# IMAGE CAPTIONING FROM WIKIPEDIA USING DEEP LEARNING FOR MULTILANGUAGE

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

**BYNA ABHISHEK AKRI ( Reg.No - 39110199 )**
**BHIMIREDDY NITHIN REDDY( Reg.No – 39110161 )**


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

# SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY**
**(DEEMED TO BE UNIVERSITY)**
**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE**
**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**
**CHENNAI - 600119**

**APRIL - 2023**

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

---

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **Byna Abhishek Akri (Reg.No - 39110199) and Bhimireddy Nithin Reddy (Reg.No - 39110161)** who carried out the Project Phase-2 entitled **"IMAGE CAPTIONING FROM WIKIPEDIA USING DEEP LEARNING FOR MULTILANGUAGE"** under my supervision from January 2023 to April 2023.

**Internal Guide**

**Dr. A. CHRISTY MCA, Ph.D**

**Head of the Department**

**Dr. L. LAKSHMANAN, M.E., Ph.D.**

**Submitted for Viva voce Examination held on ----------**

**Internal Examiner**

**External Examiner**

# DECLARATION

I, **Byna Abhishek Akri (Reg.No- 39110199),** hereby declare that the Project Phase-2 Report entitled **"IMAGE CAPTIONING FROM WIKIPEDIA USING DEEP LEARNING FOR MULTILANGUAGE"** done by me under the guidance of **Dr. A. Christy, MCA, Ph.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

**DATE:**

**PLACE: Chennai** **SIGNATURE OF THE CANDIDATE**

# ACKNOWLEDGEMENT

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

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A.Christy MCA,Ph.D ,**for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

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

# ABSTRACT

With the advancement in deep learning, the consolidation of text classification and image processing has agitated enormous scrutiny in the past years. Captioning images are advanced research mainly in the field of computer vision. Identification of relevant items, their properties, and their connection to images is required for image captioning. This image captioning is a complex and challenging task, several developments are done in this field by researchers. But now, Technology can develop a model that can bring out the closest text or captions explaining an image because of the advancement in deep learning methodologies and the enormous amounts of data that is available. In the beginning, it was contemplated impractical that a computer could characterize an image. Even, NVIDIA is developing an app to assist persons with limited or no vision using image captioning technology. The main aim of this project is to develop a model such that it can predict the closest text or caption to that particular image. This analysis can be done by utilizing deep learning methods as these methods are efficient for handling the complexity involved in image captioning. So, this project intends to build a model that predicts captions or labels from images by using deep learning models. The VGG16 architecture is employed in this research for the prediction and generation of captions for the images and this analysis can be compared with other deep learning frameworks such as LSTM or CNN to observe the performance of the model.

## TABLE OF CONTENTS

## LIST OF FIGURES

# CHAPTER – 1
# INTRODUCTION

## 1.1 GENERAL INFORMATION:

The topic of autonomously producing eloquent words or captions for images has piqued significance in the research area of computer vision along with Natural Language Processing in recent years. Computer vision has made substantial advances in the field of image processing in recent years, such as image categorization and object identification. The development of characterizing the content of an image is referred to as image captioning. Encoder-Decoder architectures are generally used in image captioning methods, with vectors of images as input to the encoder. Image captioning is a key activity that necessitates a semantic comprehension of images as the capacity to generate accurate and precise descriptions for images. Nowadays, Image captioning is a relatively new and rapidly increasing research area. Several new approaches are being launched on daily basis to attain satisfying outcomes in this research field. Image Captioning is a new demanding topic that has newly gained considerable attention. Visual Captioning is a crucial endeavor for bettering the human-computer synergy and gaining a better grasp of the principles that underpin human image characterization. Captioning images may be thought of as a whole process. The process is known as Sequence-to Sequence Process because it translates images, which are considered to be a sequence of pixels, into the sequence of sentences or words. The primary goal of this captioning of images is to create a natural language description for an input image that is given to the model. In this analysis, we introduced a model that provides the characterization of an image using natural language processing. And the features are extracted using neural networks from different architectures such as CNN and LSTM. Finally, text generation was done by utilizing these architectures.

## 1.2 DEEP LEARNING

Deep learning is based on the branch of machine learning, which is a subset of artificial intelligence. Since neural networks imitate the human brain and so deep learning will do. In deep learning, nothing is programmed explicitly. Basically, it is a machine learning class that makes use of numerous nonlinear processing units so as to perform feature extraction as well as transformation. The output from each preceding layer is taken as input by each one of the successive layers.

Deep learning models are capable enough to focus on the accurate features themselves by requiring a little guidance from the programmer and are very helpful in solving out the problem of dimensionality. Deep learning algorithms are used, especially when we have a huge no of inputs and outputs.

Since deep learning has been evolved by the machine learning, which itself is a subset of artificial intelligence and as the idea behind the artificial intelligence is to mimic the human behavior, so same is "the idea of deep learning to build such algorithm that can mimic the brain".

Deep learning is implemented with the help of Neural Networks, and the idea behind the motivation of Neural Network is the biological neurons, which is nothing but a brain cell.

### Types of Deep Learning Networks

1. Feed Forward Neural Network

A feed-forward neural network is none other than an Artificial Neural Network, which ensures that the nodes do not form a cycle. In this kind of neural network, all the perceptron’s are organized within layers, such that the input layer takes the input, and the output layer generates the output. Since the hidden layers do not link with the outside world, it is named as hidden layers. Each of the perceptron’s contained in one single layer is associated with each node in the subsequent layer. It can be concluded that all of the nodes are fully connected. It does not contain any visible or invisible connection between the nodes in the same layer. There are no back-

loops in the feed-forward network. To minimize the prediction error, the backpropagation algorithm can be used to update the weight values.

2. Recurrent Neural Network

Recurrent neural networks are yet another variation of feed-forward networks. Here each of the neurons present in the hidden layers receives an input with a specific delay in time. The Recurrent neural network mainly accesses the preceding info of existing iterations. For example, to guess the succeeding word in any sentence, one must have knowledge about the words that were previously used. It not only processes the inputs but also shares the length as well as weights crossways time. It does not let the size of the model to increase with the increase in the input size. However, the only problem with this recurrent neural network is that it has slow computational speed as well as it does not contemplate any future input for the current state. It has a problem with reminiscing prior information.

3. Convolutional Neural Network

Convolutional Neural Networks are a special kind of neural network mainly used for image classification, clustering of images and object recognition. DNNs enable unsupervised construction of hierarchical image representations. To achieve the best accuracy, deep convolutional neural networks are preferred more than any other neural network.

4. Restricted Boltzmann Machine

RBMs are yet another variant of Boltzmann Machines. Here the neurons present in the input layer and the hidden layer encompasses symmetric connections amid them. However, there is no internal association within the respective layer. But in contrast to RBM, Boltzmann machines do encompass internal connections inside the hidden layer. These restrictions in BMs helps the model to train efficiently.

5. Autoencoders

An autoencoder neural network is another kind of unsupervised machine learning algorithm. Here the number of hidden cells is merely small than that of the input cells. But the number of input cells is equivalent to the number of output cells. An

autoencoder network is trained to display the output similar to the fed input to force AEs to find common patterns and generalize the data. The autoencoders are mainly used for the smaller representation of the input. It helps in the reconstruction of the original data from compressed data. This algorithm is comparatively simple as it only necessitates the output identical to the input.

**Deep learning applications**

Self-Driving Cars

In self-driven cars, it is able to capture the images around it by processing a huge amount of data, and then it will decide which actions should be incorporated to take a left or right or should it stop. So, accordingly, it will decide what actions it should take, which will further reduce the accidents that happen every year.

Voice Controlled Assistance

When we talk about voice control assistance, then Siri is the one thing that comes into our mind. So, you can tell Siri whatever you want it to do it for you, and it will search it for you and display it for you.

Automatic Image Caption Generation

Whatever image that you upload, the algorithm will work in such a way that it will generate caption accordingly. If you say blue colored eye, it will display a blue-colored eye with a caption at the bottom of the image.

## 1.3 PROBLEM STATEMENT

Our analysis is to enhance text for image data by inspecting various Deep learning and NLP techniques to create an authentic model that is adequate of detecting analysis of text for image data. Following that, an analysis was done the in-literature survey establish the enhancing technologies that would aid in the development of the application and the modeling of our architecture for the prediction of captions. The algorithm will take images from test data and predict the text for that particular image. Our Intention was to develop a model that can predict the text for images given to our model.

# CHAPTER – 2
# LITERATURE SURVEY

The authors in this study come up with a hybrid system that utilizes an architecture called Multi-layer Convolutional Neural Network (CNN) to produce vocabulary to characterize an image and utilize other architecture known as Long Short-Term Memory (LSTM) to precisely form substantial sentences utilizing the stemming words. Image captioning is a key activity that necessitates a semantic comprehension of images as the capacity to generate accurate and precise descriptions for images. Generally, producing comprehensive and instinctive image characterization has a wide range of applications including captions for news images, characterization of images in the field of medical analysis, text-based image retrieval, Instructions for blind people, and communication between humans and robots. The concept is established on the detection of objects and actions that must be taken in the image given to the model. Vectors derived from images collected for object detection are utilized in the most effective methods. It is a difficult issue to unquestionably generate a characterization for an image by utilizing the approaches of natural language processing. The observation from the survey, it reveals that this analysis was also done by utilizing transformers which extends this object relation transformer technique by decidedly adding content about the dimensional connection among input recognized items by utilizing geometric attention. From the previous papers, it is observed that CNN is utilized to interpret visual information and locate objects in an image, whereas RNN or LSTM is utilized to generate descriptions for images. The study of analytical methods and processes that are applicable for processing techniques such as Machine learning, Natural language processing, and even deep learning. Captioning images is advanced research mainly in the field of computer vision. The main vision of this project is to create a model such that it can forecast the closest text or caption to that particular image.

## 2.1 Inferences from Literature Survey

**Paper 1:** Image Captioning - A Deep Learning Approach

**Conference:** International Journal of Physics: Conference Series

**Year:** 2021

**Basic Theory**: In this analysis, authors proposed the usage of CNN to generate text describes the images.

**Methodology:**

Here, Analysis was done by using 2 algorithms.

- CNN was used to generate vocabulary for images.
- Then, they have used LSTM to accurately structure meaningful sentence using the generated words from CNN.

**Weakness:**

In this paper, we have observed that they didn't mention effectiveness of algorithm they have used for prediction of captions.

**Paper 2**: Image Captioning Transforming objects into words

**Conference:** International Journal of Applied Sciences

**Year:** 2020

**Basic Theory**: In this work, they have introduced an object relation transformer, that helps to find relation between the images

**Methodology:**

- They found relation between input objects through geometric attention by using object related transformer.
- They demonstrated usefulness of geometric attention and by this they have improved captions or text for images.

**Weakness:**

Here, they have used only transformers for analysis. It’s better to compare and try it with another algorithm like LSTM, RNN's.

**Paper 3:** An Integrative Review of Image Captioning Research

**Conference:** International Conference on Advances in Neural Information Processing Systems

**Year:** 2019

**Basic Theory:** This paper investigates and analyses the related research of image captioning.

**Methodology:**

- This paper introduces existing image captioning methods and analyzed their principles.
- Although, they are existing, here they have improved accuracy or predictions to certain extent.

**Weakness:**

Since it is a review paper, there are no weaknesses in the paper as they have mentioned all techniques in detail.

**Paper 4:** Text Augmentation using BERT for Image Captioning

**Conference:** International Conference on Electronic Information Technology and Computer Engineering (EITCE 2018)

**Year:** 2018

**Basic Theory:** In this paper Analysis methods include augmentation with synonyms as baseline model called BERT (Bidirectional Encoder Representation from transformers.

**Methodology:**

- Analysis was done using BERT
- Here, synonyms augmentation was done to obtain new caption based on existing one.

**Weakness:**

They have mentioned in conclusion that they used BLEU algorithm but they didn't mention that technique in paper.

**Paper 5:** Image Captioning based on Deep-Neural Networks

**Conference:** International Journal of Applied Engineering Research

Year: 2018

**Basic Theory:** In this analysis was done by utilizing image captioning methods using deep learning frame works such as CNN and RNN frame work.

**Methodology:**

- They have compared model with efficient frame works like CNN and RNN.
- They have used evaluation metrices such as METEOR to compare results Meteor metric was primarily used in the Machine Translation literature. Checkout our article giving an overview of techniques for machine translation to get a better understanding of the translation application.

**Weakness:**

This can be further extent to application of image visualization and visual question answering (VOA) that has important academic and practical application value.

## 2.2 Open problems in Existing System

Despite significant progress in image captioning, there are still some open problems in existing systems. Here are a few examples:

- Improving the Quality of Captions: Despite the significant advances in image captioning, the quality of captions generated by existing models still varies greatly. Improving the quality of captions is an ongoing challenge, as it requires the development of more effective language models that can better understand the nuances of human language.
- Generating diverse and creative captions: Most image captioning systems generate relatively simple and descriptive captions that describe the content of the image. However, generating diverse and creative captions that go beyond simple descriptions remains a challenge.
- Handling rare or unseen objects and scenes: Image captioning systems often struggle with rare or unseen objects or scenes that are not well-represented in the training data. Improving the ability of these systems to handle novel or rare concepts would improve their usefulness in real-world applications.
- Incorporating commonsense knowledge: Image captioning systems typically lack common sense knowledge, which can lead to captions that are logically

inconsistent or implausible. Incorporating commonsense knowledge could improve the quality and coherence of generated captions.

- Dealing with multiple objects and relationships: Image captioning systems often struggle with describing relationships between multiple objects in an image, particularly when they are not visually salient. Improving the ability of these systems to handle multiple objects and relationships would enable more detailed and nuanced captions.
- Multimodal understanding: Image captioning systems often rely solely on visual information to generate captions, but incorporating other modalities such as audio or text could improve their ability to generate more accurate and informative captions.
- Improving efficiency: Many image captioning models are computationally expensive and require significant resources to train and run. Developing more efficient models that can be trained and run on standard hardware would make image captioning more accessible and scalable.
- Context and Common-Sense Understanding: Image captioning models sometimes fail to capture the context and common sense of the scene, leading to inaccurate or nonsensical captions. Developing models that can better understand the context and common sense of the scene to generate more meaningful captions is another open problem.
- Evaluation Metrics: There is no consensus on the evaluation metrics used to measure the performance of image captioning models. Developing more effective evaluation metrics that better reflect the quality of captions generated by the models is an ongoing challenge.

Overall, image captioning is a complex problem that requires the integration of multiple technologies such as computer vision, natural language processing, and deep learning. Addressing these open problems will require continued research and innovation to improve the quality and accuracy of image captioning systems.

# CHAPTER 3
# REQUIREMENT ANALYSIS

## 3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

Feasibility studies of image captioning have shown that it is a promising and viable approach for automatically generating captions for images. Here are some key findings from feasibility studies:

- Image captioning is feasible for a wide range of applications: Image captioning has been applied to a variety of applications, including image search, image retrieval, and assisting visually impaired individuals. These studies have demonstrated the feasibility and effectiveness of image captioning for these applications.
- Image captioning can achieve human-level performance: In some studies, image captioning systems have been shown to achieve performance that is comparable to or even better than human-generated captions. For example, the top-performing systems on benchmark datasets such as MS COCO have achieved scores that are close to or exceed human performance on some evaluation metrics.
- Image captioning can be improved with better training data: Feasibility studies have shown that image captioning systems can be improved with better training data. This includes larger and more diverse datasets, as well as datasets that include captions that go beyond simple descriptions of the image content.
- Image captioning can benefit from multimodal information: Some studies have shown that incorporating other modalities such as audio or text can improve the quality and accuracy of generated captions. For example, using audio descriptions of images in addition to visual information has been shown to improve the quality of generated captions for visually impaired individuals.
- Image captioning has potential for real-world applications: Feasibility studies have demonstrated the potential for image captioning to be used in real-world applications such as assisting visually impaired individuals and improving image search and retrieval. These studies suggest that image captioning has

significant practical value and can have a positive impact on people's lives. Overall, feasibility studies of image captioning have shown that it is a promising approach with significant potential for a wide range of applications. However, there are still challenges to be addressed, such as generating diverse and creative captions and handling rare or unseen concepts.

### 3.2 DATA SET

Due of each order's unique personality, the records each contain something unique. This report was carried out using a variety of information sources, including data generation, text capture information, picture information, pixel information photos, Resnet embeddings, and so on. This analysis was done using a sizable image collection made available via the cloud of the Wikimedia Foundation. Several languages are used to analyses these statistics. There are 79 or so languages. Using pre-processing techniques and photograph assessment processing, the statistics are assessed in natural language and processed for both text and images.

| language | page URL | Page Title | Section Title |
|---|---|---|---|
| en | https://en.wikipedia.org/wiki/Silverspoon | Silver spoon | Historical uses |
| Zh-TW | https://zh.wikipedia.org/wiki/97/kita-sendai-staion | 北仙台站 | JR東日本 |

Table I. Training Data

| Mime-Type | Caption reference | height | width | Is main page |
|---|---|---|---|---|
| Image/jpeg | Two silver-giltstrainerspoons on the table | 1194 | 2139 | false |
| Image/jpeg | 月台 | 2112 | 2816 | true |

Table II. Training Data

| Id | Image URL |
|---|---|
| 0 | https://uplaod.wikimedia.org/wikipedia/scotsGaelicspeakers2011cesnus.png |
| 1 | https://upload.wikimedia.org/wikipedia/ThermopylaeAncientcoastline.jpg |

Table III. Test Data

| Captions |
|---|
| Albert Pike [SEP] Albert Pike |
| Anna Blount [SEP] Blount and her young daughter, in 1911 |

Table IV. Caption Data

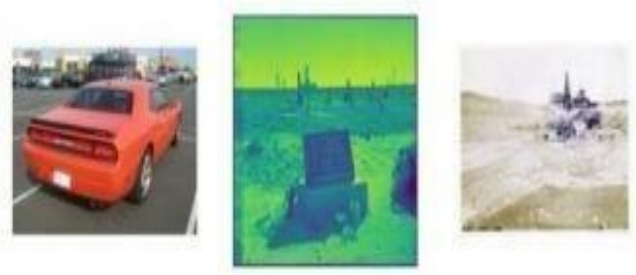


Figure 3.1. Sample images Data set

### 3.3 SOFTWARE AND HARDWARE REQUIREMENTS SPECIFICATION DOCUMENT

Software Requirements

- Operating System : Windows
- Tool : Google Collab

Hardware Requirements

- Processor : Pentium IV/III
- Hard disk : minimum 80 GB
- RAM : minimum 2 GB

# CHAPTER 4
# DESCRIPTION OF PROPOSED SYSTEM

## 4.1 SELECTED METHODOLOGY / PROCESS MODEL

The analysis and perceptions of data were done in two phases. In Phase-I, Deep learning models such as LSTM, VGG16 are utilized for training data and to predict captions for the test data. And in Phase-II, this was developed for further improvements and this project got deployed using Heroku and VS Code. Deep Learning is one of the most quickly evolving and explored fields of analysis that is permeating day-to-day life. It is commonly the development of neural networks by utilizing high-end contemporary frameworks. It enables the advancement of training and the application of considerably bigger neural networks than heretofore. Researchers have recommended hundreds of different kinds of particular neural networks as improvements or changes to the current models. CNN is most well-known. In this analysis, the model is developed such that text processing is done by utilizing Natural Language Processing and pre trained models such as Xception are used to extract features from the images, and then they are fed further into LSTM, for generating captions for the test data. By utilizing LSTM, captions for test data are predicted. After all the analysis Results are compared by using the BLEU score, and this project got deployed by utilizing Heroku.

## 4.2 ARCHITECTURE/OVERALL DESIGN OF PROPOSED SYSTEM

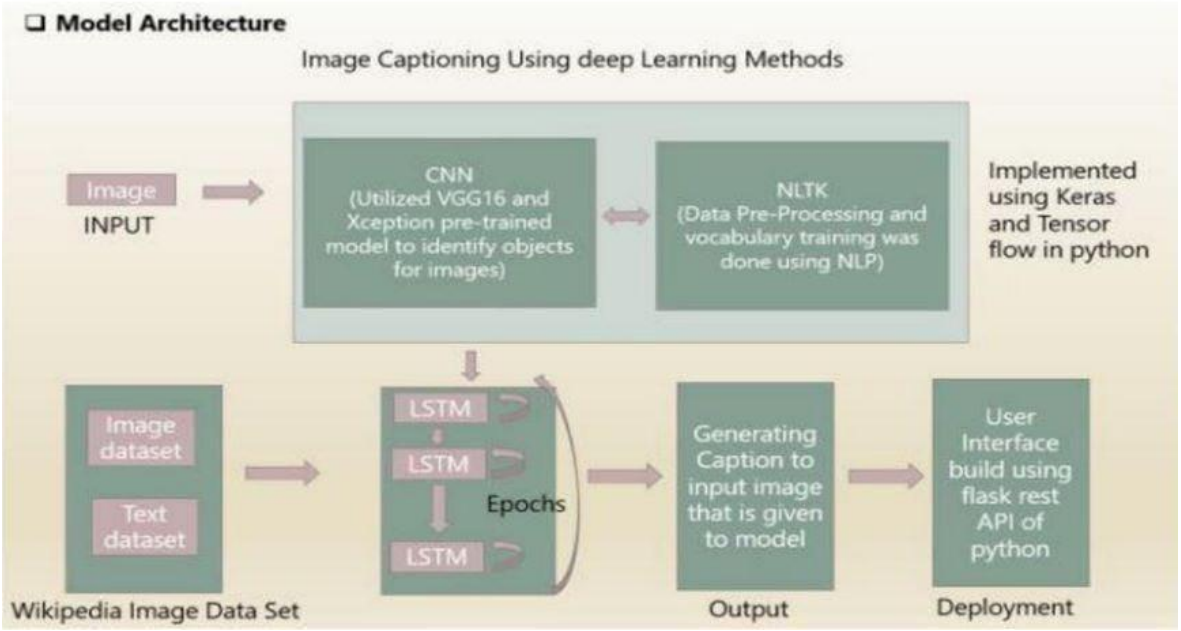



Figure 4.1 **-** Model Architecture of our Project

## 4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF PROPOSED MODEL/ SYSTEM

One of the dominant steps while interacting with data is exploring data by utilizing data pre-processing techniques. This is one of the most important steps to process the data before giving it to any model for training. It was difficult to choose the data for training the model as it has huge data for test and training such as image data, text data and captions data. In training data each image has 5 captions that are closest to the image. Each 5 represents the image in a unique way. There was a challenge in deciding which data to utilize to train the model for getting efficient results as there is more than one data set to accomplish our aim. Here, this pre-processing analysis was done by using different kinds of plots such as tree-maps using sqaurify, bar graphs and image processing was done by scaling the images and flattening them and text processing was done by using NLP (Natural Language Processing) techniques such as Lowering the text, stop words

removal, tokenizing the text, and so on.

**Tree-Map:** This is used to predict data by nesting rectangles of varied sizes together. The size of Each rectangle is comparable to the quantity of data it performs as a percentage of the total. This is a visualization technique used to divide parameters into subparts. It can assist in seeing how individuals' values combine to form an entire data. Data is frequently analyzed in the form of chunks of input data, with distinct parameters with respect to the various approaches of the data. There are various kinds of attributes in tree map such as values, text info, path bar, branch values etc.

**Tokenization:** The initial stage in the Natural Language Processing pipeline is tokenization. It has a significant impact on the remainder of the architecture. This can be accomplished in words or sentences. This is the most frequently utilized technique in any NLP Problem statement. It divides sentences into a piece of words. Tokenizer divides the data and text into characterization bits that may be thought of as separate parts.

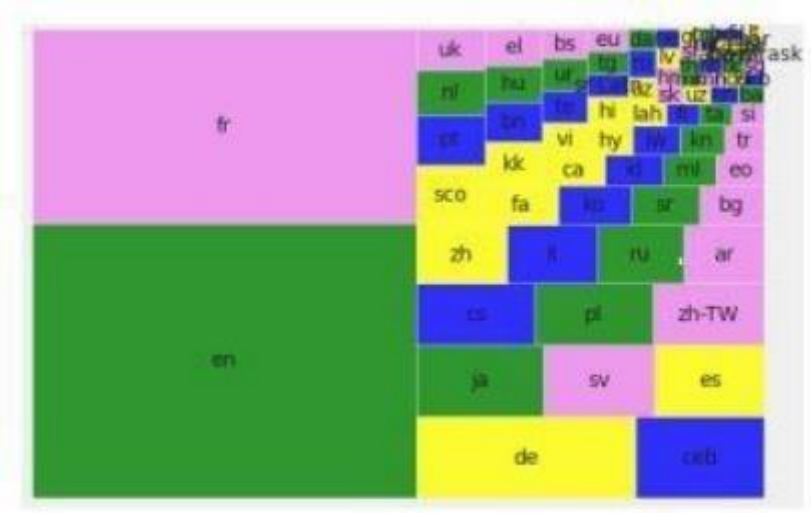



Figure 4.2 - Tree Map

The above figure interprets that out of 79 languages English and French are the most used data.

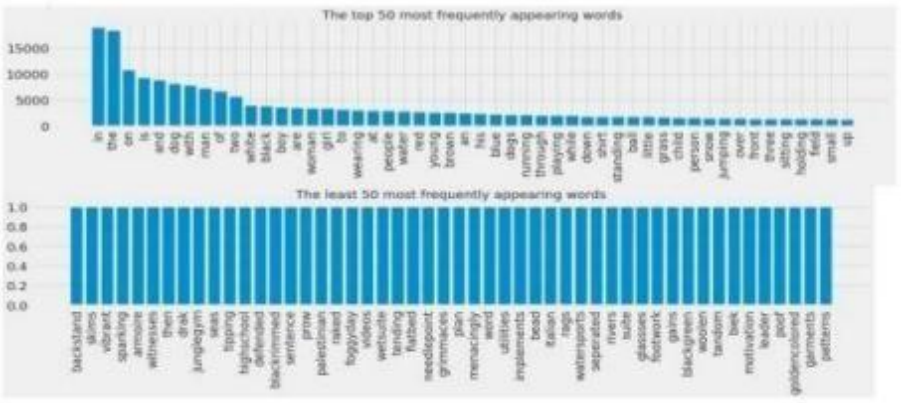



Figure 4.3 – Top 50 most frequent words

From fig 4.3, Results interprets that they are the top most 50 frequent words of data after doing text pre-processing techniques such as stemming, tokenization.

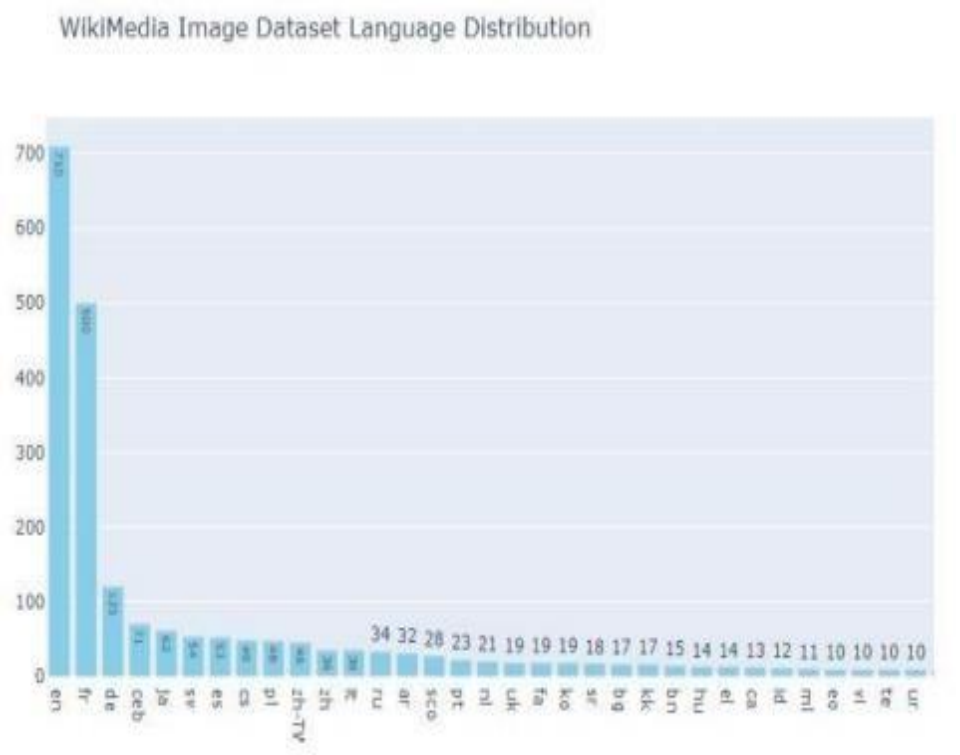



Figure 4.4- A bar graph showing Language Distribution

Fig 4.4, shows the language distribution of the data set. And it is observed that English is most utilized language.


Figure 4.5- Word Cloud for Page Title

These are some data analysis techniques concerning text training data and text data.

Apart from the text and training data processing, analysis was done even on image data such as re-sizing, flattening the image, and scaling it to pixels (Normalization). These processing techniques are done on images because we cannot just feed an image directly to a model without doing pre-processing on images.

**Flattening:** This is a method utilized to disciple multidimensional array to 1-D array. It's usually utilized in training deep learning tasks while giving input 1 -D array for classification models. This is done because multidimensional arrays utilize more memory arrays1-D array. To save time and complexity while dealing with large data sets, flattening is one of the processing should be performed on images before training the model.

**Normalization (Scaling to pixels):** This is a technique in image analysis that adjusts the pixel's range. The intention behind this is to adjust an image to its intensity values so that analysis will be efficient. It's the function that will result in the normalization of the input image that can be RGB or Grey scale.


Figure 4.6 - Image analysis corresponding to the Page title

From Figure 4.6, it is anticipated that results are corresponding to page title after performing image processing techniques on image data.

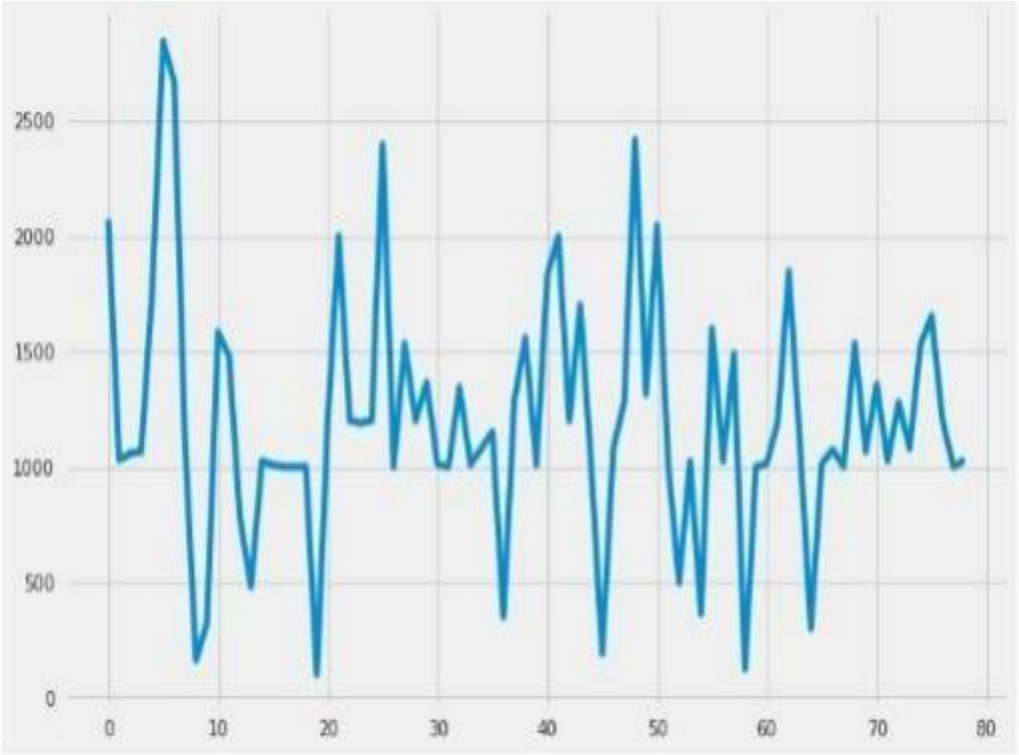



Figure 4.7 - Graph related to Image dimensions height

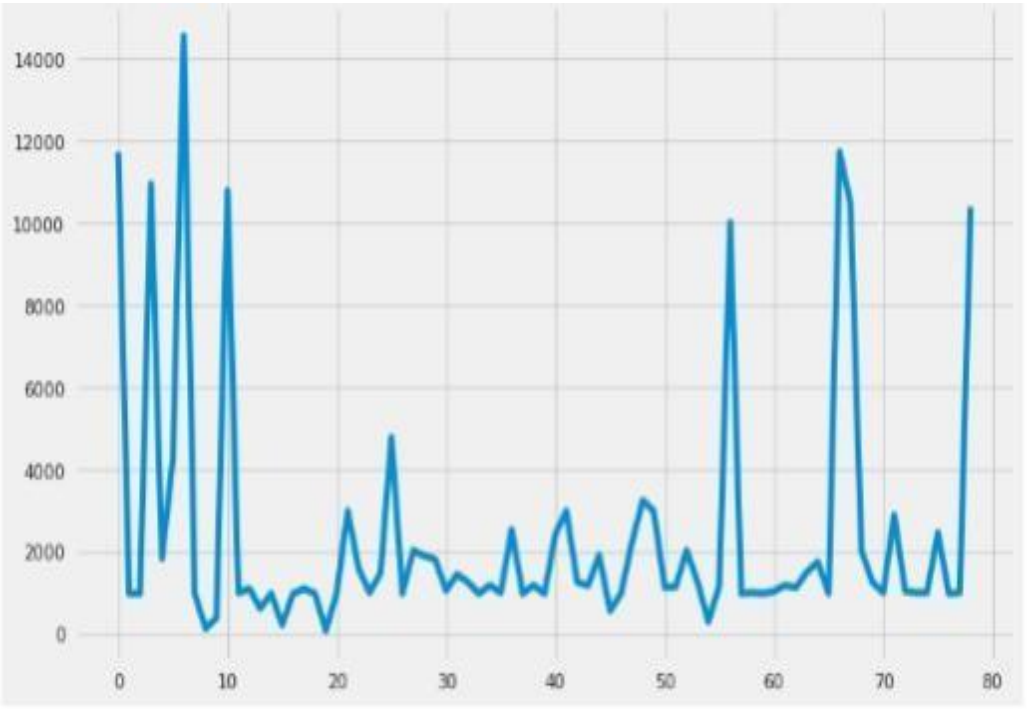



Figure 4.8 - Graph related to Image dimensions width

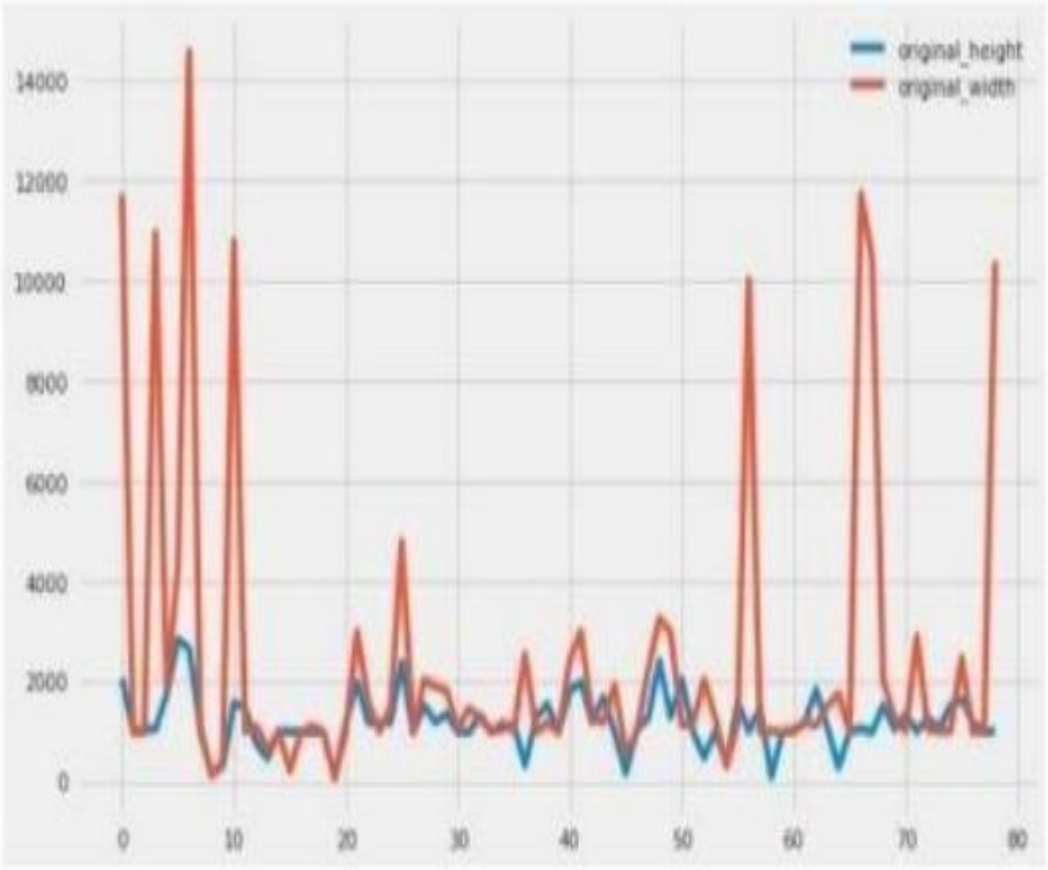



Figure 4.9 - Graph related to Image dimensions

This interprets the dimensions of images corresponding to height and width.

### 4.4 PROJECT MANAGEMENT PLAN

A project management plan for an image captioning project using CNN and LSTM models can be:

- Define the objectives, scope, and goals of the project. Develop a project plan that includes a project scope statement, schedule, and risk management plan.
- Develop and train the CNN and LSTM models using image and caption data. Optimize the models for accuracy and efficiency.
- Test the models using a validation dataset. Develop a user interface for the image captioning system. Monitor and control the project progress by tracking the project schedule. Identify and address any issues or risks that

may affect the project.

- Revise the project plan as necessary to ensure that the project stays on track. Complete the project and evaluate the project's success and identify areas for improvement. Document the lessons learned and best practices for future reference.

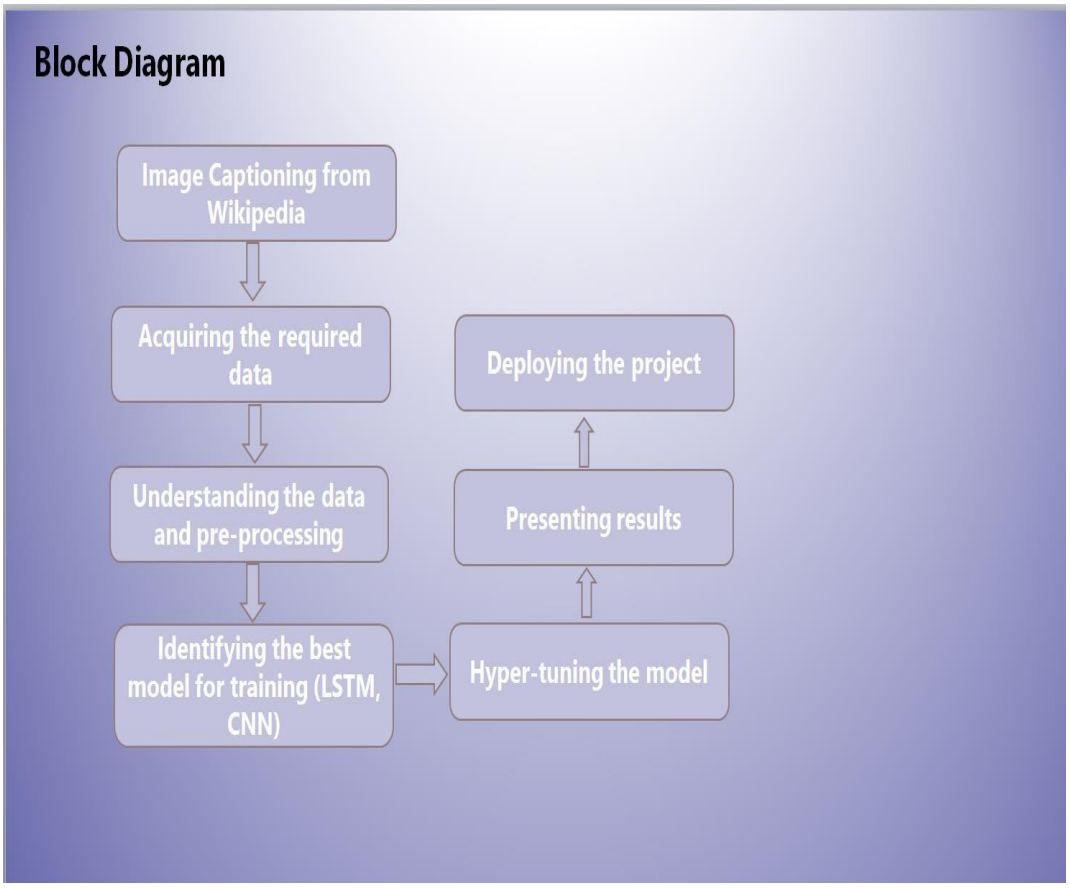



Figure 4.10 – Project Management Plan

An effective project management plan for an image captioning project using CNN and LSTM models should focus on quality, accuracy, and usability of the image captioning system.

# CHAPTER – 5
# IMPLEMENTATION DETAILS

## 5.1 DEVELOPMENT AND DEPLOYMENT SETUP

The analysis and perceptions of data were done in two phases. In Phase-I, Deep learning models such as LSTM, VGG16 are utilized for training data and to predict captions for the test data. And in Phase-II, this was developed for further improvements and this project got deployed using Heroku and VS Code. Deep Learning is one of the most quickly evolving and explored fields of analysis that is permeating day-to-day life. It is commonly the development of neural networks by utilizing high-end contemporary frameworks. It enables the advancement of training and the application of considerably bigger neural networks than heretofore. Researchers have recommended hundreds of different kinds of particular neural networks as improvements or changes to the current models. CNN is most well-known. In this analysis, the model is developed such that text processing is done by utilizing Natural Language Processing and pretrained models such as Xception are used to extract features from the images, and then they are fed further into LSTM, for generating captions for the test data. By utilizing LSTM, captions for test data are predicted. After all the analysis Results are compared by using the BLEU score

## 5.2: ALGORTHIMS

**CNN**: A convolutional neural network (CNN or convnet) is a subset of machine learning. It is one of the various types of artificial neural networks which are used for different applications and data types. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data.

There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. This makes them highly suitable for computer vision (CV) tasks and for applications where object

recognition is vital, such as self-driving cars and facial recognition.

Artificial neural networks (ANNs) are a core element of deep learning algorithms. One type of an ANN is a recurrent neural network (RNN) that uses sequential or time series data as input. It is suitable for applications involving natural language processing (NLP), language translation, speech recognition and image captioning.

The CNN is another type of neural network that can uncover key information in both time series and image data. For this reason, it is highly valuable for image-related tasks, such as image recognition, object classification and pattern recognition. To identify patterns within an image, a CNN leverages principles from linear algebra, such as matrix multiplication. CNNs can also classify audio and signal data.

A CNN's architecture is analogous to the connectivity pattern of the human brain. Just like the brain consists of billions of neurons, CNNs also have neurons arranged in a specific way. In fact, a CNN's neurons are arranged like the brain's frontal lobe, the area responsible for processing visual stimuli. This arrangement ensures that the entire visual field is covered, thus avoiding the piecemeal image processing problem of traditional neural networks, which must be fed images in reduced-resolution pieces. Compared to the older networks, a CNN delivers better performance with image inputs, and also with speech or audio signal inputs.

A deep learning CNN consists of three layers: a convolutional layer, a pooling layer and a fully connected (FC) layer. The convolutional layer is the first layer while the FC layer is the last.

This stands for Convolution Neural Network where Image data is mapped to a target variable. They have proven to be successful in that they are now the techniques of choice for any form of prediction issue utilizing data as an input to the model. CNN is a multi-layered feed-forward neural network that is built by layering several hidden layers on top of one another in a certain sequence. These layers are frequently outlawed by several layers in CNN, while activation layers are usually enhanced by layers in the convolutional network.

From the convolutional layer to the FC layer, the complexity of the CNN increases. It is this increasing complexity that allows the CNN to successively identify larger portions and more complex features of an image until it finally identifies the object in its entirety.

Convolutional layer. The majority of computations happen in the convolutional layer, which is the core building block of a CNN. A second convolutional layer can follow the initial convolutional layer. The process of convolution involves a kernel or filter inside this layer moving across the receptive fields of the image, checking if a feature is present in the image.

Over multiple iterations, the kernel sweeps over the entire image. After each iteration a dot product is calculated between the input pixels and the filter. The final output from the series of dots is known as a feature map or convolved feature. Ultimately, the image is converted into numerical values in this layer, which allows the CNN to interpret the image and extract relevant patterns from it.

Pooling layer. Like the convolutional layer, the pooling layer also sweeps a kernel or filter across the input image. But unlike the convolutional layer, the pooling layer reduces the number of parameters in the input and also results in some information loss. On the positive side, this layer reduces complexity and improves the efficiency of the CNN.

Fully connected layer. The FC layer is where image classification happens in the CNN based on the features extracted in the previous layers. Here, fully connected means that all the inputs or nodes from one layer are connected to every activation unit or node of the next layer.

All the layers in the CNN are not fully connected because it would result in an unnecessarily dense network. It also would increase losses and affect the output quality, and it would be computationally expensive.

A CNN can have multiple layers, each of which learns to detect the different features of an input image. A filter or kernel is applied to each image to produce an output that gets progressively better and more detailed after each layer. In the lower layers,

the filters can start as simple features.

At each successive layer, the filters increase in complexity to check and identify features that uniquely represent the input object. Thus, the output of each convolved image -- the partially recognized image after each layer -- becomes the input for the next layer. In the last layer, which is an FC layer, the CNN recognizes the image or the object it represents.

With convolution, the input image goes through a set of these filters. As each filter activates certain features from the image, it does its work and passes on its output to the filter in the next layer. Each layer learns to identify different features and the operations end up being repeated for dozens, hundreds or even thousands of layers. Finally, all the image data progressing through the CNN's multiple layers allow the CNN to identify the entire object.

Deep learning is a subset of machine learning that uses neural networks with at least three layers. Compared to a network with just one layer, a network with multiple layers can deliver more accurate results. Both RNNs and CNNs are used in deep learning, depending on the application.

For image recognition, image classification and computer vision (CV) applications, CNNs are particularly useful because they provide highly accurate results, especially when a lot of data is involved. The CNN also learns the object's features in successive iterations as the object data moves through the CNN's many layers. This direct (and deep) learning eliminates the need for manual feature extraction (feature engineering).

CNNs can be retrained for new recognition tasks and built on preexisting networks. These advantages open up new opportunities to use CNNs for real-world applications without increasing computational complexities or costs.

As seen earlier, CNNs are more computationally efficient than regular NNs since they use parameter sharing. The models are easy to deploy and can run on any device, even smartphones.

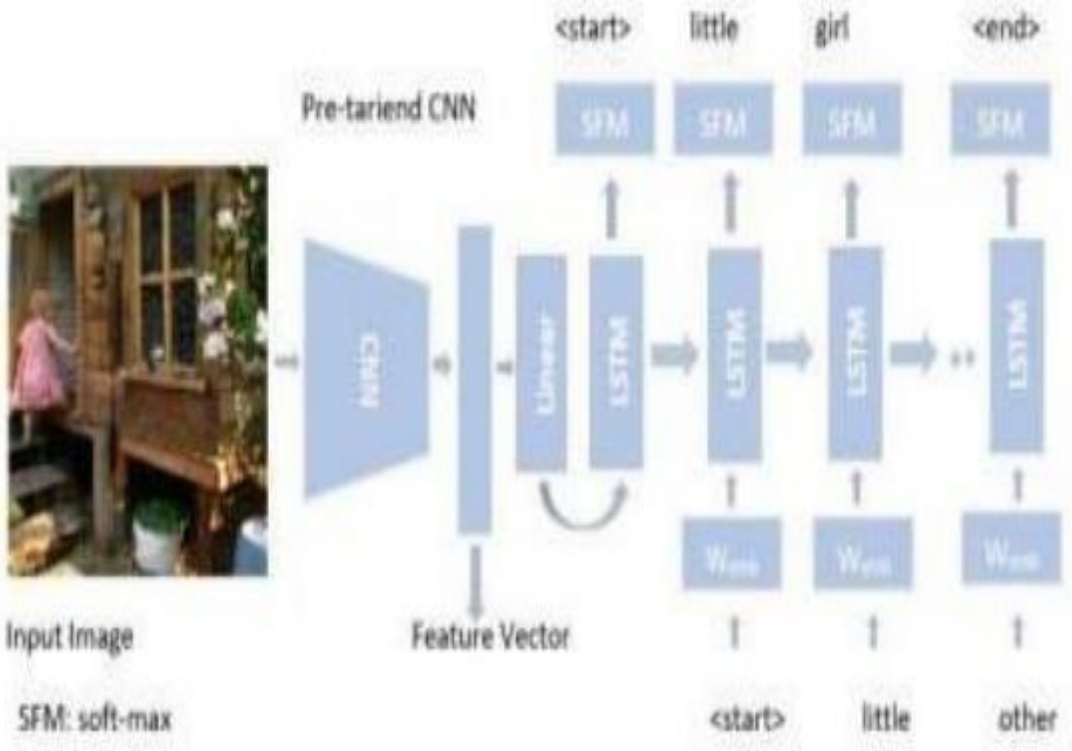



Figure 5.1 - Overview of Architecture

Figure 5.1, it shows the Architecture of the model how each layer contributes to the model.


Figure 5.2 - Training phase results

Prediction of captions for images was analyzed in three phases in this model architecture Feature Extraction for images, Sequence Processor and Decoder.

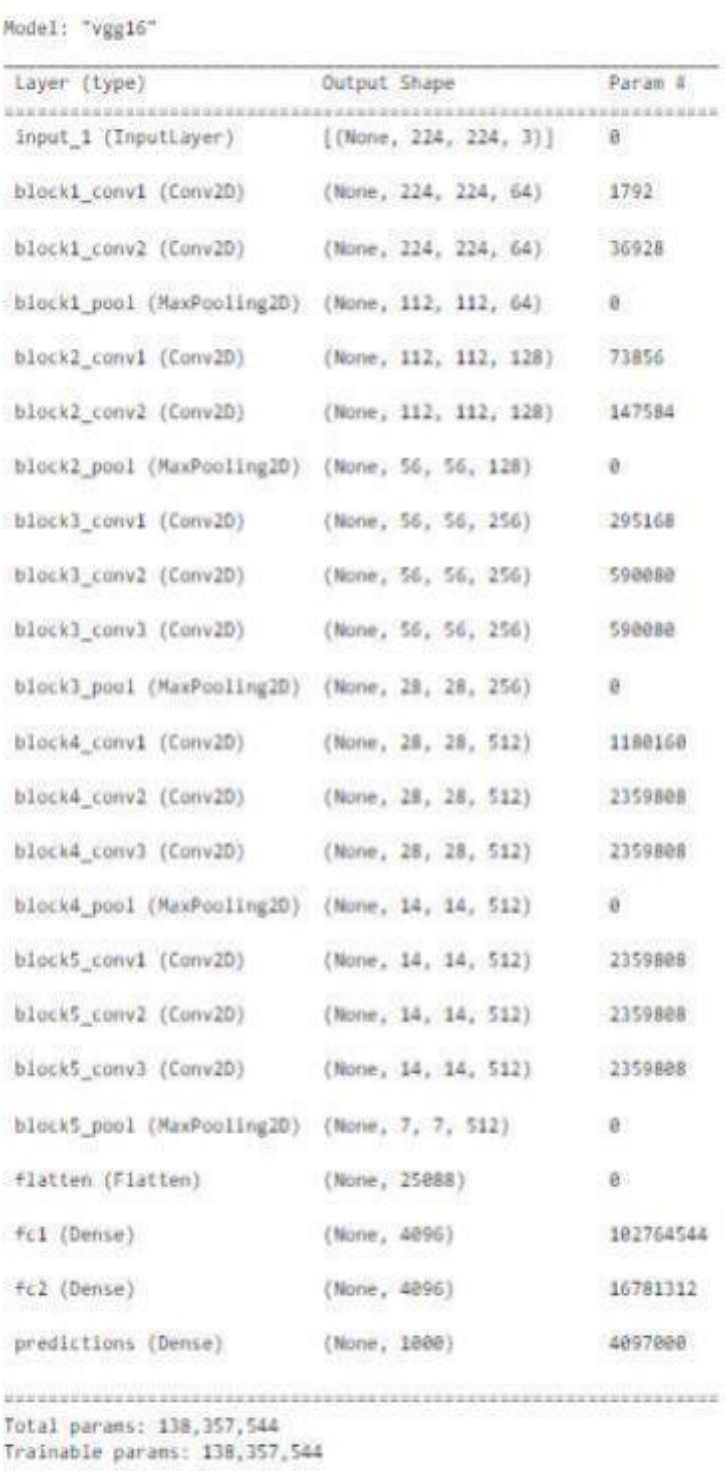
**Feature Extraction:** This is done by using pre-trained models such as Xception and VGG16. This is known as transfer learning.

**Sequence Processor:** This acts as an embedding layer, by interacting with the text data. This contains disciples for extracting text's needed characteristics.

**Decoder:** This is the concluding phase that utilized advancement techniques to merge image extractor with sequence processor, which is passed to neuron and ultimately for final output phase.

**Xception:** This is a pre-trained model that has 36 layers that permit it to learn quickly. These are given to the LSTM layer after being developed by a dense layer to provide 2048 vector enhancement of the image. This model is pre-trained on huge data and extracts features from this analysis to utilize them with current problem statement analysis. This has trained on image net data that has 1000 various kinds of images for categorization and this model can be loaded directly from applications of Keras. As this model is purely trained on image net data, we have done changes when anticipating this model. Xception is a type of convolutional neural network (CNN) that was introduced by François Chollet in 2016. It is a variant of the Inception architecture, which is known for its effectiveness in image classification tasks. The name "Xception" is derived from "Extreme Inception", as it uses an "extreme" version of the Inception module. In image captioning, Xception is often used as a feature extractor to extract image features from the input image. The extracted features can then be fed into a recurrent neural network (RNN) such as LSTM to generate a caption for the image. Xception has been shown to be effective in improving the accuracy of image captioning models compared to other CNN architectures such as VGG16 and ResNet.

Overall, Xception is a powerful CNN architecture that has been shown to be effective in a wide range of computer vision tasks, including image captioning.



```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
flatten (Flatten)            (None, 25088)             0
fc1 (Dense)                  (None, 4096)              102764544
fc2 (Dense)                  (None, 4096)              16781312
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
```

Figure 5.3 - Results from VGG16

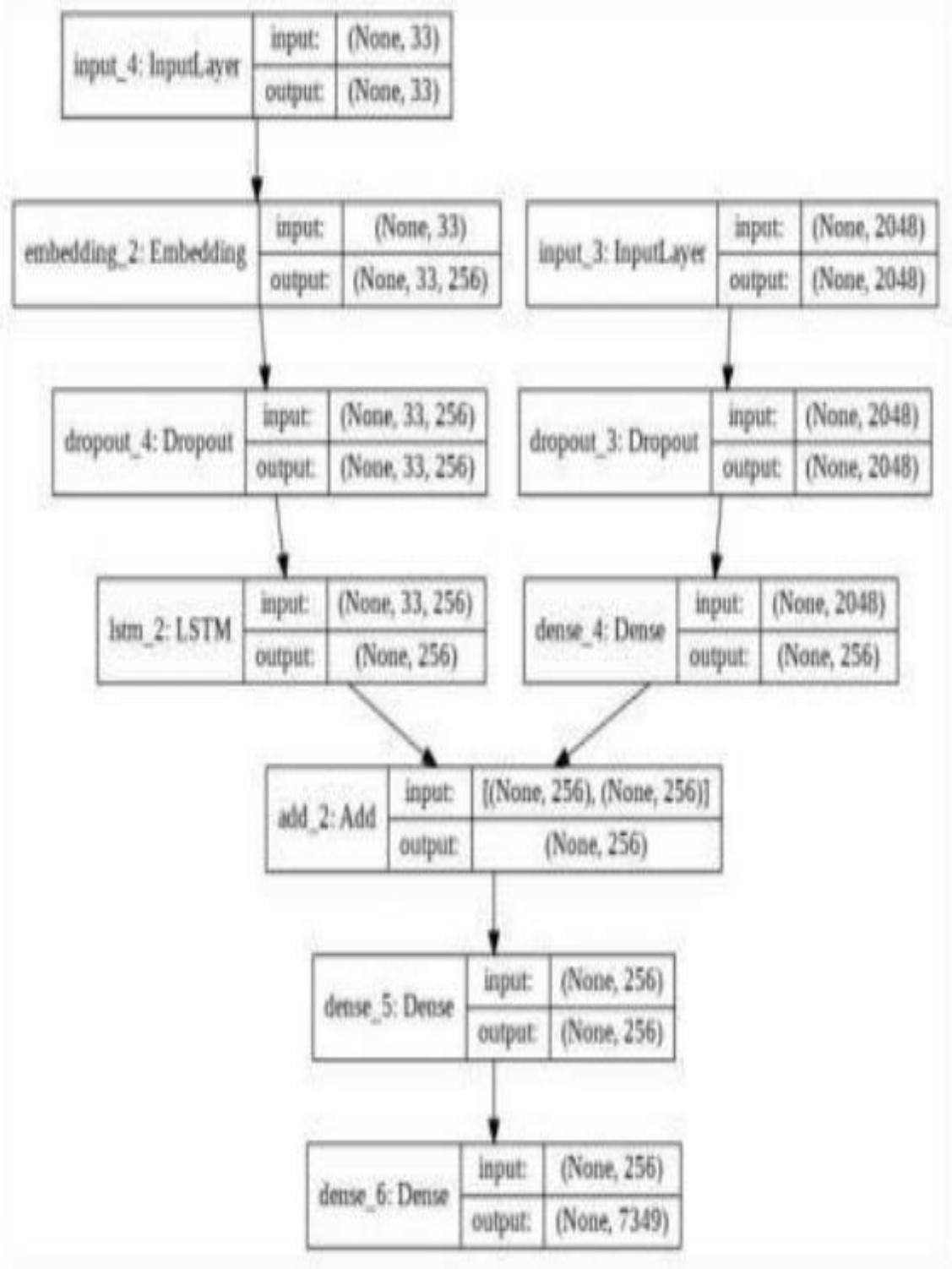



Figure 5.4 - Model Architecture of Xception Model

Here, From Figure 5.4, input 3 is the input of the Pretrained model.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.5 - Further Architecture of our Model

**LSTM:** LSTM is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. LSTM was designed by Hochreiter & Schmid Huber. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying based on time-series data. Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is specifically designed to handle sequential data, such as time series, speech, and text. LSTM networks can learn long-term dependencies in sequential data, which makes them well suited for tasks such as language translation, speech recognition, and time series forecasting.

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an extended period. The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell. The input gate controls what information is added to the memory cell. The forget gate controls what information is removed from the memory cell. And the output gate controls what information is output from the memory cell. This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies. LSTM stands for Long Short-Term Memory. Here, the CNN model is utilized to get features from images with the help of VGG16-Xception models, which are then input to the architecture of LSTM, which generates the captions for data. This is known as the CNN-LSTM model is primarily developed for the prediction issues involving inputs like images and video captures.

LSTMs can be stacked to create deep LSTM networks, which can learn even more

Fig

Figure 5.6 – Structure of LSTM

complex patterns in sequential data. LSTMs can also be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs) for image and video analysis.

LSTM has a chain structure that contains four neural networks and different memory blocks called cells. Information is retained by the cells and the memory manipulations are done by the gates. There are three gates –

1. **Forget Gate:** The information that is no longer useful in the cell state is removed with the forget gate. Two inputs x_t (input at the particular time) and h_t-1 (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state, the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.

Figure 5.7 – Forget Gate

2. **Input gate:** The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered like the forget gate using inputs h_t-1 and x_t. Then, a vector is created using tanh function that gives an output from -1 to +1, which contains all the possible values from h_t-1 and x_t. At last, the values of the vector and the regulated values are multiplied to obtain the useful information.



Figure 5.8 – Input Gate

3. **Output gate:** The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying tanh function on the cell. Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs h_t-1 and x_t. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the

next cell.



Figure 5.9 – Output Gate

These extracted features from input data by using CNN layers are paired with LSTM for forecasting vectors in further development. These models have a lot of promise and are developing being employed for more complex analysis like text categorization.


Figure 5.10 - Model Architecture of CNN-LSTM

Figure 5.10 interprets the overview architecture of the CNN LSTM model that is utilized for feature extraction and getting captions.

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 30)]         0           []

embedding (Embedding)           (None, 30, 64)       286464      ['input_3[0][0]']

input_2 (InputLayer)            [(None, 1000)]       0           []

CaptionFeature (LSTM)           (None, 256)          328704      ['embedding[0][0]']

ImageFeature (Dense)            (None, 256)          256256      ['input_2[0][0]']

add (Add)                       (None, 256)          0           ['CaptionFeature[0][0]',
                                                                  'ImageFeature[0][0]']

dense (Dense)                   (None, 256)          65792       ['add[0][0]']

dense_1 (Dense)                 (None, 4476)         1150332     ['dense[0][0]']

==================================================================================================
Total params: 2,087,548
Trainable params: 2,087,548
Non-trainable params: 0
```

Figure 5.11 - Results from LSTM

Features of images to be anticipated are developed by utilizing this VGG16-Xception Architecture. The weights of VGG16 are frozen when we develop the LSTM model. With this analysis, our model got trained on many images and text data and was given to predict test data captions.


Figure 5.12 - Testing done on images and predicted caption

So finally, after doing this analysis on the training and testing phase, these are compared using an evaluation metric called BLEU score.

**BLUE Score:** To figure out a caption, it is correlated to its captions. It stands for Bilingual Evaluation Understudy. This is an algorithm that is utilized for examining 25 the quality of the machine-translated text.


Figure 5.13 - Image caption predictions compared with BLEU score.

To improve the BLEU score we have developed our model further to get efficient. After development, the test data predictions are shown below from Table 4.1 and Table 4.2


Figure 5.14 - Training images

```
12.36% is done..
24.72% is done..
37.08% is done..
49.44% is done..
61.80% is done..
74.17% is done..
86.53% is done..
98.89% is done..
```

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_3 (InputLayer) | [(None, 30)] | 0 | [] |
| embedding (Embedding) | (None, 30, 64) | 286464 | ['input_3[0][0]'] |
| input_2 (InputLayer) | [(None, 1000)] | 0 | [] |
| CaptionFeature (LSTM) | (None, 256) | 328704 | ['embedding[0][0]'] |
| ImageFeature (Dense) | (None, 256) | 256256 | ['input_2[0][0]'] |
| add (Add) | (None, 256) | 0 | ['CaptionFeature[0][0]', 'ImageFeature[0][0]'] |
| dense (Dense) | (None, 256) | 65792 | ['add[0][0]'] |
| dense_1 (Dense) | (None, 4476) | 1150332 | ['dense[0][0]'] |

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5
102973440/102967424 [==============================] - 1s 0us/step
102981632/102967424 [==============================] - 1s 0us/step
```

Figure 5.15 - Training on test data

| |
|---|
| image-URL: Scots Gaelic speakers in 2011 census.png [SEP] |
| closest captions: |
| Sao Vicente do Penso [SEP] Escudo |
| Jocs Panamericans de 2011 [SEP] Guadalajara |
| Bois de pins et chenes de Madren [SEP] carte |
| Standard time in the United States [SEP] 1913 |
| Hebrides[SEP]Geographic distribution of speakers(2011) |

Table 5.1 – Test Data Predictions Sample

| |
|---|
| image-URL: Thermopylae ancient coastline large.png [SEP] |
| closest captions: |
| Oberlin College [SEP] logo |
| Departmento de Flores [SEP] Escudo |
| Sekolah Menengh Kebangssan Selandar [sep] cny 4 |
| Academia de Ciencias de Cuba [SEP] Sede. |
| Geothermal areas in New Zealand [SEP] Geyser Flat |

Table 5.2 - Test Data Predictions Sample

From Table 5.1 and table 5.2, these are the inferred results after training and predictions of captions for test data

| Id | Caption title predicted |
|---|---|
| 0 | Sao Vicente do Penso [SEP] Escudo |
| 1 | Oberlin College [SEP] logo |

Table 5.3 - Final Test Data Predictions

## 5.3 TESTING

```
predict_caption(enc)
```

```
'brown and white dog is running on grass'
```

Figure 5.16 - Results from final test predictions

```
predict_caption(ec)
```

```
'group of people are walking around street'
```

Figure 5.17 - Results from final test predictions

# CHAPTER 6
# RESULTS AND DISCUSSION

This is a method of putting our models in a scenario where they might be applied to an online service. VGG16-Xception and CNN LSTM models are used to predict and analyze the data. The model is set up and furnished. We used Tensor and Bottle Follow the flow of our code to create the test application. They expanded the project using GitHub and Heroku.

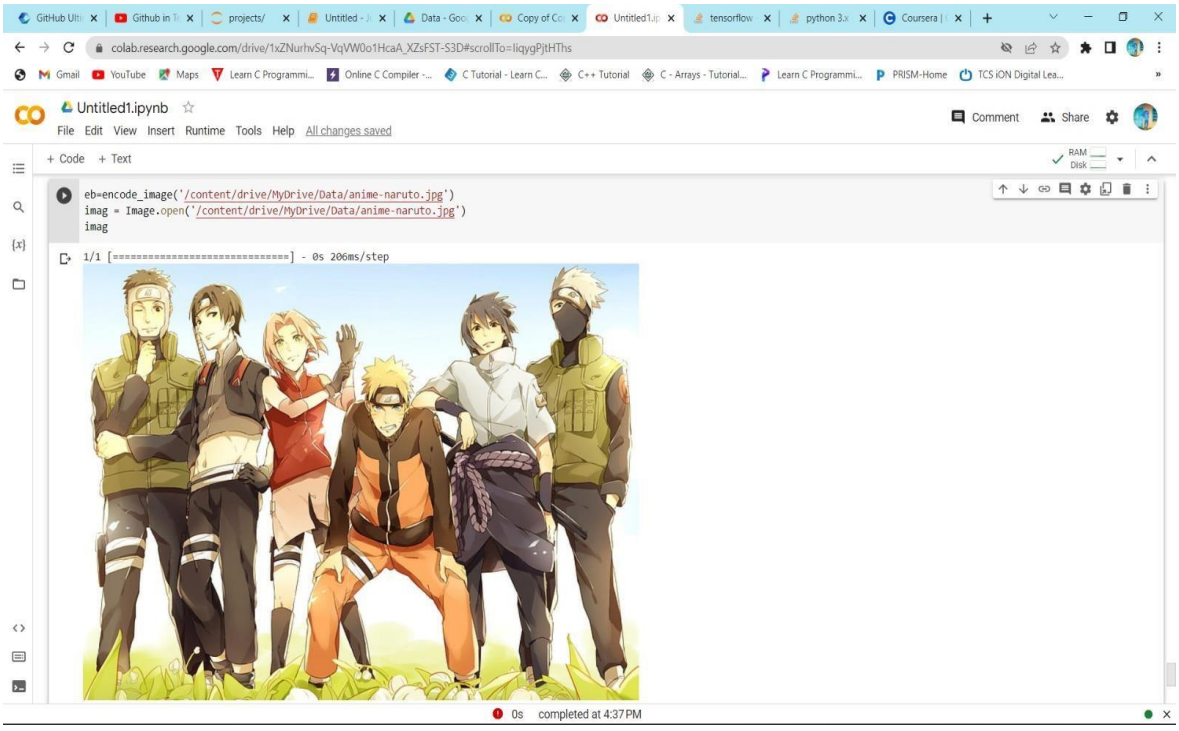



Figure 6.1 - Results

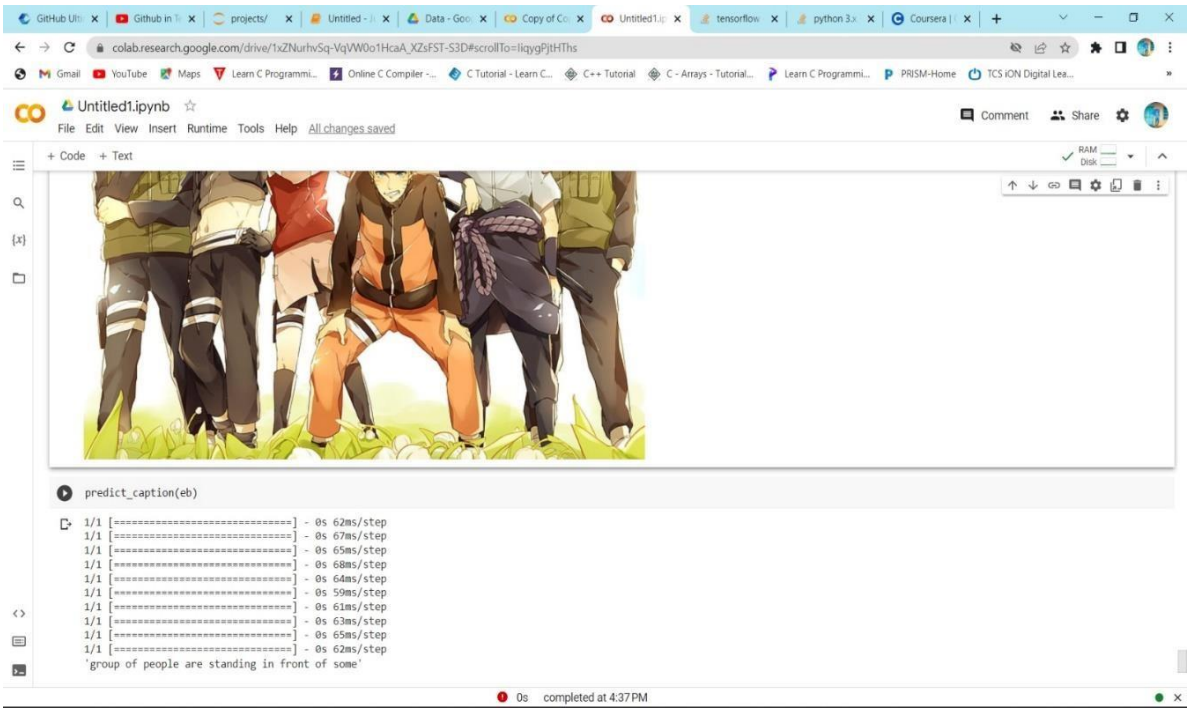



Figure 6.2 – Results

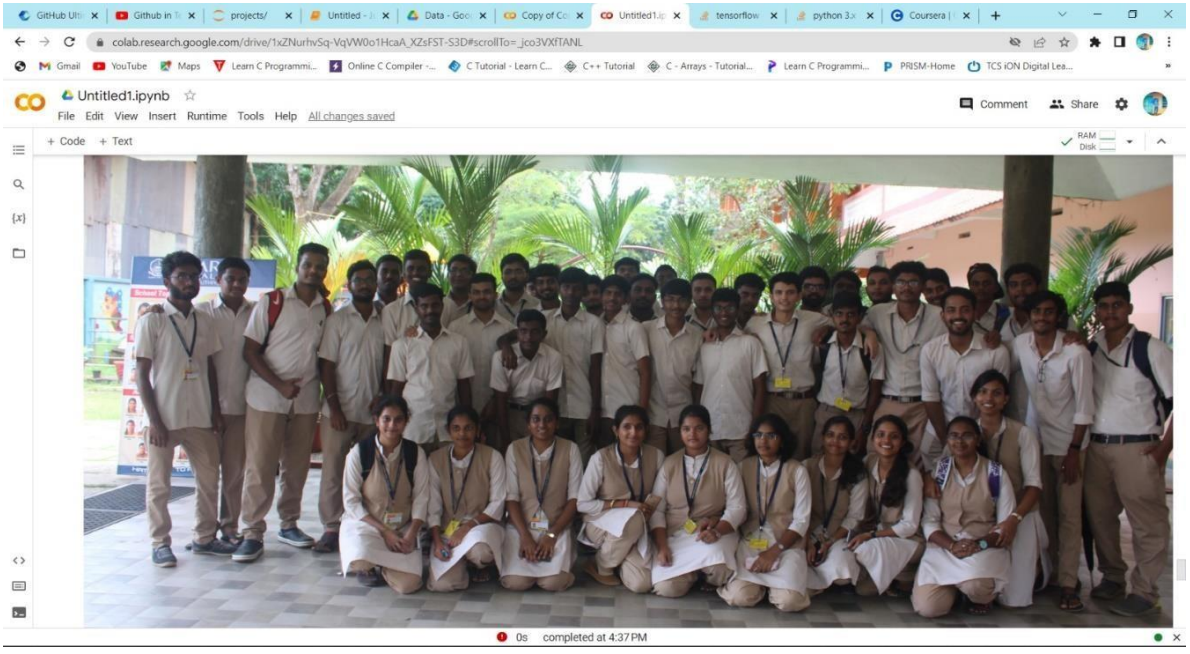



Figure 6.3 – Results

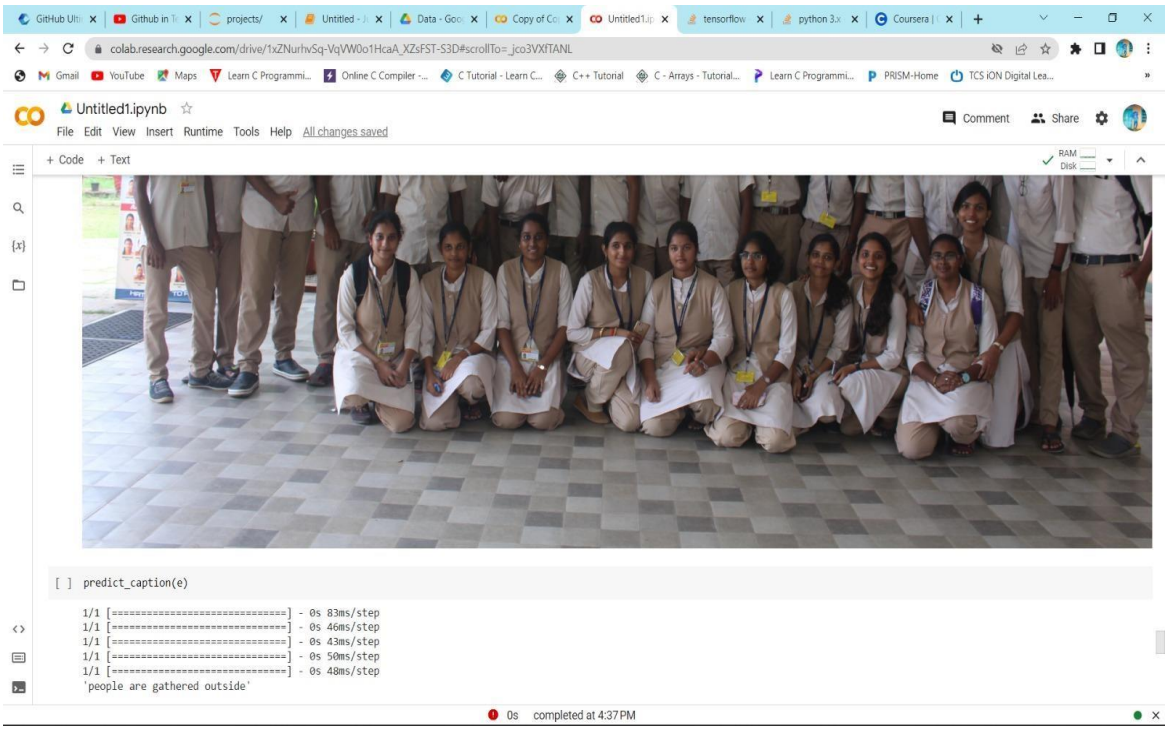



Figure 6.4 – Results

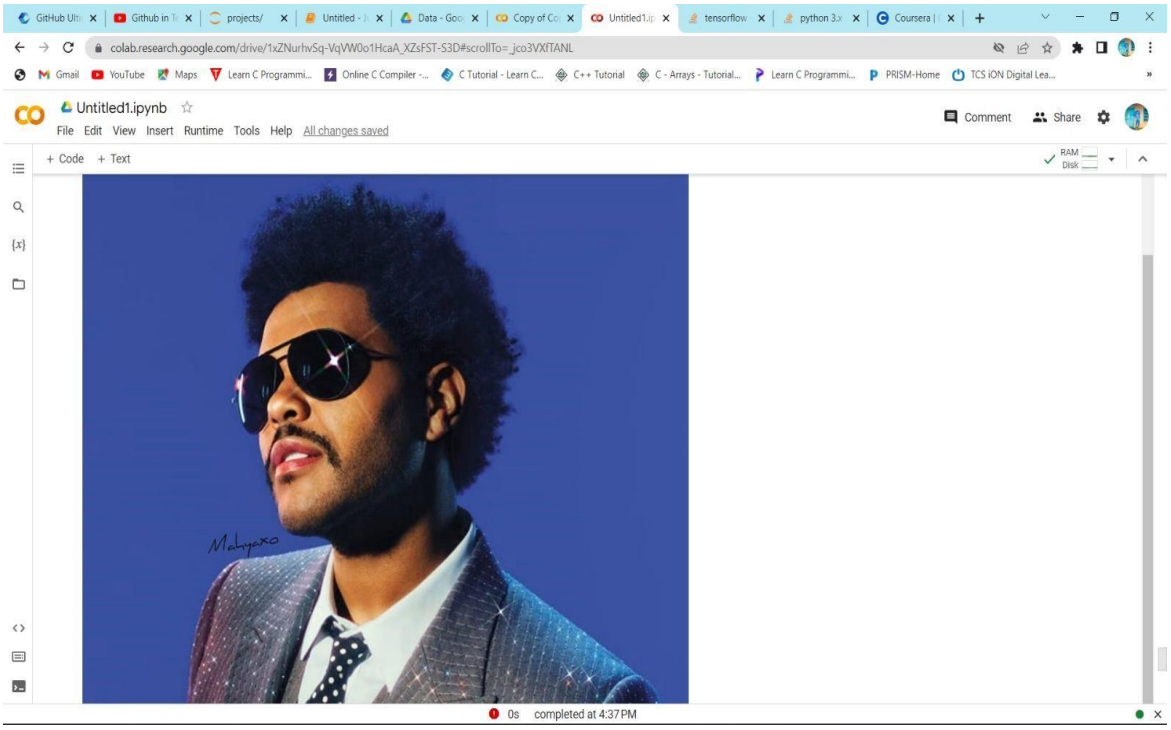



Figure 6.5 – Results

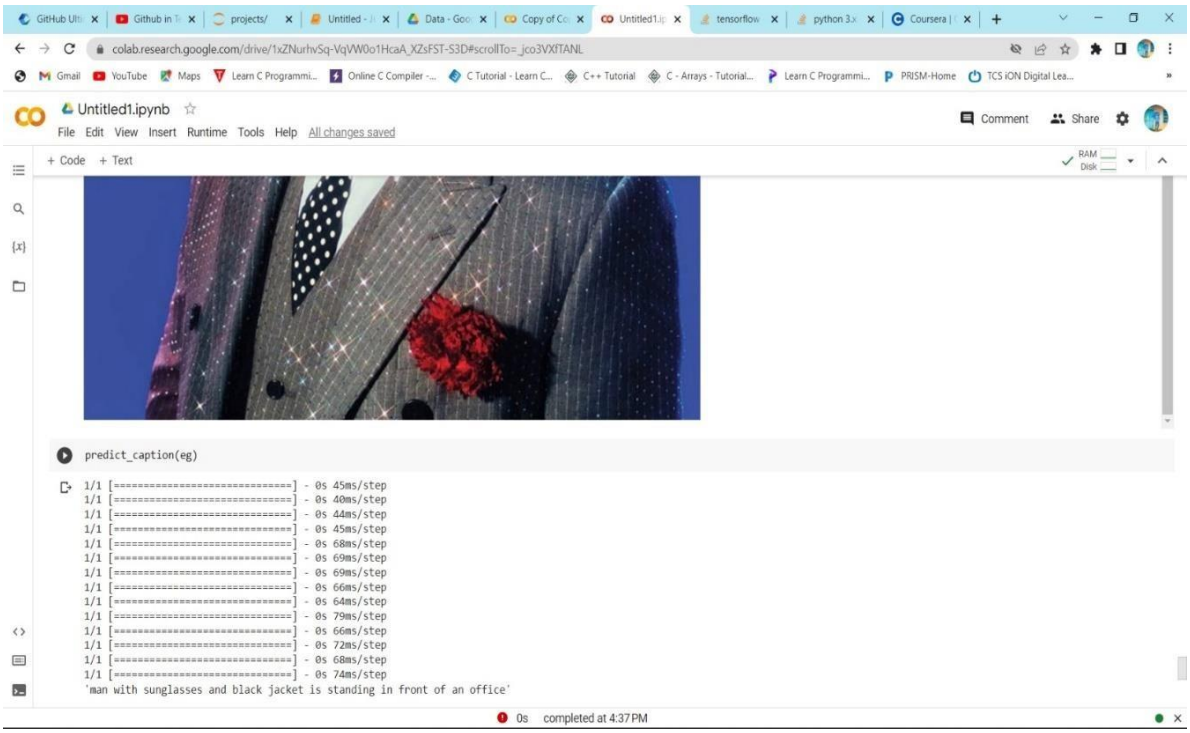



Figure 6.6 – Results

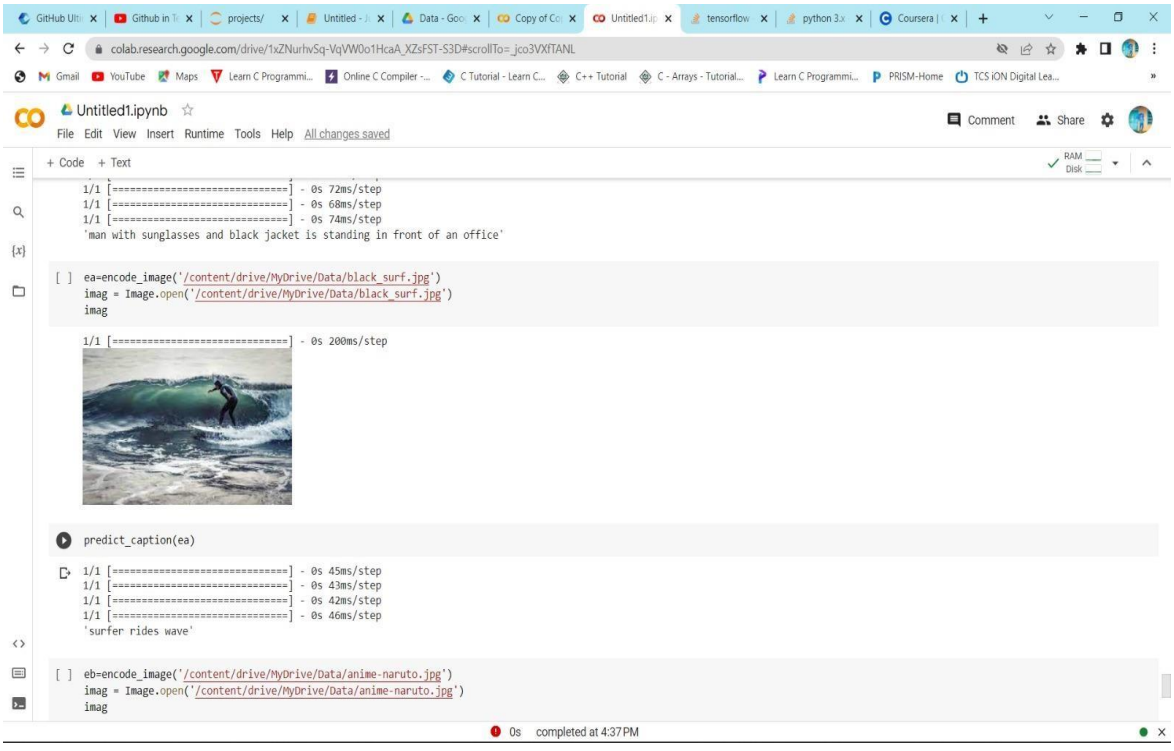



Figure 6.7 – Results

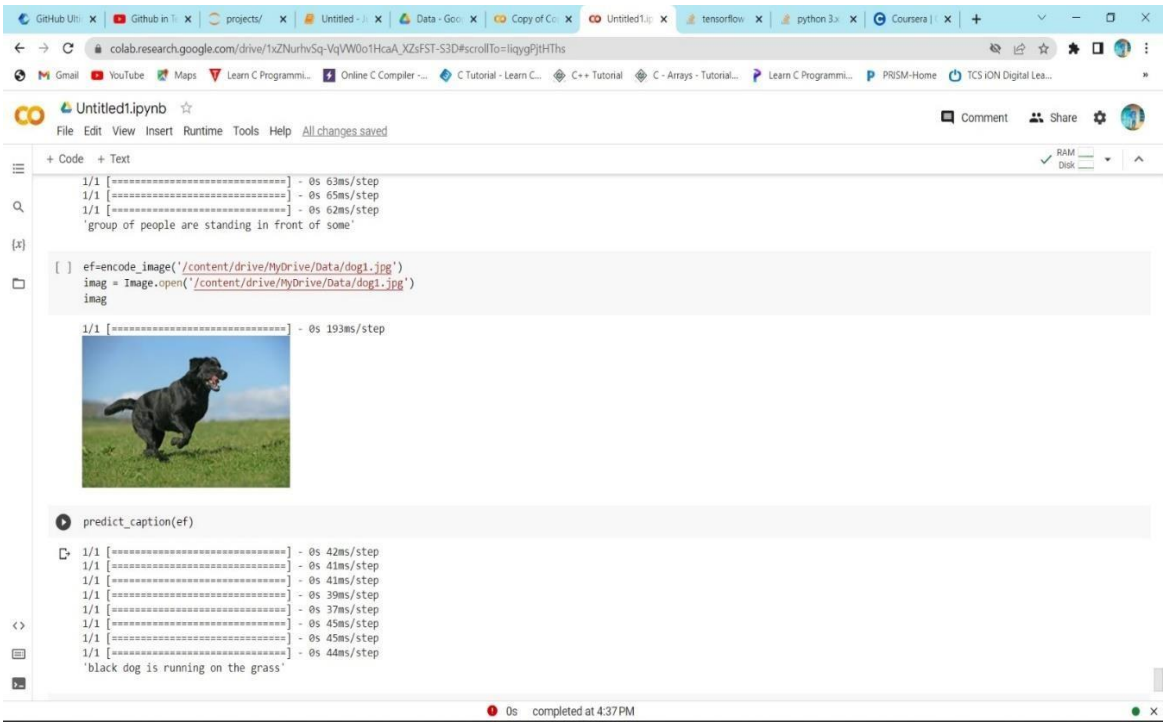



Figure 6.8 – Results

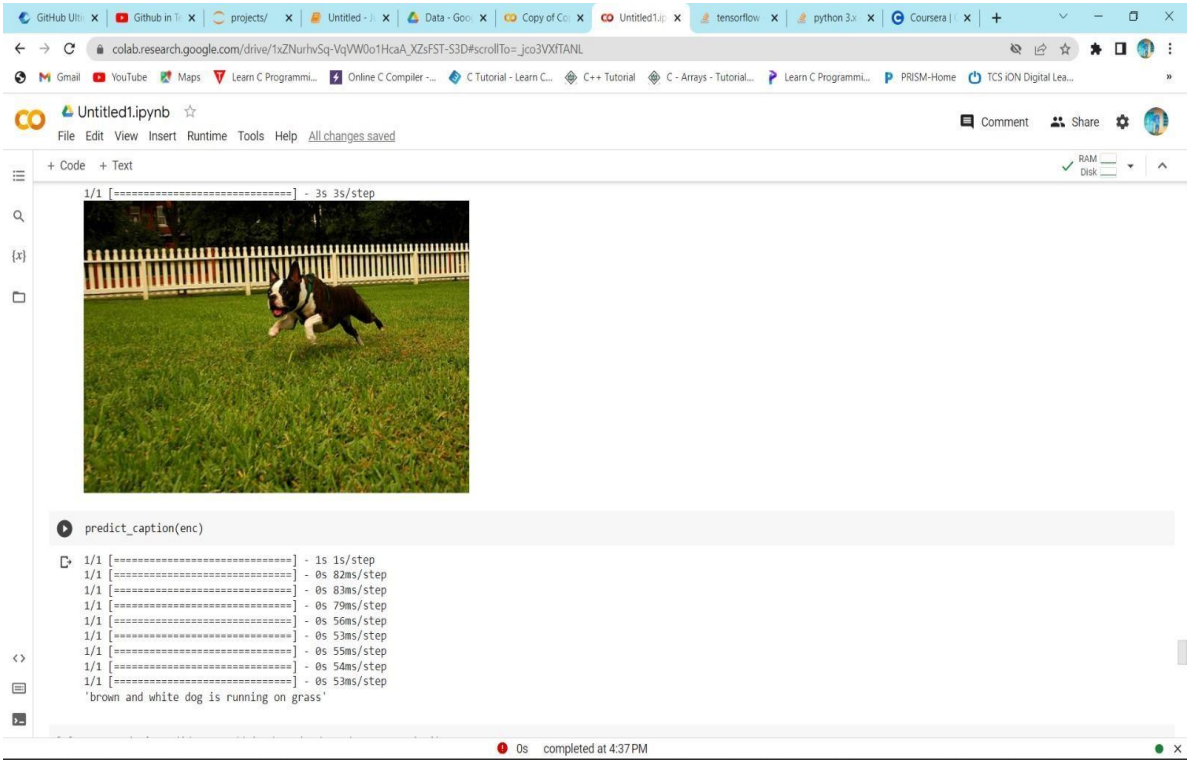



Figure 6.9 – Results

# CHAPTER 7
# CONCLUSION

## 7.1 CONCLUSION

In this Paper, Analysis was done on captions corresponding to text data and image data. This was done in two phases. In Phase-I, the VGG16-Xception model was used to excerpt features from images, and then these are fed into the CNN and LSTM framework for training and to get captions for test data. The model performed well with this data and the results are compared using the BLEU score.

In Phase-II, this model is ready for deployment and this project application was tested using VS Code and deployed in the Heroku app which gives ultimate results for our model.

## 7.2 FUTURE WORK

In practically every sophisticated field of AI (Artificial Intelligence), Captioning of images provides several advancements. This can be further extended by utilizing attention models and by using various Greedy Search Interfaces. This can be even extent for producing captions for live videos as this is a more popular research area nowadays, extracting captions from videos by utilizing text augmentations and can extend to other tasks related to security.

There is still much potential for research and development in image captioning and future work can focus on improving accuracy, relevance and practicality of image captioning models. Some potential future directions include:

Multi-modal image captioning: Current image captioning models focus on generating captions solely based on visual information. However, incorporating other modalities such as audio, text, and sensor data can improve the accuracy and relevance of generated captions.

Context-aware image captioning: Captions should not only describe the objects in the image but also incorporate contextual information such as the scene, emotions, and actions. Developing models that can effectively capture this information can lead to more informative and relevant captions.

## 7.3 RESEARCH ISSUES

There are several research issues related to image captioning using CNN and LSTM models. Some of them are:

- Generating accurate and relevant captions: One of the primary research challenges is to generate accurate and relevant captions for images. This requires developing models that can effectively capture the visual information of the image and the semantic information of the caption.
- Dealing with large datasets: Image captioning requires large datasets for training the model. However, collecting and annotating such datasets can be challenging and time-consuming. Therefore, research needs to focus on developing efficient methods for data collection and annotation.
- Handling variations in image content: Images can have different variations in terms of object placement, lighting, and size, among other factors. Research needs to address how to handle these variations and develop models that can generalize well to different types of images.
- Incorporating contextual information: Captions should not only describe the objects in the image but also incorporate contextual information such as the scene, emotions, and actions. Therefore, research needs to develop models that can effectively capture this information.
- Evaluating model performance: Evaluating the performance of image captioning models is challenging as there is no clear metric for measuring the quality of captions. Therefore, research needs to focus on developing reliable evaluation metrics that can accurately measure the quality of generated captions.

Overall, image captioning using CNN and LSTM models is a complex task that requires addressing several research challenges. Addressing these challenges can lead to the development of more accurate and relevant image captioning models

## 7.4 IMPLEMENTATION ISSUES

There are several implementation issues related to image captioning using CNN and LSTM models. Some of them are:

- Computational complexity: The CNN and LSTM models used for image captioning are computationally intensive and require significant processing power. Implementing these models efficiently can be challenging and may require specialized hardware such as GPUs.
- Preprocessing of images: The images used for training the model need to be preprocessed to extract relevant features. This requires expertise in computer vision and may involve techniques such as edge detection, feature extraction, and resizing.
- Tuning model hyperparameters: The performance of the image captioning model is dependent on several hyperparameters such as the number of layers in the CNN and LSTM models, the learning rate, and the batch size. Tuning these hyperparameters to optimize the model's performance can be time-consuming and requires expertise.
- Handling overfitting: Overfitting occurs when the model is too complex and learns to fit the training data too well, leading to poor generalization to new data. Addressing overfitting requires techniques such as regularization, dropout, and early stopping.
- Vocabulary size: The number of words in the caption vocabulary can be large, and creating the vocabulary requires significant memory. Handling the vocabulary size efficiently can be challenging and may require techniques such as word embeddings or limiting the vocabulary size.
- Transfer learning: Transfer learning techniques can be used to improve the performance of the model by leveraging pre-trained models. Implementing transfer learning requires expertise in selecting appropriate pre-trained models and fine-tuning them for the image captioning task.

Overall, implementing image captioning using CNN and LSTM models requires addressing several technical issues related to computational complexity, data preprocessing, hyperparameter tuning, handling overfitting, vocabulary size, and transfer learning. Addressing these implementation issues can lead to the development of efficient and accurate image captioning models.

# References

[1] A. Garlapati, M. Neeraj, G. Narayanan, “Classification of Toxicity in Comments Using NLP and LSTM,” 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), 2022.

[2] Sanjay S.P., Ezhilarasan N., Anand Kumar M., Soman K.P., “AMRITA-CEN@FIRE2015: Automated story illustration using word embedding (2015), CEUR Workshop Proceedings, 1587, pp. 67 -70.

[3] ] Gautam K.S., Parameswaram L., Thangavel S.K., “A Cascade Color Image Retrieval Framework,” (2020) New Trendsin Computational Vision and Bio-Inspired Computing Selected Works Presented at the ICCVBIC 2018, pp. 23-36, DOI: 10.1007/978-3-030-41862-5_3.

[4] Viswanathan S., Anand Kumar M., Soman K.P, “A Sequence-Based Machine Comprehension Modeling using LSTM and GRU” (2019), Lecture Notes in Electrical Engineering, 545, pp. 47-55, DOI: 10.1007/978-981-13- 5802-9_5

[5] Wang, Chaoyang, Ziwei Zhou and Liang Xu. “An Integrative Review of Image Captioning Research.” Journal of Physics: Conference Series 1748 (2021)

[6] Viktar and Dmitrij Sesok, “Text Augmentation Using BERT for Image Captioning,” Applied Sciences 10 (2020)

[7] Ningthoujam C., Chingtham T.S., “Comprehensive Comparative Study on Several Image Captioning Techniques Based on Deep Learning Algorithm (2022) Lecture Notes in Networks and Systems, 281, pp. 229-240, DOI: 10.1007, / 978-981-16-4244-9_18.

[8] Herdade, Simao, Armin Kappeler, Kofi Boakye and Joao Soares. “Image Captioning: Transforming Objects into Words,” NeurIPS (2019).

[9] Liu, Shuang & Bai, Liang & Hu, Yanli & Wang, Haoran., “Image Captioning Based on Deep Neural Networks,” MATEC Web of Conferences (2018).

[10] Srinivasan, Lakshmi Narasimhan, and Dinesh Sreekanthan, “Image Captioning

– A Deep Learning Approach” (2018).

[11] Wang H., Zhang Y., Yu X., “An Overview of Image Caption Generation Methods,” 2020 Computational Intelligence and Neuroscience, 2020, art. no. 3062706, DOI: 10.1155/2020/3062706.

[12] Yang Z., Wang P., Chu T., Yang J., “Human-Centric Image Captioning,” (2022) Pattern Recognition, 126, art. no. 108545, DOI: 10.1016/j. patcog.2022.108545

[13] Eleison K.C., Hutahaean S.U.I., Tampubolon S.C., Panggabean T.M., Fitriyaningsih I., “An Empirical Evaluation of Phrase-based Statistical machine translation for Indonesia slang-word translator,” (2022) Indonesian Journal for Electrical Engineering and Computer Science, 25 (3), pp. 34 1803-1813, DOI: 10.11591/ijeecs.v25.i3.pp1803-1813.

[14] Lee H., Cho H., Park J., Chae J., Kim J., “Cross EncoderDecoder Transformer with GlobalLocal Visual Extractor for Medical Image Captioning,” (2022) Sensors, 22 (4), art. no. 1429.

[15] Kumar A., Agarawal A., Ashin Shanly K.S., Das S., Harilal N., “Image Caption Generator using Siamese Graph Convolutional Networks and LSTM,” (2022) ACM International Conference Proceeding Series, pp. 306-307.

[16] Mahalakshmi P., Fatima N.S, “Summarization of Text and Image Captioning in Information Retrieval Using Deep Learning Techniques,” (2022) IEEE Access, 10, pp. 18289- 18297.

[17] A. Hani, N. Tagougui and M. Kherallah “Image Caption Generation Using A Deep Architecture,” 2019 International Arab Conference on Information Technology (ACIT), 2019, pp. 246-251.

[18] P. Shah, V. Bakrola and S. Pati, “Image Captioning using deep neural architectures,” 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017, pp. 1-4.

[19] A. Puscasiu, A. Fanca D. -I Gota and H. Valean, “Automated image Captioning,”

2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), 2020, pp. 1-6, doi: 10.1109/AQTR49680.9129930.

[20] O. Sargar and S.Kinger “Image Captioning Methods and Metrices,” 2021 International Conference on Emerging Smart Captioning and Informatics (ESCI), 2021, pp. 522-526, doi: 10.1109/ESCI505509.2021.9396839.

[21] I. Azhar, I. Afyouni and A. Elnagar, "Facilitated Deep Learning Models for Image Captioning," 2021 55th Annual Conference on Information Sciences and Systems (CISS), 2021, pp. 1-6, doi: 10.1109/CISS50987.2021.9400209.

[22] "Show and Tell: A Neural Image Caption Generator" by Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[23] "Beyond Short Snippets: Deep Networks for Video Classification" by Joseph Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[24] "Long-term Recurrent Convolutional Networks for Visual Recognition and Description" by Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[25] "Image Captioning with Recursive Neural Networks" by Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan Yuille. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[26] "Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN)" by Junhua Mao, Xu Wei, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. In Proceedings of the International Conference on Learning Representations (ICLR), 2015.

# APPENDIX

## A. SOURCE CODE

```
from google.colab import drive
drive.mount('/content/drive')

from google.colab.patches import cv2_imshow
from google.colab import drive

drive.mount('/content/gdrive')
!pip install autoviz
#!pip install datatable
!pip install rapidfuzz

from tensorflow.keras.utils import load_img
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.image import load_img
pip install squarify

# Load libraries
import matplotlib.pyplot as plt
import pandas as pd
import pickle
import numpy as np
import os
import seaborn as sns
import plotly.graph_objs as go
import plotly.offline as py
import plotly.express as px
import time
import string
from copy import copy
import progressbar
import tensorflow as tf
import keras
from collections import Counter
from nltk.tokenize import word_tokenize
import altair as alt
from tensorflow.keras.applications import VGG16
from keras import models
```

```
from keras.applications.vgg16 import preprocess_input
from collections import OrderedDict
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from keras import layers
from nltk.translate.bleu_score import sentence_bleu
from wordcloud import WordCloud
from wordcloud import STOPWORDS
import urllib
import PIL.Image
#from autoviz.AutoViz_Class import AutoViz_Class
import squarify
import random
from wordcloud import WordCloud,STOPWORDS,ImageColorGenerator
from IPython.display import Image, HTML
import base64
from PIL import Image
import io
#import datatable as dt
from urllib.parse import unquote
from rapidfuzz import process, fuzz
from tqdm.auto import tqdm
from sklearn.neighbors import LocalOutlierFactor

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

df = pd.read_feather("/content/gdrive/MyDrive/Data/train-00000-of-00005")

print("df size:", df.shape)

df.isnull().sum()
df['language'].value_counts()

## hindi
hi = df[(df['language']=='hi')].reset_index(drop=True)
hi.head()

##german
de = df[(df['language']=='de')].reset_index(drop=True)
de.head(3)
```

```
dft = pd.read_csv('/content/gdrive/MyDrive/Data/test_image_pixels_part-00000.csv', sep='\t', names=
['image_url', 'b64_bytes', 'metadata_url'])
dft
dft.describe()
def get_links(dft, num):
    return dft.image_url[:num].values

links = get_links(dft, 10)


def load_images(links):
    images = []

    for link in links:
        URL = link
        try:

            with urllib.request.urlopen(URL) as url:
                with open('./temp.jpg', 'wb') as f:
                    f.write(url.read())

            img = PIL.Image.open('./temp.jpg')
            img = np.asarray(img)
            images.append(img)
        except:
            continue
    return images

def display_images(images, title=None):
    f, ax = plt.subplots(2,5, figsize=(18,12))
    if title:
        f.suptitle(title, fontsize = 30)

    for i, image_id in enumerate(images):
        ax[i//5, i%5].imshow(image_id)

        ax[i//5, i%5].axis('off')

    plt.show()

    images = load_images(links)
    pip install matplotlib==3.5.0
    display_images(images)

    links = dft.image_url[20:30].values
images = load_images(links)
display_images(images)

links = dft.image_url[30:40].values
images = load_images(links)
display_images(images)

links = dft.image_url[40:50].values
images = load_images(links)
display_images(images)

links = dft.image_url[50:60].values
images = load_images(links)
display_images(images)
```

```
wiki_df = pd.read_csv('/content/gdrive/MyDrive/Data/test_image_pixels_part-00000.csv',
                      sep='\t', names=['image_url', 'b64_bytes', 'metadata_url'])
print(wiki_df)

sub_file = pd.read_csv('/content/gdrive/MyDrive/Data/sample_submission.csv')
sub_file

print(wiki_df.shape)
print(sub_file.shape)

filename = "/content/gdrive/MyDrive/Data/train-00000-of-00005"
sep = "\t"
dft = AV.AutoViz(
    filename,
    sep=sep,
    depVar="",
    dfte=None,
    header=0,
    verbose=0,
    lowess=False,
    chart_format="svg",
    max_rows_analyzed=15000,
    max_cols_analyzed=30,
)

squarify.plot(sizes=df['language'].value_counts().values,
              label=df['language'].value_counts().index,
              color=["green","violet","yellow", "blue"],
              alpha=.8 )
plt.axis('off')
plt.show()

fig = go.Figure(
    data=[ go.Bar(x=df['language'].value_counts().index,
            y=df['language'].value_counts().values,
            text=df['language'].value_counts().values,
            textposition='auto',name='hist', marker_color='skyblue')],
    layout_title_text="WikiMedia Image Dataset Language Distribution"
)
fig.show()

image_file = pd.read_csv('/content/gdrive/MyDrive/Data/test_image_pixels_part-00004.csv',
                         sep='\t', names=['image_url', 'b64_bytes', 'metadata_url'])
image_file
```

```
def showimages(imagelist):
    f, ax = plt.subplots(4,3, figsize=(18,12))
    image_flag=False
    for i, image_id in enumerate(imagelist):
        print(i, image_id)
        with urllib.request.urlopen(image_id) as url:
            if (image_id.lower().find('.svg') != -1):
                print ("Contains given SVG file ")
                image_flag=True
###         if (image_id.lower().find('.tiff') != -1):
###                print ("Contains given TIFF file ")
###                image_flag=True
###            if (image_id.lower().find('.tif') != -1):
###                print ("Contains given TIF file ")
###                image_flag=True
###
            if (image_flag == False):
                with open('./temp.jpg', 'wb') as f:
                    f.write(url.read())

        if (image_flag == False):
            imagetoshow=PIL.Image.open('./temp.jpg')
            print(imagetoshow)
            ax[i//3, i%3].imshow(imagetoshow)
            ax[i//3, i%3].axis('off')
    plt.show()

    manualdisplay=image_file.image_url[60:102].values
showimages(manualdisplay)

start_num=random.randrange(0, len(image_file)-30)
end_num = start_num + 12
imagelist=image_file.image_url[start_num:end_num].values
print(imagelist.dtype)
for index, image in enumerate(imagelist):
    if (image.find('.svg') != -1):
        print ("Contains given SVG file ")
        imagelist[index] =  imagelist[index-1] #work to be done

showimages(imagelist)

file_name = pd.read_feather("/content/gdrive/MyDrive/Data/train-00000-of-00005")
file_name.head(1)

kaggle_mask = np.array(Image.open('/content/gdrive/MyDrive/Data/Kaggle_logo.png'))
#kaggle_mask = np.array(Image.open('../input/kaggle/kaggle-transparent.svg'))
fig = plt.figure()
fig.set_figwidth(10)
fig.set_figheight(15)
plt.imshow(kaggle_mask, cmap=plt.cm.gray, interpolation='bilinear')
plt.axis('off')
#plt.show()

kaggle_wc= WordCloud(background_color='black',max_words = 3000,stopwords='site', mask = kaggle_mask)
kaggle_wc.generate(" ".join(file_name['page_title'].astype(str)))
fig=plt.figure()
fig.set_figwidth(20)
fig.set_figheight(16)
plt.axis('off')
plt.imshow(kaggle_wc, interpolation='bilinear')
plt.show()
```

```
embed_file_sample_df=pd.read_csv('/content/gdrive/MyDrive/Data/test_resnet_embeddings_part-00001.csv')
pixel_file_sample_df=pd.read_csv('/content/gdrive/MyDrive/Data/test_image_pixels_part-00002.csv')

print(embed_file_sample_df.head(2))
print(embed_file_sample_df.columns)

print(pixel_file_sample_df.head(2))
print(pixel_file_sample_df.columns)

file_name.columns

check_cols = ['language', 'mime_type', 'original_height',
              'original_width', 'is_main_image','page_changed_recently']
for cols in check_cols:
    print(file_name[cols].unique())

    #temp_df=file_name[[check_cols]]
temp_df1= file_name.iloc[:, 0]
temp_df2= file_name.iloc[:, 9:14]
temp_df3=pd.concat([temp_df1, temp_df2.reindex(temp_df2.index)], axis=1)
#,file_name.iloc[:,9:12])
temp_df3.head()

#temp_df3.boxplot(by='language')
temp_df3_group=temp_df3.groupby('language').agg('min')
temp_df3_group
temp_df3_group.reset_index(drop=True,inplace=True)
temp_df3_group

temp_df3_group["original_height"]=temp_df3_group["original_height"].astype(float)
temp_df3_group["original_width"]=temp_df3_group["original_width"].astype(float)

temp_df3_group['original_height'].plot(label = 'original_height', figsize = (10,6))
temp_df3_group['original_width'].plot(label = 'original_width', figsize = (10,6))

temp_df3_group['original_height'].plot(label = 'original_height', figsize = (10,6))
temp_df3_group['original_width'].plot(label = 'original_width', figsize = (10,6))
plt.legend()
plt.show()

temp_df3_group['mime_type'].value_counts().plot.bar(figsize = (10,6))
plt.legend()
plt.show()

temp_df3_group['is_main_image'].value_counts().plot.bar(label='is_main_image',figsize = (10,6))
plt.legend()
plt.show()

temp_df3_group['attribution_passes_lang_id'].value_counts().plot.bar(label='attribution_passes_lang_id'
,figsize=(10,6))
plt.legend()
plt.show()

graph_df = pd.concat([temp_df3_group['mime_type'].value_counts(),
               temp_df3_group['is_main_image'].value_counts(),
               temp_df3_group['attribution_passes_lang_id'].value_counts()],
              axis=1, sort=True)
graph_df.columns = ["Mime", "Main Image", "Attribution Passes"]
graph_df.plot.bar(figsize = (10,6))
plt.legend()
plt.show()
```

```
#Ascending order False
top_rated=df.sort_values('original_width', ascending=True)
top10=top_rated.head(10)
f=['page_title','image_url']
displ=(top10[f])
displ.set_index('page_title', inplace=True)

def path_to_image_html(path):
    '''
     This function essentially convert the image url to
     '<img src="'+ path + '"/>' format. And one can put any
     formatting adjustments to control the height, aspect ratio, size etc.
     within as in the below example.
    '''

    return '<img src="'+ path + '""/width="200" height="40" alt="Computer Hope">'

    #Ascending order False
top_rated=df.sort_values('original_height', ascending=True)
top10=top_rated.head(10)
f=['page_title','image_url']
displ=(top10[f])
displ.set_index('page_title', inplace=True)

def path_to_image_html(path):
    '''
     This function essentially convert the image url to
     '<img src="'+ path + '"/>' format. And one can put any
     formatting adjustments to control the height, aspect ratio, size etc.
     within as in the below example.
    '''

    return '<img src="'+ path + '""/width="200" height="40" alt="Computer Hope">'

HTML(displ.to_html(escape=False ,formatters=dict(image_url=path_to_image_html),justify='center'))

!unzip '/content/gdrive/MyDrive/Data/Flickr_Data.zip'

dir_Flickr_text = "/content/gdrive/MyDrive/Data/Flickr8k.token.txt"
dir_Flickr_jpg = "/content/Flickr_Data/Images"

jpgs = os.listdir(dir_Flickr_jpg)
print("The number of jpg flies in Flicker30k: {}".format(len(jpgs)))
```

```
## loading as dataframe
def load_csv(directory):
    desc=dict()
    text = pd.read_csv(directory, delimiter='|',header=None,names=["filename","index","caption"])
    text = text.iloc[1:,:]
    df_new = text[text.iloc[:,2].notnull()]
    print(df_new.iloc[:5,:])
    return df_new

    file = open(dir_Flickr_text,'r')
text = file.read()
file.close()


datatxt = []
for line in text.split('\n'):
    col = line.split('\t')
    if len(col) == 1:
        continue
    w = col[0].split("#")
    datatxt.append(w + [col[1].lower()])

df_txt = pd.DataFrame(datatxt,columns=["filename","index","caption"])


uni_filenames = np.unique(df_txt.filename.values)
print("The number of unique file names : {}".format(len(uni_filenames)))
print("The distribution of the number of captions for each image:")
Counter(Counter(df_txt.filename.values).values())

npic = 5
npix = 224
target_size = (npix,npix,3)

count = 1
fig = plt.figure(figsize=(10,20))
for jpgfnm in uni_filenames[:npic]:
    filename = dir_Flickr_jpg + '/' + jpgfnm
    captions = list(df_txt["caption"].loc[df_txt["filename"]==jpgfnm].values)
    image_load = load_img(filename, target_size=target_size)

    ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
    ax.imshow(image_load)
    count += 1

    ax = fig.add_subplot(npic,2,count)
    plt.axis('off')
    ax.plot()
    ax.set_xlim(0,1)
    ax.set_ylim(0,len(captions))
    for i, caption in enumerate(captions):
        ax.text(0,i,caption,fontsize=20)
    count += 1
plt.show()

def df_word(df_txt):
    vocabulary = []
    for i in range(len(df_txt)):
        temp=df_txt.iloc[i,2]
        vocabulary.extend(temp.split())
    print('Vocabulary Size: %d' % len(set(vocabulary)))
    ct = Counter(vocabulary)
    dfword = pd.DataFrame({"word":list(ct.keys()),"count":list(ct.values())})
    dfword = dfword.sort_values("count",ascending=False)
    dfword = dfword.reset_index()[["word","count"]]
    return(dfword)
dfword = df_word(df_txt)
dfword.head(3)

topn = 50

def plthist(dfsub, title="The top 50 most frequently appearing words"):
    plt.figure(figsize=(20,3))
    plt.bar(dfsub.index,dfsub["count"])
    plt.yticks(fontsize=20)
    plt.xticks(dfsub.index,dfsub["word"],rotation=90,fontsize=20)
    plt.title(title,fontsize=20)
    plt.show()

plthist(dfword.iloc[:topn,:],
        title="The top 50 most frequently appearing words")
plthist(dfword.iloc[-topn:,:],
        title="The least 50 most frequently appearing words")
```

```
import string
def remove_punctuation(text_original):
    text_no_punctuation = text_original.translate(str.maketrans('','',string.punctuation))
    return(text_no_punctuation)

def remove_single_character(text):
    text_len_more_than1 = ""
    for word in text.split():
        if len(word) > 1:
            text_len_more_than1 += " " + word
    return(text_len_more_than1)

def remove_numeric(text,printTF=False):
    text_no_numeric = ""
    for word in text.split():
        isalpha = word.isalpha()
        if printTF:
            print("    {:10} : {:}".format(word,isalpha))
        if isalpha:
            text_no_numeric += " " + word
    return(text_no_numeric)

    def text_clean(text_original):
    text = remove_punctuation(text_original)
    text = remove_single_character(text)
    text = remove_numeric(text)
    return(text)

with progressbar.ProgressBar(max_value=len(df_txt.caption.values)) as bar:
    for i, caption in enumerate(df_txt.caption.values):
        newcaption = text_clean(caption)
        df_txt["caption"].iloc[i] = newcaption
        bar.update(i)

        dfword = df_word(df_txt)
plthist(dfword.iloc[:topn,:],
        title="The top 50 most frequently appearing words")
plthist(dfword.iloc[-topn:,:],
        title="The least 50 most frequently appearing words")


        def add_start_end_seq_token(captions):
    caps = []
    for txt in captions:
        txt = 'startseq ' + txt + ' endseq'
        caps.append(txt)
    return(caps)
df_txt0 = copy(df_txt)
df_txt0["caption"] = add_start_end_seq_token(df_txt["caption"])
df_txt0.head(5)
del df_txt
```

```
modelvgg = VGG16(include_top=True,weights=None)
## load the locally saved weights
modelvgg.load_weights("/content/gdrive/MyDrive/Data/vgg16_weights_tf_dim_ordering_tf_kernels.h5")
modelvgg.summary()

modelvgg.layers.pop()
modelvgg = models.Model(inputs=modelvgg.inputs, outputs=modelvgg.layers[-1].output)
## show the deep learning model
modelvgg.summary()

from tensorflow.keras.utils import img_to_array

images = OrderedDict()
npix = 224
target_size = (npix,npix,3)
with progressbar.ProgressBar(max_value=len(jpgs)) as bar:
    for i,name in enumerate(jpgs):
        # load an image from file
        filename = dir_Flickr_jpg + '/' + name
        image = load_img(filename, target_size=target_size)
        # convert the image pixels to a numpy array
        image = img_to_array(image)
        nimage = preprocess_input(image)
        y_pred = modelvgg.predict(nimage.reshape( (1,) + nimage.shape[:3]))
        images[name] = y_pred.flatten()
        bar.update(i)
    #print(i,filename)

    dimages, keepindex = [],[]
nd=(df_txt0["index"].values)
b = [(int(i)==0) for i in nd]
#for i in nd:
 #   print(int(i)==0)
#df_txt0 = df_txt0.loc[b,: ]
df_txt0 = df_txt0.loc[df_txt0["index"].values == "0",: ]

for i, fnm in enumerate(df_txt0.filename):
    if fnm in images.keys():
        dimages.append(images[fnm])
        keepindex.append(i)

fnames = df_txt0["filename"].iloc[keepindex].values
dcaptions = df_txt0["caption"].iloc[keepindex].values
dimages = np.array(dimages)
print(df_txt0["index"][:5])

## the maximum number of words in dictionary
count_words=22000
#nb_words = 31782
tokenizer = Tokenizer(num_words=8000)
tokenizer.fit_on_texts(dcaptions)
vocab_size = len(tokenizer.word_index) + 1
print("vocabulary size : {}".format(vocab_size))
dtexts = tokenizer.texts_to_sequences(dcaptions)
print(dtexts[:5])

prop_test, prop_val = 0.2, 0.2

N = len(dtexts)
Ntest, Nval = int(N*prop_test), int(N*prop_val)

def split_test_val_train(dtexts,Ntest,Nval):
    return(dtexts[:Ntest],
           dtexts[Ntest:Ntest+Nval],
           dtexts[Ntest+Nval:])

dt_test,  dt_val, dt_train   = split_test_val_train(dtexts,Ntest,Nval)
di_test,  di_val, di_train   = split_test_val_train(dimages,Ntest,Nval)
fnm_test,fnm_val, fnm_train  = split_test_val_train(fnames,Ntest,Nval)

maxlen = np.max([len(text) for text in dtexts])
maxlen
```

```
def preprocessing(dtexts,dimages):
    N = len(dtexts)
    print("# captions/images = {}".format(N))

    assert(N==len(dimages))
    Xtext, Ximage, ytext = [],[],[]
    for text,image in zip(dtexts,dimages):

        for i in range(1,len(text)):
            in_text, out_text = text[:i], text[i]
            in_text = pad_sequences([in_text],maxlen=maxlen).flatten()
            out_text = to_categorical(out_text,num_classes = vocab_size)

            Xtext.append(in_text)
            Ximage.append(image)
            ytext.append(out_text)

    Xtext  = np.array(Xtext)
    Ximage = np.array(Ximage)
    ytext  = np.array(ytext)
    print(" {} {} {}".format(Xtext.shape,Ximage.shape,ytext.shape))
    return(Xtext,Ximage,ytext)

Xtext_train, Ximage_train, ytext_train = preprocessing(dt_train,di_train)
Xtext_val,   Ximage_val,   ytext_val   = preprocessing(dt_val,di_val)
# pre-processing is not necessary for testing data
#Xtext_test,  Ximage_test,  ytext_test  = preprocessing(dt_test,di_test)

print(vocab_size)
## image feature

dim_embedding = 64

input_image = layers.Input(shape=(Ximage_train.shape[1],))
fimage = layers.Dense(256,activation='relu',name="ImageFeature")(input_image)
## sequence model
input_txt = layers.Input(shape=(maxlen,))
ftxt = layers.Embedding(vocab_size,dim_embedding, mask_zero=True)(input_txt)
ftxt = layers.LSTM(256,name="CaptionFeature")(ftxt)
## combined model for decoder
decoder = layers.add([ftxt,fimage])
decoder = layers.Dense(256,activation='relu')(decoder)
output = layers.Dense(vocab_size,activation='softmax')(decoder)
model = models.Model(inputs=[input_image, input_txt],outputs=output)

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

print(model.summary())

start = time.time()
#checkpoint_path = "training_1/cp.ckpt"
#checkpoint_dir = os.path.dirname(checkpoint_path)

# Create checkpoint callback
#cp_callback = tf.keras.callbacks.ModelCheckpoint(checkpoint_path,
 #                                                 save_weights_only=True,
  #                                                verbose=2)

hist = model.fit([Ximage_train, Xtext_train], ytext_train,
                 epochs=7, verbose=2,
                 batch_size=64,
                 validation_data=([Ximage_val, Xtext_val], ytext_val))
               #callbacks = [cp_callback])
end = time.time()
print("TIME TOOK {:3.2f}MIN".format((end - start )/60))

print(Ximage_train.shape,Xtext_train.shape,ytext_train.shape)
```

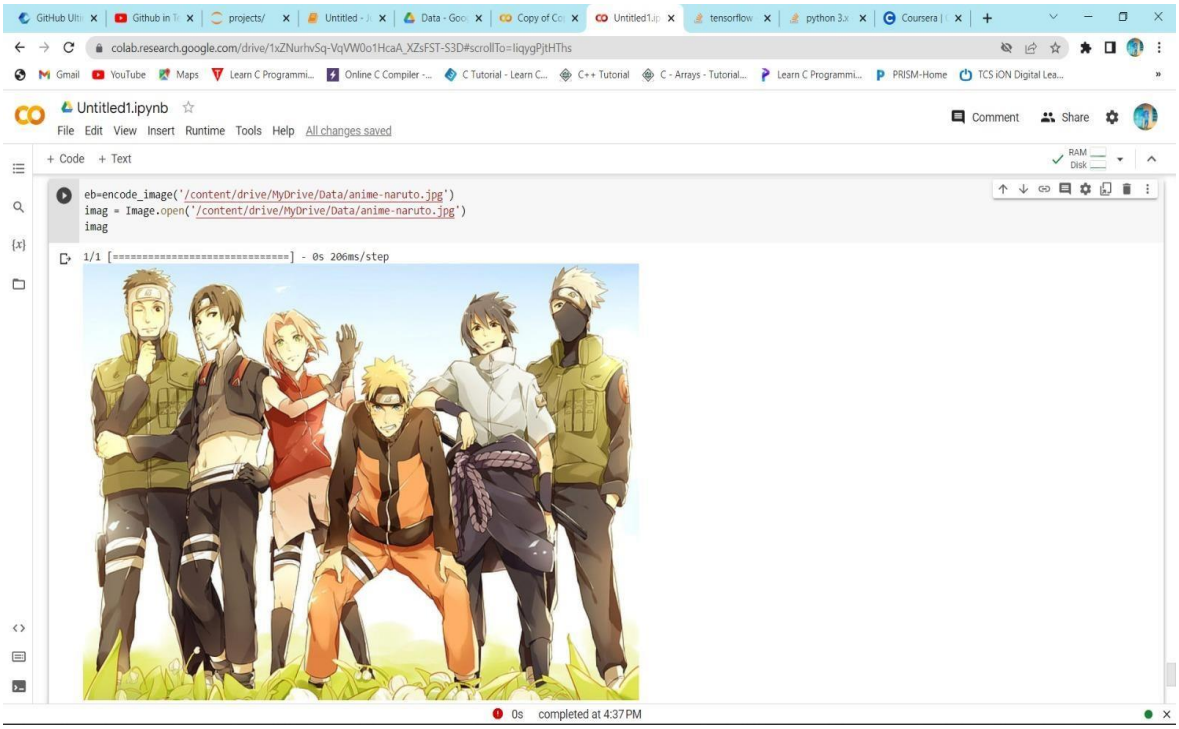
```
index_word = dict([(index,word) for word, index in tokenizer.word_index.items()])
def predict_caption(image):
    '''
    image.shape = (1,4462)
    '''

    in_text = 'startseq'

    for iword in range(maxlen):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence],maxlen)
        yhat = model.predict([image,sequence],verbose=0)
        yhat = np.argmax(yhat)
        newword = index_word[yhat]
        in_text += " " + newword
        if newword == "endseq":
            break
    return(in_text)

    npic = 5
npix = 224
target_size = (npix,npix,3)

count = 1
fig = plt.figure(figsize=(10,20))
for jpgfnm, image_feature in zip(fnm_test[:npic],di_test[:npic]):
    ## images
    filename = dir_Flickr_jpg + '/' + jpgfnm
    image_load = load_img(filename, target_size=target_size)
    ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
    ax.imshow(image_load)
    count += 1

    ## captions
    caption = predict_caption(image_feature.reshape(1,len(image_feature)))
    ax = fig.add_subplot(npic,2,count)
    plt.axis('off')
    ax.plot()
    ax.set_xlim(0,1)
    ax.set_ylim(0,1)
    ax.text(0,0.5,caption,fontsize=20)
    count += 1

plt.show()

index_word = dict([(index,word) for word, index in tokenizer.word_index.items()])


nkeep = 5
pred_good, pred_bad, bleus = [], [], []
count = 0
for jpgfnm, image_feature, tokenized_text in zip(fnm_test,di_test,dt_test):
    count += 1
    if count % 200 == 0:
        print("  {:4.2f}% is done..".format(100*count/float(len(fnm_test))))

    caption_true = [ index_word[i] for i in tokenized_text ]
    caption_true = caption_true[1:-1] ## remove startreg, and endreg
    ## captions
    caption = predict_caption(image_feature.reshape(1,len(image_feature)))
    caption = caption.split()
    caption = caption[1:-1]## remove startreg, and endreg

    bleu = sentence_bleu([caption_true],caption)
    bleus.append(bleu)
    if bleu > 0.7 and len(pred_good) < nkeep:
        pred_good.append((bleu,jpgfnm,caption_true,caption))
    elif bleu < 0.3 and len(pred_bad) < nkeep:
        pred_bad.append((bleu,jpgfnm,caption_true,caption))
```

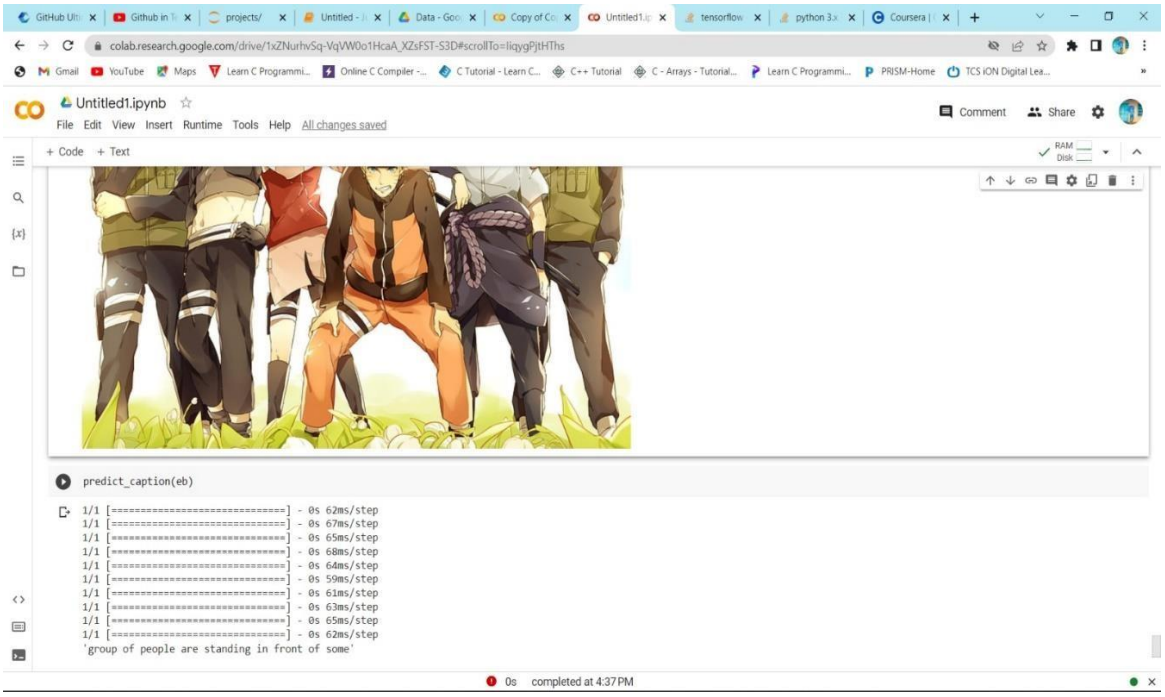
```
def plot_images(pred_bad):
    def create_str(caption_true):
        strue = ""
        for s in caption_true:
            strue += " " + s
        return(strue)
    npix = 224
    target_size = (npix,npix,3)
    count = 1
    fig = plt.figure(figsize=(10,20))
    npic = len(pred_bad)
    for pb in pred_bad:
        bleu,jpgfnm,caption_true,caption = pb
        ## images
        filename = dir_Flickr_jpg + '/' + jpgfnm
        image_load = load_img(filename, target_size=target_size)
        ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
        ax.imshow(image_load)
        count += 1

        caption_true = create_str(caption_true)
        caption = create_str(caption)

        ax = fig.add_subplot(npic,2,count)
        plt.axis('off')
        ax.plot()
        ax.set_xlim(0,1)
        ax.set_ylim(0,1)
        ax.text(0,0.7,"true:" + caption_true,fontsize=20)
        ax.text(0,0.4,"pred:" + caption,fontsize=20)
        ax.text(0,0.1,"BLEU: {}".format(bleu),fontsize=20)
        count += 1
    plt.show()

print("Good Caption")
plot_images(pred_good)

sub = pd.read_csv('/content/gdrive/MyDrive/Data/sample_submission.csv')
sub.head(10)

captions = pd.read_csv('/content/gdrive/MyDrive/Data/test_caption_list.csv')
print(len(captions))
captions.head()

test = pd.read_csv('/content/gdrive/MyDrive/Data/test.tsv', sep='\t')
test.head()

for i in range(5):
    print(test.image_url.loc[i])

    import pandas as pd
    tst0 = pd.read_csv('/content/gdrive/MyDrive/Data/test_image_pixels_part-00000.csv', sep='\t',
header=None)
tst0.head()

from keras.preprocessing import image
import base64

password = "hello world"
encoded = base64.b64encode(password.encode("utf-8"))
print(encoded)
decoded = base64.b64decode(encoded)
print(decoded)

image_64_decode = base64.b64decode(tst0[1].loc[0])
img = Image.open(io.BytesIO(image_64_decode))
img

t = test.image_url.loc[2]

def convert(t):
    t = t.rsplit('/',1)[1]
    t = unquote(t)
    t = t.replace('_', ' ')
    t = t + ' [SEP]'
    return(t)

    test['prediction'] = test['image_url'].apply(convert)
```

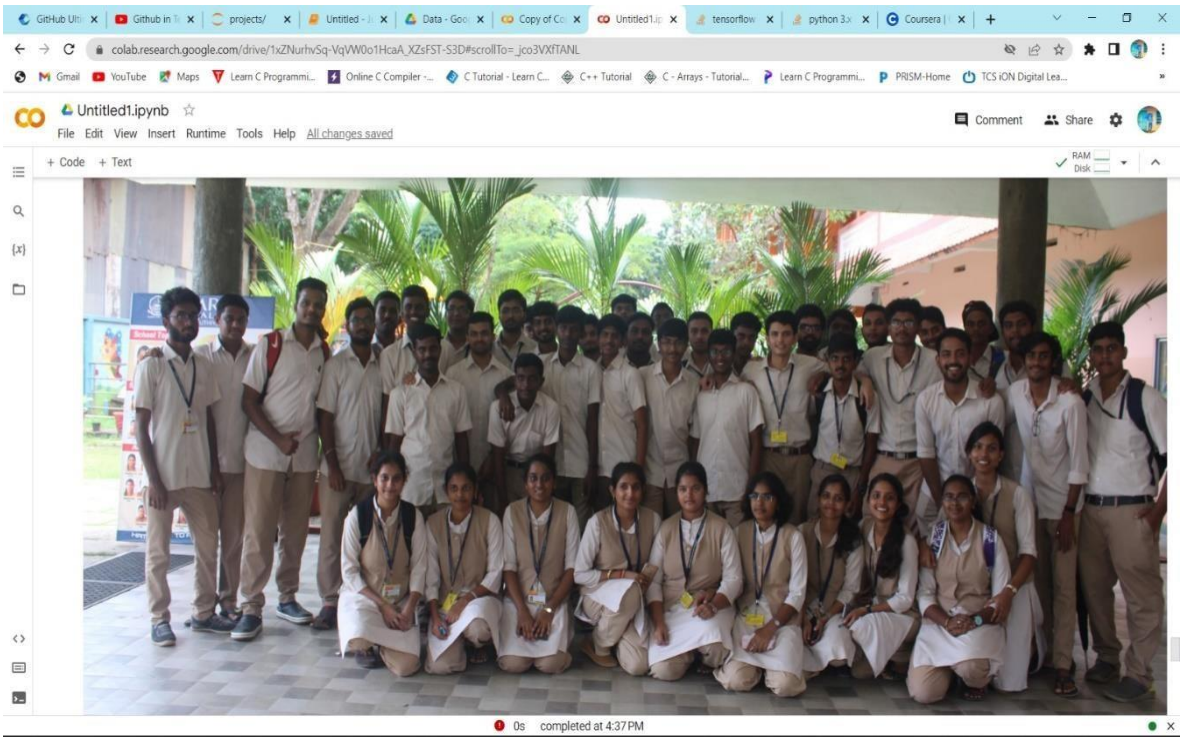
```
test.head()
CAPTIONS = captions.caption_title_and_reference_description.values.tolist()
len(CAPTIONS)

for i in range(5):
    s = test.prediction.loc[i]
    print(f'image_url: {s}')
    res = process.extract(s, CAPTIONS, scorer=fuzz.ratio, processor=None, limit=5)
    print(f'closest captions:')
    for c in res:
        print(c[0])
    print('*'*60)
    print()

    def find_closest_match(s):
    res = process.extract(s, CAPTIONS, scorer=fuzz.ratio, processor=None, limit=5)
    res = [x[0] for x in res]
    return res

    tqdm.pandas()

    test['caption_title_and_reference_description'] =
test['prediction'].progress_apply(find_closest_match)

    sub = test[['id', 'caption_title_and_reference_description']]
sub = sub.explode('caption_title_and_reference_description')
sub.head()

sub.to_csv('testdataprediction.csv', index=False)

from keras_preprocessing.sequence import pad_sequences

from keras.layers import concatenate

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import keras
import json
import pickle
import numpy as np
from PIL import Image
from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input, decode_predictions
from keras.preprocessing import image
from keras.models import Model, load_model
from tensorflow.keras.utils import to_categorical
from keras.layers import Input, Dense, Dropout, Embedding, LSTM

model = load_model("/content/gdrive/MyDrive/Data/model_9.h5")

model_temp = ResNet50(weights="imagenet", input_shape=(224,224,3))

model_resnet = Model(model_temp.input, model_temp.layers[-2].output)

def preprocess_image(img):
    img = image.load_img(img, target_size=(224,224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    return img
```

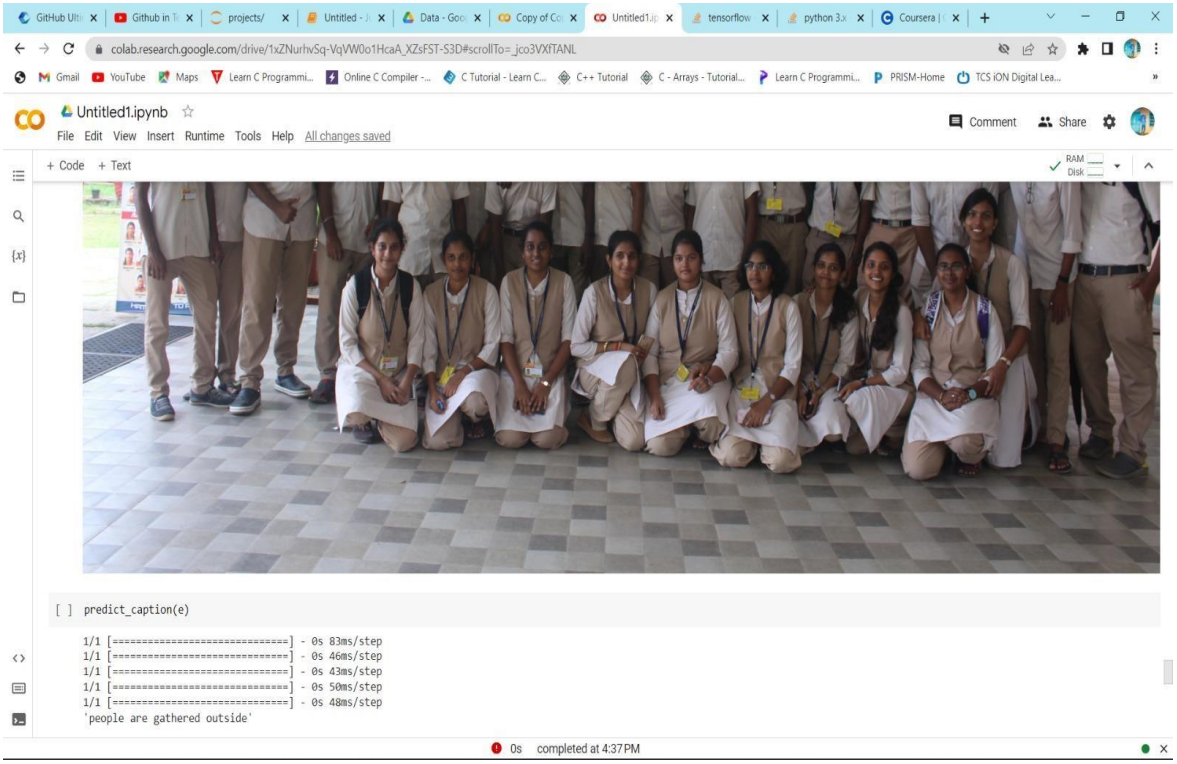
```
def encode_image(img):
    img = preprocess_image(img)
    feature_vector = model_resnet.predict(img)
    feature_vector = feature_vector.reshape(1, feature_vector.shape[1])
    return feature_vector

    def predict_caption(photo):
    in_text = "startseq"
    max_len = 35
    for i in range(max_len):
        sequence = [word_to_idx[w] for w in in_text.split() if w in word_to_idx]
        sequence = pad_sequences([sequence], maxlen=max_len, padding='post')

        ypred =  model.predict([photo,sequence])
        ypred = ypred.argmax()
        word = idx_to_word[ypred]
        in_text+= ' ' +word

        if word =='endseq':
            break


    final_caption =  in_text.split()
    final_caption = final_caption[1:-1]
    final_caption = ' '.join(final_caption)

    return final_caption

    with open("/content/gdrive/MyDrive/Data/word_to_idx.pkl", 'rb') as w2i:
    word_to_idx = pickle.load(w2i)

with open("/content/gdrive/MyDrive/Data/idx_to_word.pkl", 'rb') as i2w:
    idx_to_word = pickle.load(i2w)

    import tensorflow.compat.v2 as tf

    tf.keras.preprocessing.image.load_img
    from tensorflow.keras.utils import load_img, img_to_array
    from keras_preprocessing.image import load_img
    pip install keras
    from tensorflow.keras.preprocessing import image

    enc = encode_image('/content/gdrive/MyDrive/Data/1009434119.jpg')
img = Image.open("/content/gdrive/MyDrive/Data/1009434119.jpg")
img

predict_caption(enc)
ec=encode_image('/content/gdrive/MyDrive/Data/101093029.jpg')
imag = Image.open('/content/gdrive/MyDrive/Data/101093029.jpg')
imag

predict_caption(ec)

enic = encode_image('/content/gdrive/MyDrive/Data/surf.jpg')
img = Image.open('/content/gdrive/MyDrive/Data/surf.jpg')
img

e=encode_image('/content/gdrive/MyDrive/Data/IMG_1430.JPG.jpg')

imag = Image.open('/content/gdrive/MyDrive/Data/IMG_1430.JPG.jpg')
imag.thumbnail((600, 400))
imag
```

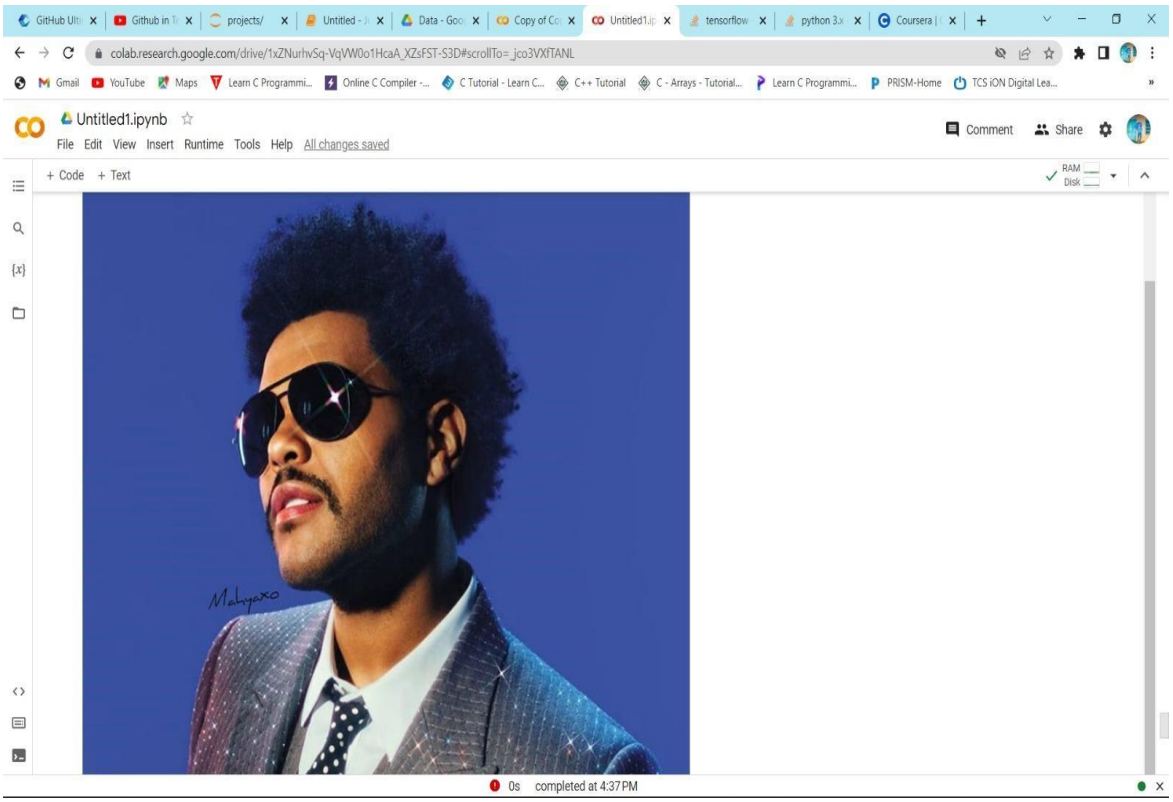
## B. SCREEN SHOTS

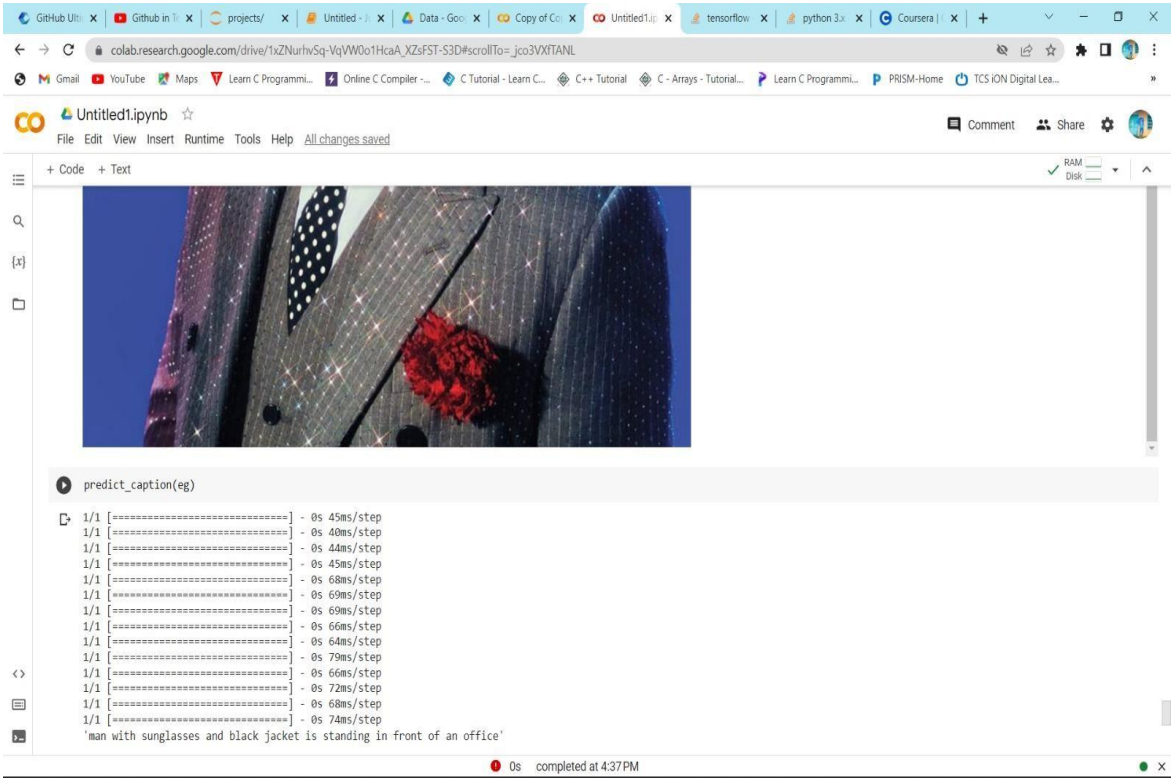


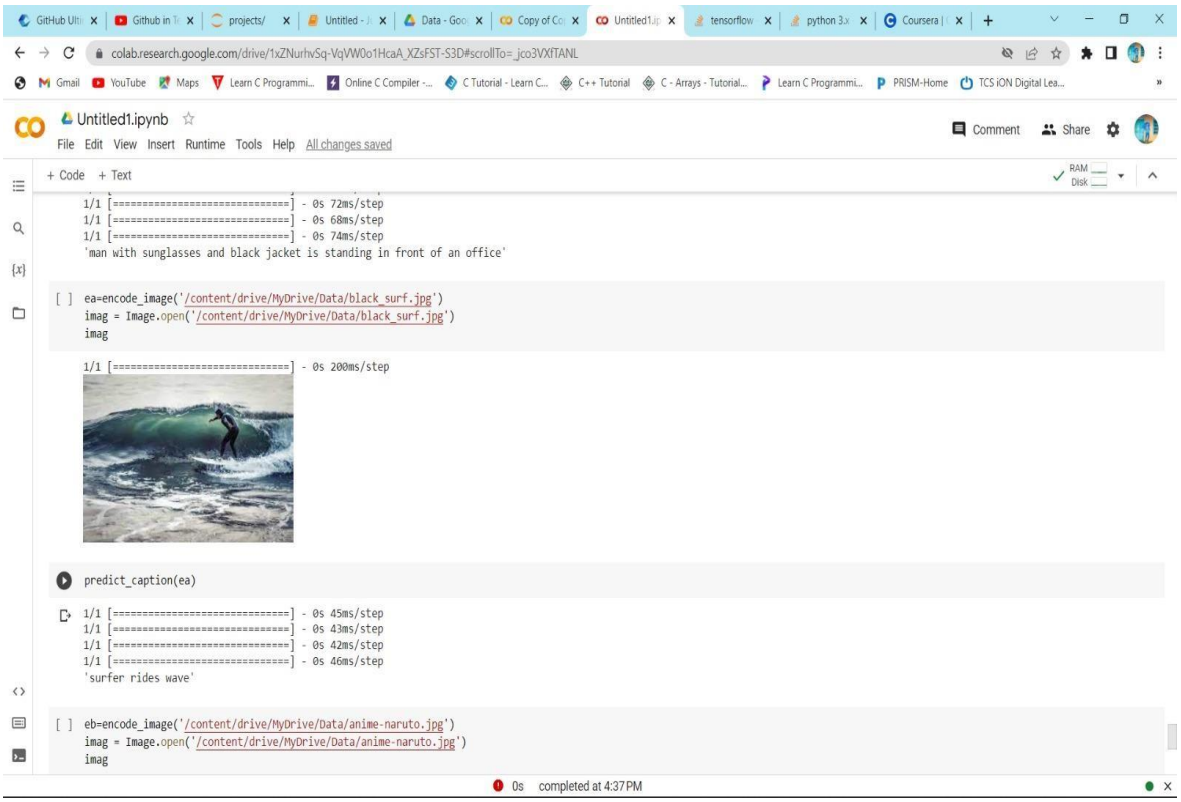



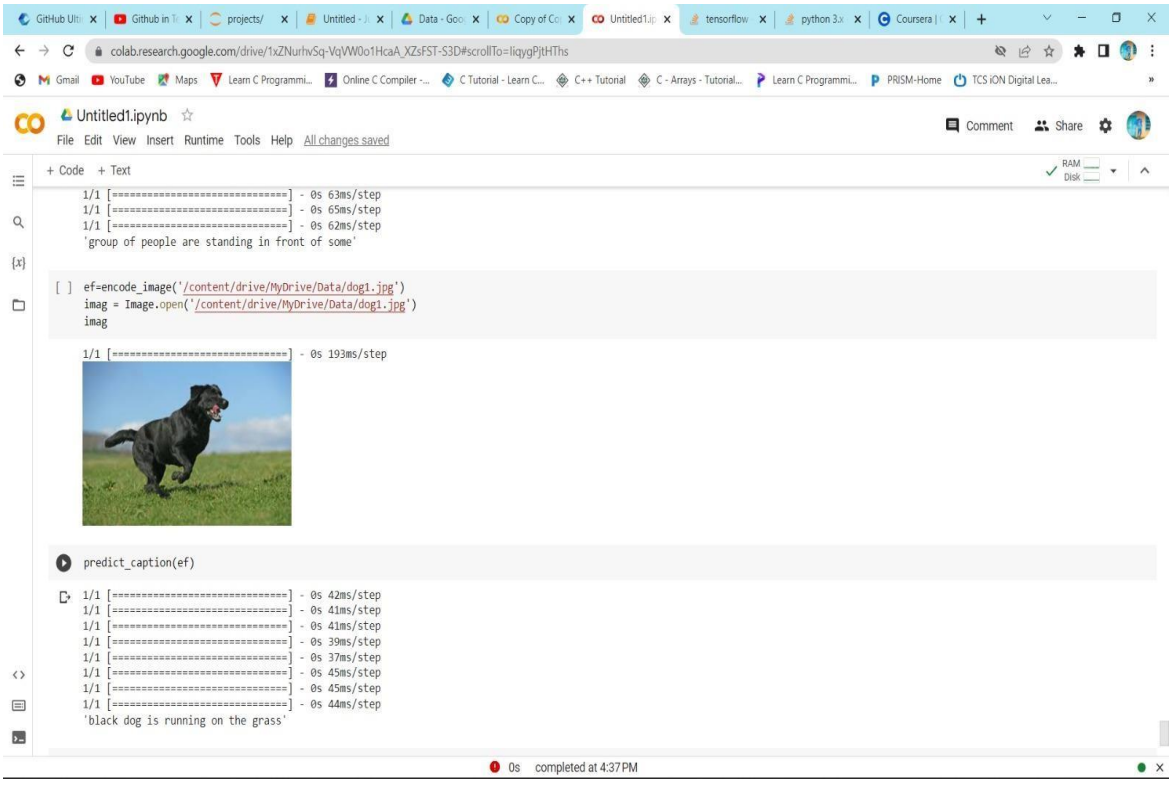

colab.research.google.com/drive/1xZNurhvSq-VqVW0o1HcaA_XZsFST-S3D#scrollTo=_jco3VXfTANL
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
0s completed at 4:37 PM


colab.research.google.com/drive/1xZNurhvSq-VqVW0o1HcaA_XZsFST-S3D#scrollTo=_jco3VXfTANL
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[ ] predict_caption(e)
1/1 [==============================] - 0s 83ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 48ms/step
'people are gathered outside'
0s completed at 4:37 PM


colab.research.google.com/drive/1xZNurhvSq-VqVW0o1HcaA_XZsFST-S3D#scrollTo=_jco3VXfTANL
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
0s completed at 4:37PM


colab.research.google.com/drive/1xZNurhvSq-VqVW0o1HcaA_XZsFST-S3D#scrollTo=_jco3VXfTANL
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
predict_caption(eg)
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 79ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 72ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 74ms/step
'man with sunglasses and black jacket is standing in front of an office'
0s completed at 4:37PM


colab.research.google.com/drive/1xZNurhvSq-VqVW0o1HcaA_XZsFST-S3D#scrollTo=_jco3VXfTANL
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
1/1 [==============================] - 0s 72ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 74ms/step
'man with sunglasses and black jacket is standing in front of an office'
[ ] ea=encode_image('/content/drive/MyDrive/Data/black_surf.jpg')
imag = Image.open('/content/drive/MyDrive/Data/black_surf.jpg')
imag
1/1 [==============================] - 0s 200ms/step
predict_caption(ea)
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 46ms/step
'surfer rides wave'
[ ] eb=encode_image('/content/drive/MyDrive/Data/anime-naruto.jpg')
imag = Image.open('/content/drive/MyDrive/Data/anime-naruto.jpg')
imag
0s completed at 4:37PM


colab.research.google.com/drive/1xZNurhvSq-VqVW0o1HcaA_XZsFST-S3D#scrollTo=liqygPjtHThs
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 62ms/step
'group of people are standing in front of some'
[ ] ef=encode_image('/content/drive/MyDrive/Data/dog1.jpg')
imag = Image.open('/content/drive/MyDrive/Data/dog1.jpg')
imag
1/1 [==============================] - 0s 193ms/step
predict_caption(ef)
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 44ms/step
'black dog is running on the grass'
0s completed at 4:37PM

## C. RESEARCH PAPER

# Image captioning from wikipedia using deeplearning for multilanguage

Nithin Reddy Bhimireddy
Department of CSE
Sathyabama Institute of Science and Technology
Chennai,India
bhimireddynithinreddy@gmail.com

Byna Abhishek Akri
Department of CSE
Sathyabama Institute of Science and Technology
Chennai,India
abhishekakri2@gmail.com

Dr.A.Christy
Department of CSE
Sathyabama Institute of Science and Technology
Chennai,India
ac.christy@gmail.com

***Abstract*— The combination of text class and picture processing has drawn a lot of interest recently due to the advancement of advanced mastering. A significant area of research in the field of computer vision now centers on image captioning. Picture capture requires knowledge of a factor's popularity, attributes, and connections to images. The creation of captions for pictures is a difficult and responsible task; considerable advancements have been made in this area. But because to advancements in deep learning technology and the abundance of information available, the era can now create a version that might display the closest text or subtitles that identify the picture. At first, it was believed to be impossible for a computer to create a photograph. Moreover, NVIDIA is creating a utility to aid visually challenged or blind persons in using photo capture technology. This project's major goal is to produce a version that makes it possible to foresee the caption or text that will appear next to the image. Deep learning techniques can be used to complete this analysis because they are efficient at resolving difficult picture processing problems. So, the goal of this project is to build a model that anticipates captions or labels for images in order to study deep styles. This study utilized the VGG16 structure to predict and provide picture captions, and this analysis may be contrasted with other deep learning systems, including LSTM.**

**Keywords— Text-Classification, Image captioning, Deep Learning, Computer Vision**

## I. INTRODUCTION

The topic of the generation of spoken words or collected images has taken on significant importance in recent years in the field of computer vision with works on natural language processing. Computer vision has recently made significant strides in image processing, including picture categorization and object identification. The term "photo captioning" refers to the explanation of the image's content. Typically, photo capture systems with photo vector input to the encoder employ encoder-decoder designs. A primary interest in photography is taking pictures, which necessitates semantic knowledge of photos and the ability to create accurate, detailed descriptions of pictures. The field of study of photo capture is currently visibly fresh and rapidly evolving. Every day, a number of new approaches are introduced to address the findings of the research in this domain. Picture captions are a brand-new trend that is currently generating a lot of curiosity. Visual capture is a crucial step in improving human-computer interaction and better understanding the underlying concepts of human image recognition. It might be thought about taking pictures of the entire system. The sequential technique is one that uses a series of sentences or words to create visuals that include the elements listed below. The main objective of this photo capture is to generate a natural language description for the entry photo that is sent to the version. In this analysis, we provide a model that depicts characterization as a linguistic technique using herbs. Also, the neural network's functionalities are taken from extraordinary architectures like CNN and LSTM. Finally, using those designs, texture generation became successful.

## II. LITERATURE REVIEW

The researchers behind this work created a hybrid device that employs the Multilayer Convolutional Neural Network (CNN) architecture to produce a term for image categorization and the

Long Short Memory (LSTM) architecture to produce the necessary information. sentences that employ the same words and phrases. Taking pictures is a popular past time that necessitates semantic knowledge of the pictures, including the capacity to create accurate and detailed descriptions of the pictures. In general, full image advent and self-characterization are used in a wide range of applications, such as information image capture, Braille instructions, text-based image retrieval, concept images for clinical analysis, and human-robot communication.

On an image provided by a version, the concept of object identification and appearing actions is obvious. The simplest methods for object detection use the vectors amassed from the images. Using natural language processing to create a function image is undoubtedly difficult. The results of the survey show that this analysis also employs improvements to this approach of item relation transformation that makes it more effective, such as the spatial link between the input facts of the features utilizing geometrical attention. It should be obvious from the articles that came before that CNN is used to translate visual statistics and locate objects in images, whereas RNN or LSTM are used to provide image descriptions. Research analytical techniques and procedures related to processing techniques, such as device learning, natural language processing, and in-depth learning. Several studies have been conducted on image captioning, particularly in the field of computer vision. Creating a program that can predict the text or caption that will show next to an image is the major objective of this research.

## III. DATA SET

Due of each order's unique personality, the records each contain something unique. This report was carried out using a variety of information sources, including data generation, text capture information, picture information, pixel information photos, Resnet embeddings, and so on. This analysis was done using a sizable image collection made available via the cloud of the Wikimedia Foundation. Several languages are used to analyze these statistics. There are 79 or so languages. Using pre-processing techniques and photograph assessment processing, the statistics are assessed in natural language and processed for both text and images.

TABLE I. TRAINING DATA

| language | page URL | Page Title | Section Title |
|---|---|---|---|
| en | https://en.wikipedia.org/wiki/Silverspoon | Silver spoon | Historical uses |
| Zh-TW | https://zh.wikipedia.org/wiki/97/kita-sendai-staion | 北仙台站 | JR 東日本 |

TABLE II. TRAINING DATA

| Mime-Type | Caption reference | height | width | Is main page |
|---|---|---|---|---|
| Image/jpeg | Two silver-giltstrainerspoons on the table | 1194 | 2139 | false |
| Image/jpeg | 月台 | 2112 | 2816 | true |

TABLE III. TEST DATA

| Id | Image URL |
|---|---|
| 0 | https://uplaod.wikimedia.org/wikipedia/scotsGaelicspeakers2011cesnus.png |
| 1 | https://upload.wikimedia.org/wikipedia/ThermopylaeAncientcoastline.jpg |

TABLE IV. CAPTION DATA

| Captions |
|---|
| Albert Pike [SEP] Albert Pike |
| Anna Blount [SEP] Blount and her young daughter, in 1911 |

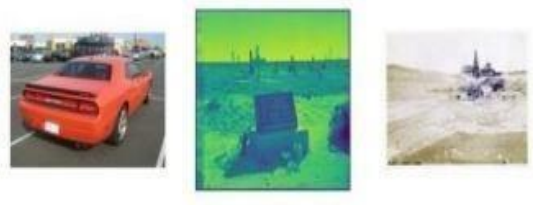


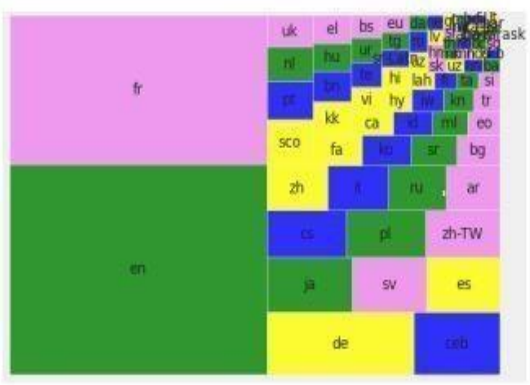
Fig 1. Sample images Data set

## IV. DATA ANALYSIS

Using pre-processing methods to find the information is one of the key phases in data mining. One of the most important processes to process the data before it is used to create any version is this one. With the vast amount of data to test and train the model on, including picture facts, text facts, and caption information, it became challenging to choose the statistics. Each image in the educational information contains five labels that are closest to the image. Every single one of the five accurately captures the image. Given that there were many shooters created with the purpose in mind, it was difficult to decide which information to use in order to create a model for green effects. Here, this initial assessment was conducted utilizing a variety of tools, including histograms, tree tables with sqaurify, photo processing with scalability, alignment, and NLP-based text processing (Natural Language Processing). Stop words and text omission strategies removal techniques, password text techniques, etc.

**Tree Map**: utilized to forecast statistics by

layering rectangles of varying sizes. Each rectangle's size is expressed as a percentage of the total quantity of information that is sent. The elements of the parameters are separated using this display technique. This makes it easier to see how many human values interact to create a body of knowledge. For unique access to the data, data is frequently resolved as pieces of input statistics with exceptional parameters. The tree desk has various types of attributes, including values, text records, route bars, branch values, and many




others.

Fig 2. Tree Map

From Fig 2. Explains that English and French are the two languages that are most commonly utilised in data out of 79.

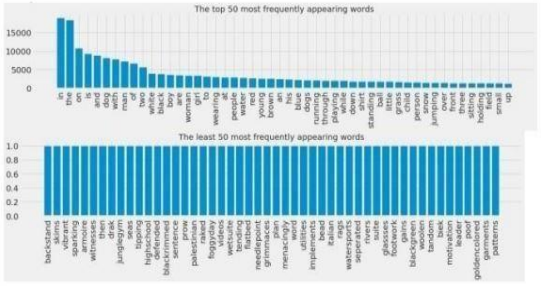



Fig 3. Top most frequent words

**Tokenization.** The password is the first stage in the pipeline for natural language processing. This has a significant effect on how the structure relaxes. You can accomplish this with words or sentences. This is the approach that is used the most frequently in NLP method issues. It breaks down sentences into phrases' constituent parts. Statistics and textual material are divided into chunks of data that can be regarded as independent components by a tokenizer.

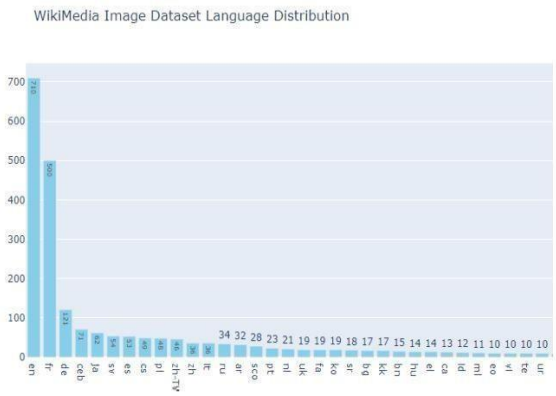



Fig 4. Language Distribution bar graph


Fig 5. Page Title Word Cloud

. The pages connected to the word cloud in the listing are suggested in Figure 5, It lists the terms that are most frequently used in the data.

These are a few of the methods used to analyze the facts and letters contained in the book.

Analysis of picture statistics, including scaling, image smoothing, and pixels, was done in addition to textual content processing and installation information (normalization). These processing methods are used on images since we are unable to simply feed a photograph to a model without first processing it.

**Flattening:** A multi-dimensional array is transformed into a one-dimensional array using this method. It is frequently applied while imposing a one-dimensional apparatus for type styles and carrying out deep mastering challenges. This is because I-D arrays utilize less memory than multidimensional arrays do. The image needs to be softened before you shape the model in order to save time and make working with large sheets easier.

**Normalization (to scale factors):** is a technique for analyzing pictures that modifies the capacity of images. The goal is to make the evaluation green by adjusting the image in accordance with the depth of the fee. The RGB or grayscale photograph entered is normalized using this function.


Fig 6. Analysis of the images related to the page title

Once image processing techniques have been applied to the image data, it is predicted that the results will match the page title.

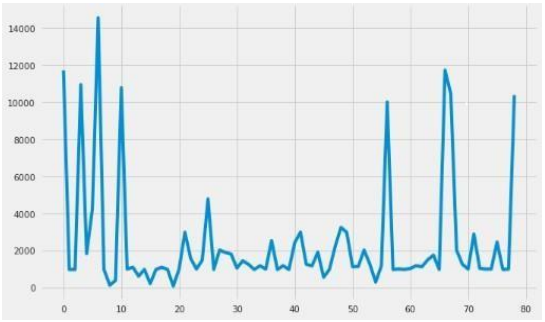



Fig 7. Graph showing picture width dimensions

## V. METHODOLOGY

The two steps of analysis and belief are separated. In phase1, test signatures are trained and predicted using deep learning models like LSTM and VGG16. Phase 2 of this project was developed utilizing VS Code and Heroku, and it grew more advanced for similar development. One of the areas that is most researched and is evolving quickly in everyday life is deep learning. With high-give up contemporary frameworks, neural networks are frequently evaluated in this way. This enables you to employ neural networks that are much larger than before and improve training.

Researchers have proposed a vast array of specific neural network types as upgrades or modifications to the current trends. The rhyme has been used before. This analytical version uses Xception to extract characteristics from photos before sending them to LSTM to create captions. Text is processed using natural language processing and pre-trained models in this version. The highest count for the examined records is predicted by LSTM. The results are compared utilizing the call BLEU after doing all essential analyses, and this daily undertaking is established by the use of Heroku.

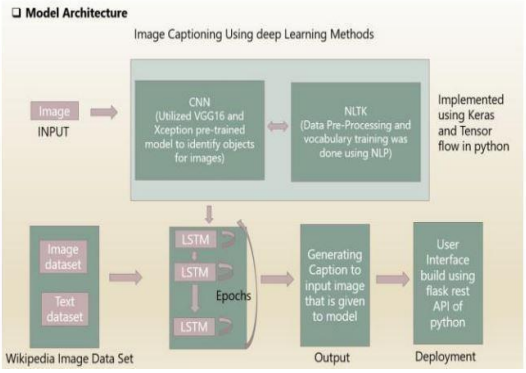



Fig 8. Architectural Model

CNN: Convolutional neural network community, the provided image is warped an objective variable in this sentence. They demonstrate their effectiveness in that they have become the go-to solutions for any forecasting issue that uses data as input to a model. A neural community called a multilayer feedforward network is created by stacking numerous hidden layers in a specific order. While the game stages are frequently supported by layers in the convolutional community layer, these steps are frequently hindered by many layers inside the grid.

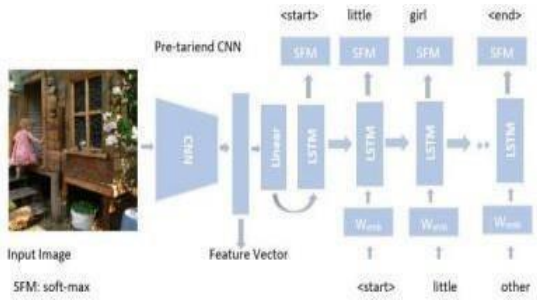



Fig 9. Overview of Architecture

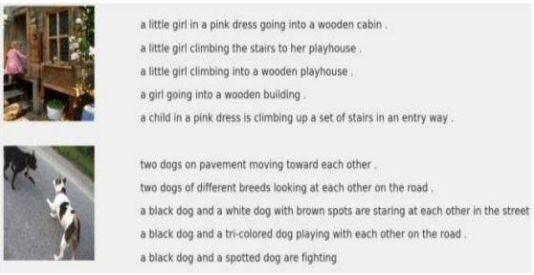



Fig 10. Training phase results

On this architectural model, the predictive photo capture is examined in three stages: function photo extraction, sequential processing, and decoder.

**Feature Extraction:** This is accomplished using Xception, VGG16, and pre-trained styles. This process is called switch studying.
**Sequential Processing:** The embedding layer serves as a means for text data to communicate with one another. It teaches beginners how to extract the given textual features.
Decoder: This is the final level, from which methods for the image extractor with the subsequent processor are developed.
The processor then passes the data to the neuron, which eventually exceeds the final level of output.
**Xception**: With quick setup in mind, this 36-layer premade model has been created. These are advanced to an LSTM layer with processing in a dense layer to provide an upgrade of 2048 vector images. This version is based on extensive data, with functions taken from this analysis and applied to the analysis of the problem's cutting-edge issue. A community photo library with 1,000 different types of photos organized by class is what it needs to be taught on, and this model can be imported right away using Keras packages. We have adjusted our assumptions to reflect the fact that this version is fully loaded with community image.

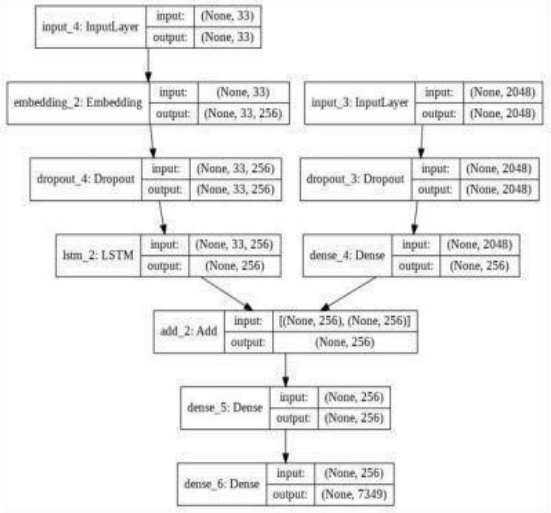
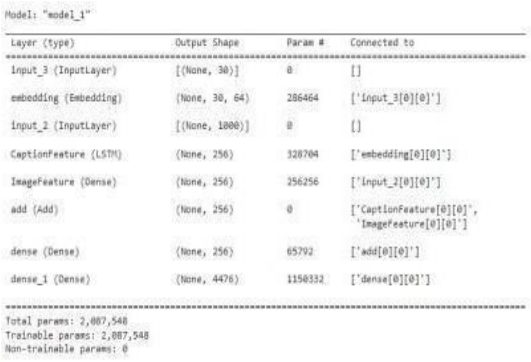



Fig 11. Xception Model Architecture

LSTM:
Short Memory is what LSTM stands for. Using VGG16-Xception models, a raster version is used to extract capabilities from the images in this case. The capabilities can then be in order to generate labels, fed into an LSTM architecture. This version is referred to as range-LSTM, which was developed especially for prediction problems with input data including images and video capture. In order to predict the vectors and further improve prediction, these concepts are retrieved from the input records using the CNN layer and combined with LSTM. These models are being developed for more complicated analyses, such as text categorization, and they show significant potential.


Fig 12. CNN-LSTM Model Architecture

The architectural layout of the multiple LSTM models utilized for function extraction and header derivation is interpreted in Figure 4.6. The table below in Figure 4.7 shows the outcomes of the LSTM training section.



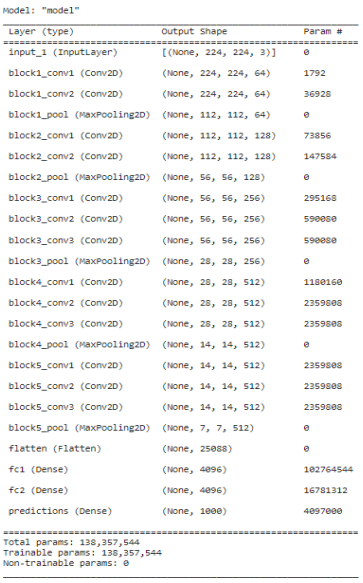
Model: "model_1"

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_3 (InputLayer) | [(None, 30)] | 0 | [] |
| embedding (Embedding) | (None, 30, 64) | 286464 | ['input_3[0][0]'] |
| input_2 (InputLayer) | [(None, 1000)] | 0 | [] |
| CaptionFeature (LSTM) | (None, 256) | 328704 | ['embedding[0][0]'] |
| ImageFeature (Dense) | (None, 256) | 256256 | ['input_2[0][0]'] |
| add (Add) | (None, 256) | 0 | ['CaptionFeature[0][0]', 'ImageFeature[0][0]'] |
| dense (Dense) | (None, 256) | 65792 | ['add[0][0]'] |
| dense_1 (Dense) | (None, 4476) | 1150332 | ['dense[0][0]'] |

Total params: 2,087,548
Trainable params: 2,087,548
Non-trainable params: 0

Fig 13. LSTM effects

The VGG16-Xception framework is used to build the anticipated picture features. To make the LSTM version larger, the VGG16 weights are frozen.
In this investigation, the labels in the test facts were predicted by our model after it had been trained on several categorized images and texts.



Model: "model"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | [(None, 224, 224, 3)] | 0 |
| block1_conv1 (Conv2D) | (None, 224, 224, 64) | 1792 |
| block1_conv2 (Conv2D) | (None, 224, 224, 64) | 36928 |
| block1_pool (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 112, 112, 128) | 73856 |
| block2_conv2 (Conv2D) | (None, 112, 112, 128) | 147584 |
| block2_pool (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 56, 56, 256) | 295168 |
| block3_conv2 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_conv3 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_pool (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 28, 28, 512) | 1180160 |
| block4_conv2 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| block4_conv3 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| block4_pool (MaxPooling2D) | (None, 14, 14, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| block5_conv2 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| block5_conv3 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| block5_pool (MaxPooling2D) | (None, 7, 7, 512) | 0 |
| flatten (Flatten) | (None, 25088) | 0 |
| fc1 (Dense) | (None, 4096) | 102764544 |
| fc2 (Dense) | (None, 4096) | 16781312 |
| predictions (Dense) | (None, 1000) | 4097000 |

Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0

Fig 14. VGG16 results


Fig 15. Trail Images and captions

As a result, they compare using an assessment metric known as BLEU after conducting this examination of schooling and examination duration.
RED scale: It refers back to the titles in order to understand the identity. Accepted dual language understudy's evaluation.
This collection of guidelines is used to evaluate a text translation engine's first-class performance.


Fig 16. Image caption predictions as opposed to BLEU rating.

Our model was further developed to be more green in order to enhance BLEU estimation. Tables V and VI check results for the forecasts following the change are shown below.


TABLE V: TEST DATA PREDICTIONS SAMPLE

| image-URL: Scots Gaelic speakers in 2011 census.png [SEP] |
|---|
| closest captions: |
| Sao Vicente do Penso [SEP] Escudo |
| Jocs Panamericans de 2011 [SEP] Guadalajara |
| Bois de pins et chenes de Madren [SEP] carte |
| Standard time in the United States [SEP] 1913 |
| Hebrides[SEP]Geographic distribution of speakers(2011) |

TABLE VI: TEST DATA PREDICTIONS SAMPLE

| image-URL: Thermopylae ancient coastline large.png [SEP] |
|---|
| closest captions: |
| Oberlin College [SEP] logo |
| Departmento de Flores [SEP] Escudo |
| Sekolah Menengh Kebangssan Selandar [sep] cny 4 |
| Academia de Ciencias de Cuba [SEP] Sede. |
| Geothermal areas in New Zealand [SEP] Geyser Flat |

From Table V and VI, These are the outcomes that may be deduced from training and caption predictions for test data.

TABLE VII: FINAL TEST DATA PREDICTIONS

| Id | Caption title predicted |
|---|---|
| 0 | Sao Vicente do Penso [SEP] Escudo |
| 1 | Oberlin College [SEP] logo |


Fig 17. Final predicted Image


Fig 18. Final Predicted Image

## VI. DEPLOYMENT

This is a method of putting our models in a scenario where they might be applied to an online service. VGG16-Xception and CNN LSTM models are used to predict and analyze the data. The model is set up and furnished. We used Tensor and Bottle Follow the flow of our code to create the test application. They expanded the project using GitHub and Heroku.


Fig 19. Web Page


Fig 20. Results


Fig 21. Results

## VII. CONCLUSION

Analysis of signatures in this text that match linguistic and visual information. This was accomplished in stages. In Phase 1, features from the images were extracted using the VGG16-Xception version, and they were then Injections were done into the CNNLSTM form in order to create and collect signatures in the experimental data. The results are compared to the usage of the call BLEU, and the model performed well with these records. This version is prepared for deployment in Phase II, and to deliver our final product, the app design is reviewed for the use of Code and delivered to the Heroku app.

## VII. FUTURE SCOPE

The image offers some improvements in practically all complex areas of AI (artificial intelligence). The usage of interest models and multiple greedy search engines like Google can similarly extend this. Because it has been a more well-known site for research recently, it may also be used to create live captions, extract captions from videos using colored text, and perform other security-related tasks.

## REFERENCES


- Garlapati, M. Neeraj, G. Narayanan, "Classification of Toxicity in Comments Using NLP and LSTM," 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), 2022.
- Sanjay S.P., Ezhilarasan N., Anand Kumar M., Soman K.P., "AMRITACEN@FIRE2015: Automated story illustration using word embedding (2015), CEUR Workshop Proceedings, 1587, pp. 67 -70.
- Wang H., Zhang Y., Yu X., "An Overview of Image Caption Generation Methods," 2020 Computational Intelligence and Neuroscience, 2020, art. no. 3062706, DOI: 10.1155/2020/3062706.
- Yang Z., Wang P., Chu T., Yang J., "Human-Centric Image Captioning," (2022) Pattern Recognition, 126, art. no. 108545, DOI: 10.1016/j. patcog.2022.108545
- Eleison K.C., Hutahaean S.U.I., Tampubolon S.C., Panggabean T.M., Fitriyaningsih I., "An Empirical Evaluation of Phrase-based Statistical machine translation for Indonesia slang-word translator," (2022) Indonesian Journal for Electrical Engineering and Computer Science, 25 (3), pp.1803-1813,DOI: 10.11591/ijeecs.v25.i3.pp1803-1813.