIGURE 7.2

- **Social Media Implementationing :**

  Facebook and other social media platforms can tell from a picture where you are (ocean side, restaurant, etc.), what you're wearing (variety), and most importantly, what you're doing (as it were). This enables businesses to increase promotions for a certain client who benefits them.



Search

like **Boom Social with Kim Garst**.

Boom Social with Kim Garst
Sponsored ·

INSTANT DOWNLOAD: What You Need To Know To Get Started On Twitter's New LIVE STREAMING App

FREE Periscope 101 eBook!
Heard of Twitter's new live...
kimgarst.com

Download

32 likes 3 comments

Like Comment Share

FIGURE 7.3

Appendix A

Appendix

train.py

import pandas as pd

import cv2

import os

from glob import glob

import tensorflow as tf

from keras.applications.resnet import ResNet50

from keras.models import Model

from keras.utils import pad$_sequences$

$fromkeras.utilsimportto_categorical$

$fromkeras.modelsimportModel, Sequential$

$fromkeras.layersimportDense$

$fromkeras.layersimportLSTM$

$fromkeras.layersimportEmbedding$

$fromkeras.layersimportDense, LSTM, TimeDistributed, Embedding, Activation, RepeatVector, Con$

$fromkeras.modelsimportSequential, Model$

images$_path =' C : /Users/Gaurav/Desktop/G-17ImageCaptionGenerator/code/training_model/Flic$

$images_path =' onspot/Images/'$

$images = glob(images_path +' *.jpg')$

import matplotlib.pyplot as plt

plt.close('all')

for i in range(len(images)):

fig, ax = plt.subplots() Create a new figure and axis object

img = cv2.imread(images[i])

ax.imshow(img)

plt.close(fig)

incept$_model = ResNet50(include_top = True)$

last = incept$_model.layers[-2].output$

$modele = Model(inputs = incept_model.input, outputs = last)$

images$_features =$

$count = 0$

$foriinimages :$

$img = cv2.imread(i)$

$img = cv2.cvtColor(img, cv2.COLOR_{B}GR2RGB)$

$img = cv2.resize(img, (224, 224))$

img = img.reshape(1,224,224,3)

pred = modele.predict(img).reshape(2048,)

img$_{n}ame = i.split('/')[-1]$

images$_{f}eatures[img_{n}ame] = pred$

count += 1

if count > 8000:

break

elif count print(count)

caption$_{p}ath =' C : /Users/Gaurav/Desktop/G{-}17ImageCaptionGenerator/code/training_{m}odel/Fli$

$caption_{p}ath =' onspot/text/data.txt'$

$captions = open(caption_{p}ath,' rb').read().decode('utf - 8').split('')$

captions$_{d}ict =$

$foriincaptions :$

$try :$

$img_name = i.split($[illegible]$)[0][: -2]caption = i.split($[illegible]$)[1]ifimg_nameinimages_features : ifimg_namenotincap$

import matplotlib.pyplot as plt

for i in range(len(images)):

plt.figure()

img$_n ame = images[i]$

img = cv2.imread(img$_n ame$)

$img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)$

$plt.xlabel(captions_dict[img_name.split('/')[-1]])$

$plt.imshow(img)$

def preprocessed(txt):

modified = txt.lower()

modified = 'startofseq ' + modified + ' endofseq'

return modified

for k,v in captions$_d ict.items() :$

$forvvinv :$

$captions_dict[k][v.index(vv)] = preprocessed(vv)$

count$_w ords =$

$fork, vvincaptions_dict.items() :$

for v in vv: for word in v.split(): if word not in count$_words :$

count$_words[word] = 0$

else: count$_words[word]+ = 1$

THRESH = -1 count = 1 new$_dict = fork, vincount_words.items() : ifcount_words[k] > THRESH :$
$new_dict[k] = countcount+ = 1$

new$_dict['< OUT >'] = len(new_dict)$

captions$_backup = captions_dict.copy()captions_dict = captions_backup.copy()fork, vvincaptions_dict.items$
$forvinvv : encoded = []forwordinv.split() : ifwordnotinnew_dict : encoded.append(new_dict['<$
$OUT >'])else : encoded.append(new_dict[word])$

captions$_dict[k][vv.index(v)] = encoded$

MAX$_LEN = 0fork, vvincaptions_dict.items() : forvinvv : iflen(v) > MAX_LEN : MAX_LEN =$
$len(v)print(v)$

Batch$_size = 5VOCAB_SIZE = len(new_dict)$

def generator(photo, caption): n$_samples = 0$

X = [] y$_in = []y_out = []$

for k, vv in caption.items(): for v in vv: for i in range(1, len(v)): X.append(photo[k])

in$_seq = [v[: i]]out_seq = v[i]$

in$_seq = pad_sequences(in_seq, maxlen = MAX_LEN, padding =' post', truncating =' post')[0]out_seq =$
$to_categorical([out_seq], num_classes = VOCAB_SIZE)[0]$

y$_in.append(in_seq)y_out.append(out_seq)$

return X, y$_in, y_out$

X, y$_in, y_out = generator(images_features, captions_dict)$

X = np.array(X) y$_in = np.array(y_in, dtype =' float64')y_out = np.array(y_out, dtype =' float64')$

embedding$_size = 128max_len = MAX_LENvocab_size = len(new_dict)$

image$_model = Sequential()$

image$_model.add(Dense(embedding_size, input_shape = (2048,), activation =' relu'))image_model.add(Re$

image$_model.summary()$

language$_model = Sequential()$

language$_model.add(Embedding(input_dim = vocab_size, output_dim = embedding_size, input_length =$
$max_len))language_model.add(LSTM(256, return_sequences = True))language_model.add(TimeDistribu$

language$_model.summary()$

conca = Concatenate()([image$_model.output, language_model.output])x = LSTM(128, return_sequences =$
$True)(conca)x = LSTM(512, return_sequences = False)(x)x = Dense(vocab_size)(x)out =$
$Activation('softmax')(x)model = Model(inputs = [image_model.input, language_model.input], outputs =$
$out)$

model.load$_weights(”../input/model_weights.h5”)model.compile(loss =' categorical_crossentropy', optim$
$RMSprop', metrics = ['accuracy'])$

model.fit([X, y$_in], y_out, batch_size = 512, epochs = 100)$

inv$_dict = v : kfork, vinnew_dict.items()$

model.save('./img$_caption_model_v1.h5')$

model.save$_weights('./img_caption_model_v1_weights.h5')$

np.save('./img$_caption_model_v1_vocab.npy',\ new_dict)$

App.py

from flask import Flask, render$_template, request$

$importcv2$

$fromkeras.modelsimportload_model$

$importnumpyasnp$

$fromkeras.applications.resnetimportResNet50$

$fromkeras.layersimportDense, LSTM, TimeDistributed, Embedding, Activation, RepeatVector, Con$

$fromkeras.modelsimportSequential, Model$

$fromkeras.preprocessingimportimage, sequence$

$importcv2$

$fromkeras.utilsimportpad_sequences$

$fromtqdmimporttqdm$

app = Flask($_name_)$

app.config['SEND$_FILE_MAX_AGE_DEFAULT'] = 1$

vocab = np.load('mine$_vocab.npy', allow_pickle = True)$

vocab = vocab.item()

inv$_vocab = v : kfork, vinvocab.items()$

print("+" * 50)

print("vocabulary loaded")

embedding$_{s}ize = 128$

vocab$_{s}ize = len(vocab)$

max$_{l}en = 40$

image$_{m}odel = Sequential()$

image$_{m}odel.add(Dense(embedding_{s}ize, input_{s}hape = (2048,), activation =' relu'))$

image$_{m}odel.add(RepeatVector(max_{l}en))$

language$_{m}odel = Sequential()$

language$_{m}odel.add(Embedding(input_{d}im = vocab_{s}ize, output_{d}im = embedding_{s}ize, input_{l}ength = max_{l}en))$

language$_{m}odel.add(LSTM(256, return_{s}equences = True))$

language$_model.add(TimeDistributed(Dense(embedding_size)))$

conca = Concatenate()([image$_model.output, language_model.output]$)

x = LSTM(128, return$_sequences = True)(conca)$

x = LSTM(512, return$_sequences = False)(x)$

x = Dense(vocab$_size)(x)$

out = Activation('softmax')(x)

model = Model(inputs=[image$_model.input, language_model.input], outputs = out$)

model.compile(loss='categorical$_crossentropy', optimizer =' RMSprop', metrics = ['accuracy']$)

model.load$_weights('mine_model_weights.h5'$)

print("=" * 150)

print("MODEL LOADED")

resnet = ResNet50(include$_{t}op = False, weights =' imagenet', input_{s}hape = (224, 224, 3), pooling ='$ $avg')$

app = Flask($_{n}ame_{)}$

app.config['SEND$_{F}ILE_{M}AX_{A}GE_{D}EFAULT'] = 1$

Index.html

<div class="main-background">

<div class="main-head">

<h1 data-shadow='dang!' class="main-caption">Image Caption Generator</h1>

</div>

<div class="main-form">

<form action="url$_{f}or('after')"method =' POST'enctype =' multipart/form - data'style =$ $"margin - top : 150px;"class = "main - form" >$

$< inputtype = "file"name =' file1'class = "main - img" >$

$< inputtype = "submit"name = "btn"value =' Generate'class =' button' >$

$< /form >$

$< /div >$

$< /div >$

Base.html

<!doctype html>

<html lang="en">

<head>

```
<!– Required meta tags –>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">


<!– Bootstrap CSS –>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ
crossorigin="anonymous">


<link rel="stylesheet" href="url_for('static', filename =' xyz.css')” >


<title>Image Captioning</title>
</head>
<body class="background_mainPage”style = ”height : 100vh; ” >


<!– Optional JavaScript –>
<!– jQuery first, then Popper.js, then Bootstrap JS –>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha3
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossori-
gin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV" crossorigin="anon
</body>
```

```
</html>
```

After.html

```
<center>
<h1 class='display-2' style="color: black;">Generated Description </h1>
<div>
</div>
<img src="url_for('static', filename =' file.jpg')" alt = "image" height = 400pxclass = "img" >

< h2class =' head2' >
< span > " < /span > data < span > " < /span >
< /h2 >
< br >
< AHREF =' http : //127.0.0.1 : 5000/'class =' buttonbtn' > Trywithanewimage < /A >
< /center >
```

References

[1] Shuang Liu, Liang Bai,a, Yanli Hu and Haoran Wang. Image Captioning Based on Deep Neural Networks. EITCE (2018)

[2] Lakshminarasimhan Srinivasan, Dinesh Sreekanthan,Amutha A.L. I2T: Image Captioning - A Deep Learning Approach (2018).

[3] Simao Herdade, Armin Kappeler, Kofi Boakye, Joao Soares.Image Captioning: Transforming Objects into Words . San Francisco, CA (2019).

[4] Aishwarya Maroju ,Sneha Sri Doma ,Lahari Chandarlapati , Image Caption Generating Deep Learning Model ,J.N.T.U, Hyderabad , Sreenidhi Institute of Science And Technology (2021).

[5] Zhongliang Yang, Yu-Jin Zhang, Sadaqat ur Rehman, Yongfeng Huang: Image Captioning with Object Detection and Localization, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

[6] Oriol Vinyals , Alexander Toshev , Samy Bengio Dumitu Erhan(2014). Show and Tell : A Neural Image Caption Generator . Google

[7] Andrej Karpathy Li Fei Fei(2015). Deep Visual-Semantic Alignments for Generating Image Descriptions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3128-3137.

[8] Elliott, D., Keller, F. (2013). Image Description using Visual Dependency Representations. EMNLP.

[9] Farhadi, A., Hejrati, M., Sadeghi, M.A., Young, P., Rashtchian, C., Hockenmaier, J., Forsyth, D.A. (2010). Every Picture Tells a Story: Generating Sentences from Images. ECCV.

[10] Fang, H., Gupta, S., Iandola, F.N., Srivastava, R.K., Deng, L., Dollár, P., Gao, J., He, X., Mitchell, M., Platt, J.C., Zitnick, C.L., Zweig, G. (2015). From captions to visual concepts and back. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1473-1482.

[11] Kuznetsova, P., Ordonez, V., Berg, A.C., Berg, T.L., Choi, Y. (2012). Collective Generation of Natural Image Descriptions. ACL.

[12] Lebret, Rémi et al. "Simple Image Description Generator via a Linear PhraseBased Approach." CoRR abs/1412.8419 (2014): n. pag.

[13] Li, S., Kulkarni, G., Berg, T.L., Berg, A.C., Choi, Y. (2011). Composing Simple Image Descriptions using Web-scale N-grams. CoNLL.

[14] Chen, X., Zitnick, C.L. (2015). Mind's eye: A recurrent visual representation 49 for image caption generation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2422-2431.

[15] Donahue, Jeff et al. "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 2625-2634.

Image Caption Generator

ORIGINALITY REPORT

10%	9%	4%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Euhid Aman, Sukarna Jana, Kunal Gurudath Athikary, Ramesh Chinnakurli Suryanarayana. "AI Inspired ATC, Based on ANN and Using NLP", SAE International, 2023 Publication	1%
2	medium.com Internet Source	1%
3	he30.co-aol.com Internet Source	1%
4	github.com Internet Source	1%
5	www.coursehero.com Internet Source	1%
6	api.openalex.org Internet Source	1%
7	dspace.dtu.ac.in:8080 Internet Source	<1%
8	www.irjmets.com Internet Source	<1%

9	rcciit.org.in Internet Source	<1 %
10	ro.ecu.edu.au Internet Source	<1 %
11	dspace.daffodilvarsity.edu.bd:8080 Internet Source	<1 %
12	Sayani Ghatak, Hrithik Paul, Debmitra Ghosh. "Voicebot For Mental Disease Prediction And Treatment Recommendation Using Machine Learning", Institute of Electrical and Electronics Engineers (IEEE), 2023 Publication	<1 %
13	cse.anits.edu.in Internet Source	<1 %
14	ethesis.nitrkl.ac.in Internet Source	<1 %
15	scholars.lib.ntu.edu.tw Internet Source	<1 %
16	ieeexplore.ieee.org Internet Source	<1 %
17	doczz.net Internet Source	<1 %
18	Jiechao Yang, Xuelei Wang, Ruihua Wang, Huanjie Wang. "Combination of Convolutional Neural Networks and Recurrent Neural	<1 %

Networks for predicting soil properties using Vis–NIR spectroscopy", Geoderma, 2020
Publication

19	de.slideshare.net Internet Source	<1 %
20	infocyb.ase.ro Internet Source	<1 %
21	web.archive.org Internet Source	<1 %
22	www.cse.cuhk.edu.hk Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		