INTRODUCTION

1. INTRODUCTION

Humans can see an image and can tell what the image is about, but a machine cannot tell what the image is about. The process of describing the context of an image using simple English sentences is a very important task. Moreover, it could be of great assistance for visually impaired people as it would help them understand the context of the image available on the web or locally saved in their system. Also, it is very helpful in providing information about the images depicted in the scenarios such as images captured in smartphones. Moreover, a textual description of the images is not sufficient, but an image must also express how the objects are related to each other in terms of their attributes and activities involved in it.

Human beings usually describe a scene using natural languages that are concise and compact. However, computers describe the scene by taking an image that is a 2-D array. The objective is to map the images and captions to the same space and learning a mapping from the image to the sentences. The RNN method not only models the one-to-many (words) image captioning but also models many-to-one action generation and many-to-many image descriptions. The model has three components. The first component is a CNN that is used to understand the content of the image. The understanding of image answers the typical questions in Computer Vision such as “What are the objects?”, “Where are the objects?” and “The features in the image?” For example, given an image of “Girl sitting on the grass-covered in paint,” the CNN has to recognize the “Girl”, “grass” and their relative locations in the image. The second component is an RNN that is used to generate a caption given the visual feature. For example, the RNN has to generate a sequence of probabilities of words given two words “Girl, grass”. The third component is used to generate a caption by exploring the combination of the probabilities.

1.1 Problem Statement

Human beings can see an image and can easily tell what the image is about, but a computer cannot tell what the image is representing. Also, it would be a greater help to visually impaired people to help them better understand the content of the images on the web or locally saved in their system. It is very useful in providing more accurate and compact information of images.

1.2 Purpose

Nowadays, the impact of social media is much more in anyone's life. Use of Caption Generator may be helpful in many situations such as providing assistance to help visually impaired people as it would help them understand the context of the images, Auto-subtitling.

1.3 Scope

The application work for captured images. It can even work for images that do not have any objects. In such cases, the application would not generate any caption depicting that the image is not having any object or features in it to predict the actions. The application is limited to accept one image at a time.

1.4 Proposed Solution

Machine learning can be defined as a computer system that is used to perform a specific task without using explicit instructions with the empirical study of algorithms and statistical models. With the improvement in Machine Learning techniques and accessibility of huge datasets and computation power, it is possible to create models that will generate captions for an image. These algorithms are used for building a model with the help of training data and then used to achieve the objective by testing the model on the testing data.

- The system is made to learn with the different images using CNN and LSTM algorithms. The model is to be tested for its accuracy with BLEU Score.
- The basic working of the application is that the objects are identified from the images and features are extracted from it using pre-trained VGG16 model (CNN) and then fed to the LSTM model along with the captions to train.
- The trained model is then capable of generating captions for any images that are fed to it.

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Background Study

The Background study is done to explore the basic technicalities required to build the system and check whether there is a feasibility to build the product.

The background study is done to find the answer to the following questions:

1. How features can be extracted from text for caption identification?
2. How objects are identified from the images?
3. How features can be extracted from the images which includes objects & actions performed by the objects?
4. How the captions again to be converted as meaningful sentences?

Any image comprises objects in it. Objects in the images can be a person, thing or an animal. Every object has a unique characteristic that performs unique actions. Pillow is a python library that supports a variety of different image manipulating formats like opening, manipulating and saving the images. Pickle library is used to store the file in pickle file format which can be later called when the objects are identified from the images and features are extracted from them.

Flickr8k dataset is selected instead of Flickr30k and MSCOCO datasets as both of them take weeks to train the model. The dataset consists of approximately 8092 images and 60000 corpus for the images. The better the model is trained; the better caption gets generated for the image.

For better training, the features have to be extracted and it has to be trained. Various CNN algorithms can be used for feature extraction. Some of the algorithms are VGG16, VGG19, InceptionV3. “Comprehensive Guide to Convolutional Neural Networks” article explains how objects are identified from the images and features are extracted from them. But machines find it difficult to process the images in RGB format. So the images are converted into Greyscale format ie in X-Y axis format with the help of ConvNet so that the images are reduced into a form which is found easier to process, without losing features which are critical for getting a good prediction. This is done by converting the images in the 2-D array. The objects in the frame can be identified with the help of correlation of X coordinates. If two X coordinates are

of same correlation, then it means the object is of the same type. It could be a human, animal or a thing. But if there is a correlation between X and Y coordinates, then it means the image is having more than an object in it and from that, it could be identified what kind of actions is taking place in it. The Pooling layer method similar to Convolutional layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. It is useful for extracting dominant features that are efficient in training the model.

The initial text pre-processing can be done using any text pre-processing package. But how to remember the words, its sequences and sentence formation while learning. For this purpose, the recurrent neural network (RNN) can be useful. But the greater problem with recurrent neural network is that recurrent neural network has only short-term memory. So, while processing the text, it might leave the important information as it moves on to the later steps or layers. In simple, it has the vanishing gradient problem. To solve this problem, LSTM model can be substituted in place of RNN. In LSTM, the words get converted to vectors and RNN passes through each vector and while processing it takes the previous step state and the current state and holds the information seen before. The LSTM decides to keep or forget the information based on its significance level. This methodology may be helpful in generating the captions from the extracted text.

2.2 Feasibility Study

A Feasibility study is conducted to determine whether the project is feasible from the organization or a particular stakeholder's point of view. The word Feasibility means the process of analysing a proposed system to determine whether it is feasible or not. When the advantages are more than disadvantages, the system is considered feasible.

- **Technical Feasibility**

This project is developed by using open source software and hence no licenses are required to develop the application. Also, there is no difficulty in maintaining the system as well. With the help of Anaconda distribution, there is no hassle of installing large packages in python. It has most of the packages pre-installed in its distribution. Newer releases and patches are easily available to download as well. Therefore the system is technically feasible.

- **Economic Feasibility**

Development of this application is economic. No cost is involved in purchasing this software. Hence zero development cost. Amount of time for developing is also less. It was built for over 4 months. Therefore, the system is economically feasible.

- **Operational Feasibility**

The application requires only a browser like Google Chrome, Firefox or Safari and software's like Python3, TensorFlow and Keras preinstalled for working. The application has got a very simple user interface. The user does not require any specific training to work on the system. Hence the system is operationally feasible.

2.3 Related Work

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, Barbara Plank, (Journal of Artificial Intelligence Research submitted on 15th Jan 2016) have discussed about their project as a challenging problem as the model was not working properly with natural images that have recently received a huge amount of attraction from the computer vision and natural language processing communities. Also, they have classified the existing approaches based on how they conceptualized this problem. They have helped in reviewing the detailed description of existing models along with their advantages and disadvantages.

Jiuxiang Gu, Gang Wang, Jianfei Cai, Tsuhan (2017 IEEE International Conference on Computer Vision) explained about the effectiveness of their approach is validated on two datasets: Flickr30K and MS COCO. The extensive experimental results show that their method outperforms the vanilla recurrent neural network-based language models and is competitive with state-of-the-art methods. With 30000 images, the author was able to get 76 % accuracy.

MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, Hamid Laga, (Journal on Deep Learning for Image Captioning submitted on 14th October 2018) said that although deep

learning-based image captioning methods have achieved remarkable progress in recent years, a robust image captioning method that can generate high-quality captions for all images is yet to be achieved.

Kenneth P. Camilleri Marc Tanti, Albert Gatt, (Journal on Computer Vision and Pattern Recognition submitted on 7th August 2017) said that a recurrent neural network (RNN) is typically viewed as the primary 'generation' component. The authors suggest that the image features should be 'injected' into the RNN. They have viewed the RNN algorithm as only encoding the previously generated words. According to the authors, RNN algorithm should only be used to encode linguistic features and that only the final representation should be 'merged' with the image features at a later stage. Two architectures are being compared in this journal. As suggested RNNs are better viewed as encoders, rather than generators.

2.4 Drawbacks In Existing System

Consider an example of Captionbot.ai application, a product from Microsoft. It is an ML application that can understand the content of any image. When a person uploads a photo, it is sent to Microsoft for image analysis. But it doesn't allow users to upload any images of their choice. The users can only test the application with the given template of images on their website.

HARDWARE AND SOFTWARE REQUIREMENTS

3. HARDWARE AND SOFTWARE REQUIREMENTS

3.1 Hardware Requirements

RAM: 4GB and above

Hard disk: 120GB and above

Processor: Intel i3 and above

3.2 Software Requirements

Operating System: Ubuntu 16.04, Windows 8 and above

Front end: HTML5, CSS3, Bootstrap

Back end: Flask 1.x

Development Tool: Python 3.7

Storage (Dataset): Google Drive

IDE: Jupyter Notebook (Anaconda 3), VS Code

Packages: TensorFlow, Keras

Other Technologies used: Git and GitHub

3.3 Tools And Technologies

3.3.1 Tools

- **Anaconda 3**

A premium predictive analytics, scientific computing and large-scale data processing. The open-source software aims to simplify application development through its wonderful application deployment through package management. The tool fulfils to set up the user-defined better environment by integrating different required packages that are essential to explore the essentials.

- **Jupyter Notebook**

A web application that allows users to live code, write equations, visualize graphs and texts. Mainly meant for Data Analytics, Data Science and Machine Learning purposes.

- **VS Code**

It is an editor from Microsoft that supports all types of Operating Systems. Advantages of this application are that it allows users to install all kinds of extensions necessary to help the person to code such as gitlens, gitcontrol, auto code completion etc.

- **Draw.io**

A free online diagram software that allows users for making flowcharts, process diagrams, charts, UML, ER and network diagrams.

- **GitHub**

It is an application used during software development for tracking changes in source code. Any changes in the code can be tracked with the help of this integration tool. The changes can be added, committed and pushed to the Repository. Benefits of it include distributed, non-linear workflows, data integrity and speed.

3.3.2 Technologies

- **Python 3.7**

Python is a free, expressive, extensible, cross-platform programming language. It is an interpreted language that provides easy integration. The pseudo-code aspect of Python is one of its most prominent features. Python supported Flask framework even helps to integrate the code with HTML5. Python is incredibly simple to learn with.

- **Flask**

A web framework which supports many tools, libraries and technologies which help in running a web application on localhost server. Flask is very simple to get started for a beginner level of coding as it helps in deploying the application on the server very easily.

- **HTML (Hypertext Mark-up Language)**

A markup programming language to create webpage layouts to be displayed in web browsers like Google Chrome etc. It can be assisted with technologies such as CSS and JavaScript.

- **CSS (Cascading Style Sheets)**

A programming language (style sheet) for good presentation of documents coded with the help of HTML. It also works alongside HTML and JavaScript.

- **Bootstrap**

A framework that helps in making the website created by the user responsive with the help of different templates, navigation buttons, forms, typography and button designs.

- **TensorFlow**

TensorFlow uses many functions some important function and usabilities are mentioned here. Keras use TensorFlow in the background for learning and generating the model. Keras uses

TensorFlow to enable deep neural network. Placeholder() function helps as to declare image as runtime variable where feed_dict variable inside placeholder() helps to assign different frames all the time. variable_scope() helps to understand what are all the variables going to be used in that function scope.get_variable() helps to use a variable with some stored memory for it. These variables can even be used again anywhere inside the variable scope. conv2d() function is used to convert the image to 2D image for further processing as handling 3D data is more complex than handling 2D data. relu() is an activation function on neurons during learning to generate the model. Here relu converts values to 1 and -1. Some basic functions like reduce_sum() to calculate the sum of the given value, div() to divide the values from one another. reduce_max() to calculate the maximum value in given values.

- **Keras**

It is an open-source neural network library written in Python which is capable of running on top of R, Microsoft Cognitive Toolkit, TensorFlow, Theano or PlaidML. It is designed to enable fast experimentation with deep neural networks as it focuses on being modular, user-friendly and extensible.

- **NLTK**

Natural language Toolkit is a package of libraries for statistical natural language processing for text written in English in Python programming language. It is mainly used for text processing, classification, tokenization, stemming, tagging, parsing and semantic reasoning. This tool has been really wonderful for teaching and working especially to play with natural language.

- **Pillow**

Pillow is a free, open source python imaging library for the python programming language that helps in opening, manipulating and saving many different image file formats. It is mainly used for image processing for reading, writing, cutting, pasting, and merging of images. Geometrical transforms, colour transforms, image enhancement, image sequences etc.

- **Numpy**

Numpy helps in performing many basic calculations like empty() is to create array of zeros with mentioned size. argmax() helps to find the max value in given array. ndarray() to define values as array and accessible with square brackets. transpose() converts matrix rows into column and column into rows. ndim() is used to find dimension of the array defined whether the array is 2-D or 3-D. shape is used to get the number of elements in each dimension.

- **Pickle**

Python object structure can be serialized and de-serialized with the help of python pickle module so that it can be pickled and saved on disk. It is a way to convert a python object like list, dict etc into character stream.

- **Tokenizer**

Tokenizer is a library in Keras which is used for preparing and pre-processing text data. It helps in splitting words with text_to_word sequence. Also used for one hot encoding, hash encoding with hashing_trick.

- **Sequential model**

Sequential model is a layers library inside keras package where each layer has exactly one input tensor and one output tensor. It is not appropriate when the model has multiple inputs or multiple outputs.

- **ModelCheckpoint**

Machine learning model can take hours, days or even weeks to train. If it is stopped unexpectedly, you might lose your work progress. So with the help of keras library, it is possible to checkpoint the machine learning model.

- **Load_model**

Load_model library is used to load the saved model as they can be saved during and after training. It is used to resume the model progress where it was last stopped and avoid long training times.

- **Werkzeug**

Werkzeug is Web Server Gateway Interface web application library. It is a CGI library used to serve some HTTP requests by adding header into the response.

SOFTWARE REQUIREMENTS SPECIFICATION

4. SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Users

The users of the system can be anyone who wishes to generate a caption for the images that they have captured. Visually impaired people can also be the major users when the input capture is automated.

4.2 Functional Requirements

Following are the functional requirements of the system

- **Data Gathering**

For ML projects, the gathering of a valid dataset is the most basic and essential functional requirement. The dataset used for building the caption generator model is Flickr8k dataset. Out of 8092 images in the dataset, 6000 is used for training. The dataset also consists of around 60000 corpus for the images which are used as a textual description for training the model.

- **Pre-Processing**

The corpus in the dataset is pre-processed for removal of stop words, articles, punctuation marks and digits from the textual descriptions. This is done to generate a bag of words thus helping in the mapping of words with the corpus. The file is saved in pickled format for training purpose.

- **Object Identification**

This is done by converting the image from RGB format to Greyscale to find the correlation between X coordinates. If there is a correlation between two X coordinates, then it means the object is the same. An image is converted to Greyscale format by converting the image in 2-D array. Having images in Greyscale format makes it easier to process without losing features.

- **Feature Extraction From Images**

This is possible with the help of VGG16 model, one of the CNN algorithm. Feature extraction is done to predict the action being portrayed in the image. Extracted features are then dumped in a pickle file format for model generation.

- **Model Generation Using CNN**

The extracted features dumped in pickle file are then passed through the VGG16 model along with a pre-processed bag of words for model generation.

- **Building LSTM Model**

Simultaneously the CNN model is passed through the LSTM model for predicting the caption of the images and displaying the same.

- **Validating The Model**

The developed model is evaluated by calculating the BLEU Score. It helps in evaluating the quality of text which has been machine-translated from one natural language to another.

- **Deploy The ML Model In The Web Application**

The model is deployed into the web application with the help of Flask as it helps in integration of python code with the HTML page.

4.3 NON FUNCTIONAL REQUIREMENTS

- **Usability**

It can be easily used by anyone in one go with one image at a time. The project has user interface support that gives easy interaction for the user. It helps to give a better visualization of the image by depicting the scenario in the image and predicting the action taking place in it. No maintenance is required for this application.

- **Reliability**

The failure frequency of the system is very low since all type of images are handled carefully. So the system is durable. The predictability of the ML model is good enough to portray the scenario in the image and tell what kind of action is taking place in it. Hence, the learned rate is also good. Therefore, we can rely on the results obtained.

- **Supportability**

Supportability is the one which can be measured using the runnable platform of the project. The project can be executed in Windows/Linux/macOS machines. This project requires Python and a few packages like TensorFlow, Keras pre-installed in it. Hence one can conclude that this project is more supportable.

SYSTEM DESIGN

5. SYSTEM DESIGN

The Design represents how the system can be built. This can be better understood with the help of design diagrams. This chapter comprises a few design diagrams to better understand and represent the system.

5.1 Data Flow Diagram

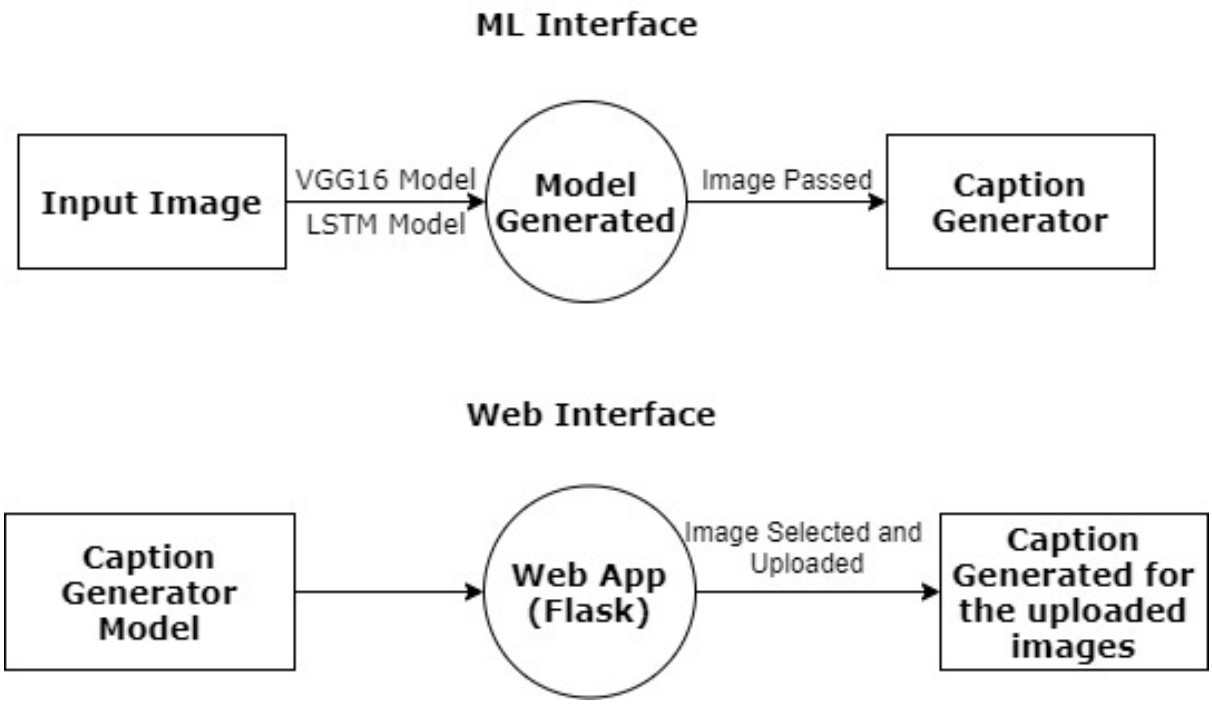

Fig 5.1 DFD Level 0

The above image tells about the Data Flow taking place in the Project. Here in ML interface, one can see that the images are passed through the VGG16 Model (CNN Algorithm) for feature extraction and then passed through the LSTM Model that helps in predicting captions. The model is generated and then saved in pickle format. The images are passed through this saved model which helps in generating captions.

To deploy the ML model in the web application, the pickle file format of the model is loaded into the flask that in turn helps in binding the python code with HTML and running it on localhost server.

5.2 Process Flow Diagram

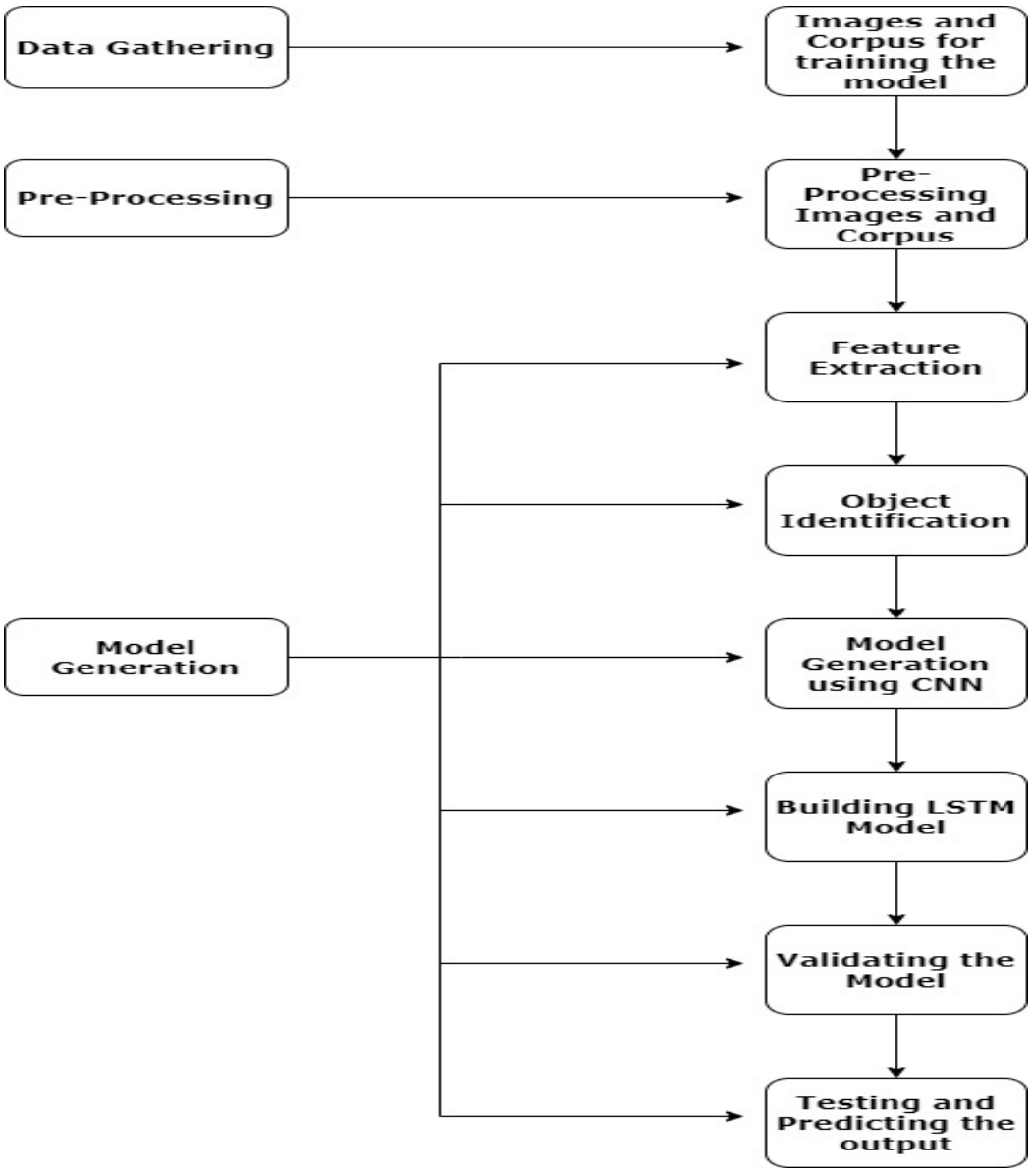

Fig 5.2 Process Flow Diagram

Above diagram explains about how the data is gathered, pre-processed and the model is generated.

Data Gathering involves in getting images and its associated captions. One image can have multiple captions also.

Initially, the corpus is pre-processed for removal of stop words, articles, punctuation marks and digits. This is done to shorten the training time and better performance of the model. A bag of words is generated which consists of mapping of images with the pre-processed Corpus. The file is saved in pickled format for training purpose.

The images have to be processed to identify the important objects in them and extract features from it. Later on, the extracted features need to be stored in a pickle file format for future use. Presence of objects in the images can be identified by finding the correlation between the X-axis.

After feature extraction with the help of CNN (Convolutional Neural Network) Model, the feature stored pickle file along with a pre-processed token of words taken as textual description from Corpus is passed to the LSTM (Long Short Term Memory) Algorithm for caption predicting of the images. This is possible because LSTM is a type of RNN (Recurrent Neural Network) Algorithm that is capable of learning order dependence in sequence prediction problems. Unlike feedforward neural algorithms, LSTM is a feedback connections algorithm. The advantage of this algorithm is that the algorithm keeps on learning from previous outputs and takes it as input. So the algorithm takes in previous output as present input, processes and learns it and gives out the output. Therefore, with the help of this algorithm, the model keeps on learning the captions for the images to be generated from the previously generated outputs and hence predicts the next caption for the image uploaded and generates the caption for the user.

5.3 Methodology

Following are the system of methods used for the project.

- **How Images are Pre-Processed**

CNN model could be used directly for building the caption generator model. Since the model is large and running each image through the 16 layers of neural layers of VGG16 algorithm is a tiresome process. Therefore, by running all the images through n-iterations of epochs, all the features from the images can be extracted and be stored in a pickle file format. These features can be loaded later for caption generation.

This optimization helps in faster training of model and lesser consumption of memory.

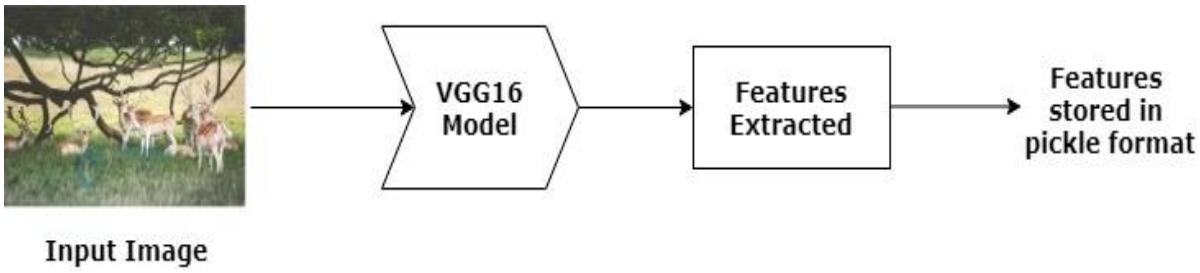

Fig 5.3 Image Pre-Processing steps

- **How Captions are Pre-Processed**

The Flickr8k (dataset) has multiple textual descriptions for each image and these descriptions require some minimum cleaning. Once the file containing all of the descriptions of the images, the objective is to find the unique identifier of each image. It is used on the image filename and file containing textual descriptions.

Once the list of photo descriptions are loaded, which are in dictionary format of key-value pair where image names are key and their multiple textual descriptions are the value. These need to be cleaned. Then the cleaned descriptions are tokenized so that they are easier to work with.

The text is cleansed in order to decrease the size of vocabulary words. This process is followed to convert all the words to lowercase, remove stop words, punctuation marks, articles, numbers etc which are unwanted in a description. This is done to summarize vocabulary size.

An ideal caption is one which is small and expressive.

Finally, the pickle file is generated which consists of all descriptions of the image along with the image filename where filename is the key and their description is the value as they are encoded in key value pair format.

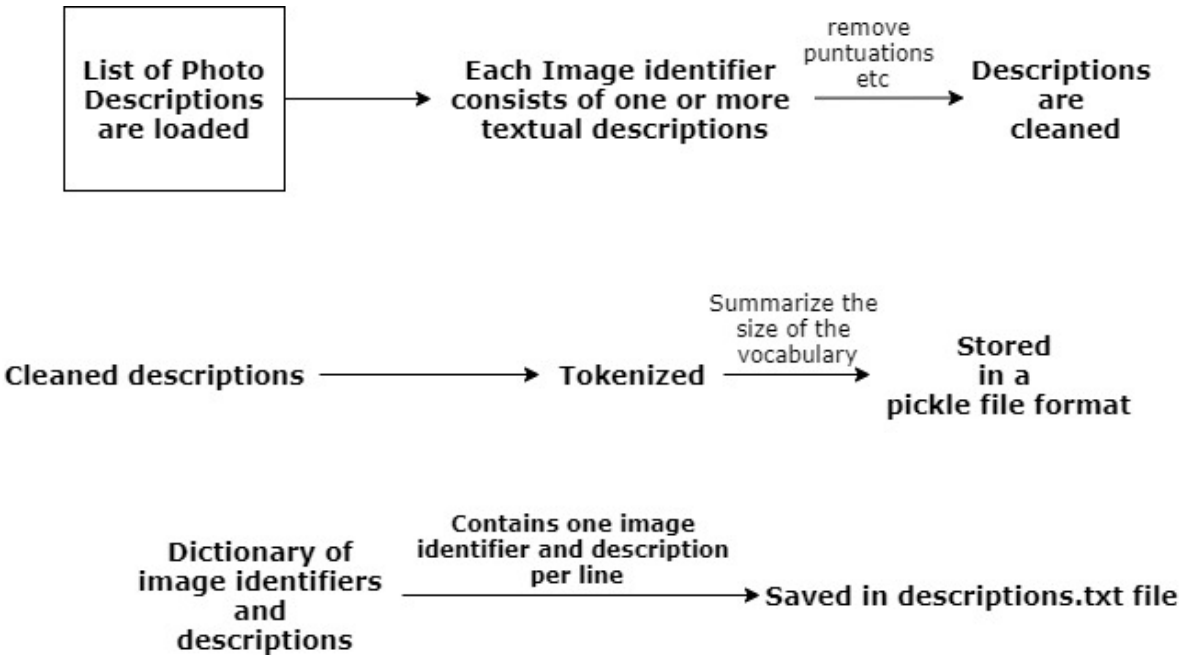

Fig 5.4 Caption Pre-Processing steps

- **CNN (Convolutional Neural Networks)**

CNN Algorithm – With the help of CNN Algorithm, features are extracted from the images with the help of pre-trained VGG16 model.

CNN Algorithm is being used in this project as they are specialised 16 layers of neural networks which can process the image in the 2D format only. CNN algorithm is the one which helps in converting the images from RGB format to greyscale (2D) format which helps in processing the images.

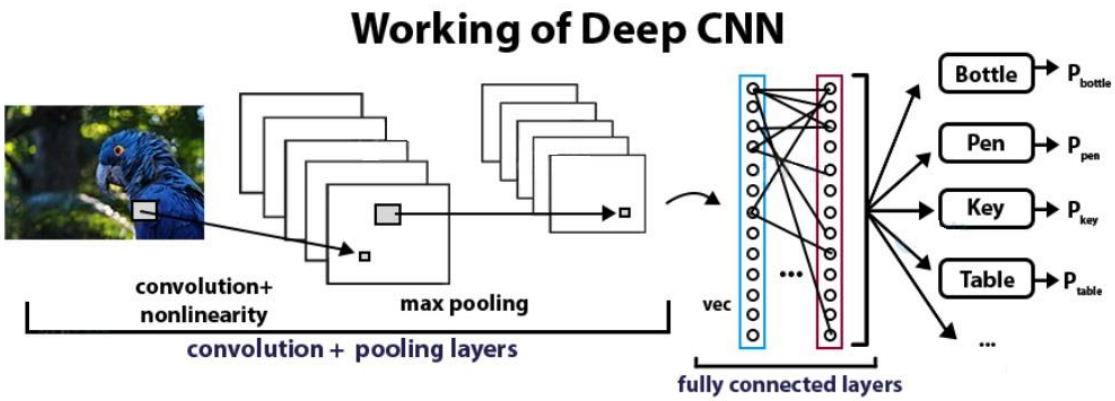

Fig 5.5 Working of CNN Algorithm

- **LSTM Cell Structure (Long Short Term Memory)**

LSTM – From the images with the help of CNN Algorithm, features are extracted and then caption is generated based on the important feature extracted from the images with the help of LSTM model.

LSTM is a feedback connections type of algorithm that is well suited for sequence prediction problems. With the help of LSTM, it can predict what the next word will be. The information generated from CNN will be used by LSTM to help in generating caption for the uploaded images.

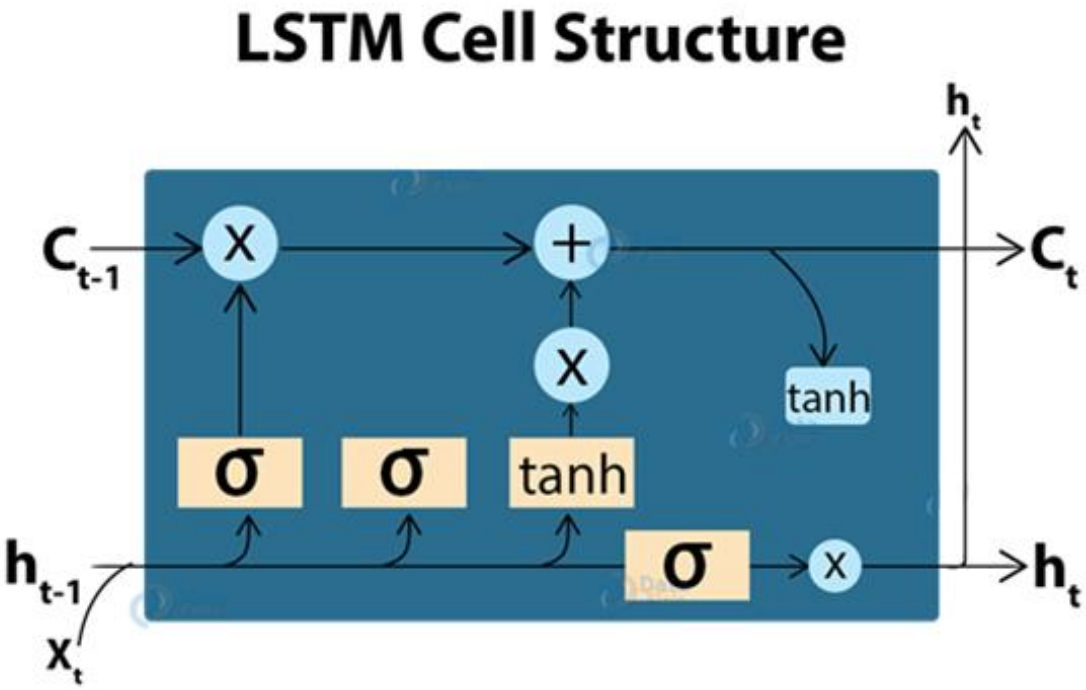

Fig 5.6 LSTM Cell Structure

- **Model Generation**

Once the images and corpus are pre-processed, the images are passed through Pretrained CNN model (VGG16 Model) which has been used in this project. With the help of VGG16 Model, features are stored in a pickle format file called ‘features.pkl’ once the features are extracted from it. Then with the help of LSTM Algorithm, ‘features.pkl’ and the pre-processed corpus of the images stored as ‘descriptions.txt’ along with the token of words stored in ‘tokenizer.pkl’ are passed through the LSTM Model which starts generating the final ML Model with the help of Epochs for generating captions for the images. Among the generated epochs of models, the best model is selected for development of application based on BLUE Score that helps in the evaluation of the translated text.

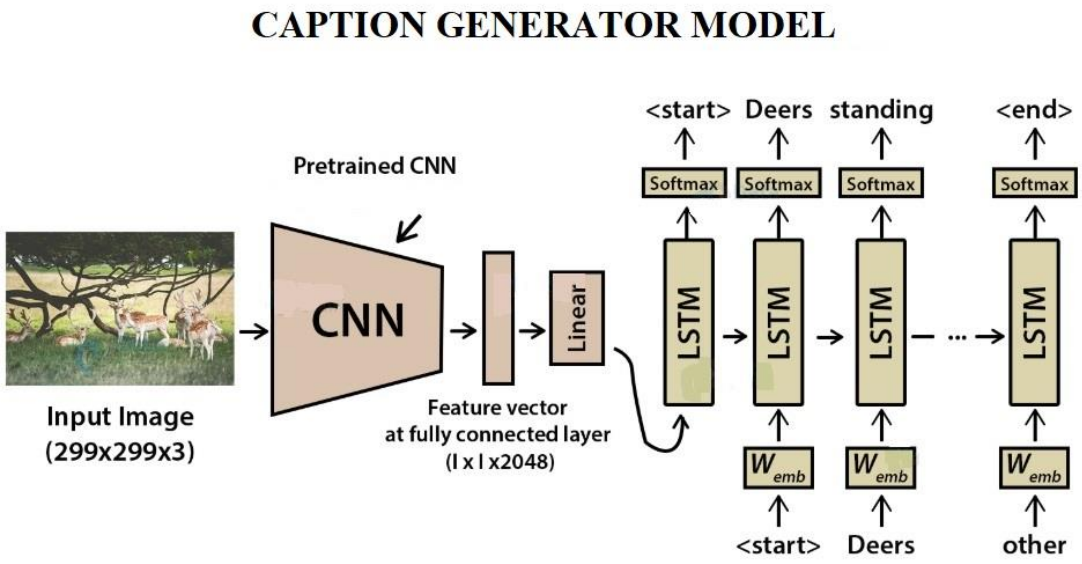

Fig 5.7 Working of Caption Generator

IMPLEMENTATION

6. IMPLEMENTATION

The implementation is the process of putting plan to effect. In software development, it is one of the important phase where the source code is written which forms as an instruction to the computer and make the plan to execute.

6.1 Source Code

The source code explained here is the snippet of some important functionalities of the project.

Configuring the GPU memory of training purposes

```
configuration = tf.ConfigProto()

configuration.gpu_options.per_process_gpu_memory_fraction     =
0.95

configuration.gpu_options.visible_device_list = "0"

set_session(tf.Session(config=config))


def set_seed (sd=144):

    from numpy.random import seed

    from tensorflow import set_random_seed

    import random as rn

    seed(sd)

    rn.seed(sd)

    set_random_seed(sd)
```

The above code is used for setting the GPU capacity to 95% else it will lead to overloading of GPU due to full consumption.

Plotting few images and their corpus from dataset

```
no_pic = 5

no_pix = 224

target_size = (nopix, nopix, 3)


count = 1

fig = plt.figure(figsize = (10, 20))

for jpgfnm in uni_filenames[-5:]:

    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()
```

The above code is used to plot images and their respective corpus from the dataset to find relevant unique caption for each image.

Applying Principle Component Analysis on the dataset

```
encoder = np.array(list(images.values()))

pca = PCA(n_components=2)

y_pca = pca.fit_transform(encoder)


picked_pic = OrderedDict()

picked_pic["red"]     = [2720,4250,4983,5862,4079]

picked_pic["green"]   = [2070,3784,7545,4644, 4997]

picked_pic["magenta"] = [6320,3432,1348,7472, 1518]

picked_pic["blue"]    = [3901,2168,3465,5285,5328]

picked_pic["yellow"]  = [144,1172,4423,4780,4448]

picked_pic["purple"]  = [5087]


fig, ax = plt.subplots(figsize=(15,15))

ax.scatter(y_pca[:,0],y_pca[:,1],c="white")


for irow in range(y_pca.shape[0]):

   ax.annotate(irow,y_pca[irow,:],color="black",alpha=0.5)

for color, irows in picked_pic.items():

   for irow in irows:
```

```
        ax.annotate(irow,y_pca[irow,:],color=color)
ax.set_xlabel("pca embedding 1",fontsize=30)
ax.set_ylabel("pca embedding 2",fontsize=30)
plt.show()


fig = plt.figure(figsize=(16,20))
count = 1
for color, irows in picked_pic.items():
    for ivec in irows:
        name = jpgs[ivec]
        filename = dir_Flickr_jpg + '/' + name
        image = load_img(filename, target_size=target_size)


        ax = fig.add_subplot(len(picked_pic),5,count,
                        xticks=[],yticks=[])
        count += 1
        plt.imshow(image)
        plt.title("{} ({})".format(ivec,color))
plt.show()
```

The above code is used to do Principle Component Analysis on the image dataset to reduce the dimensions of the features.

Pre-Processing Of Corpus

```
def load_descriptions(self, doc):
          mapping = dict()
          # process lines
          for line in doc.split('\n'):
               # split line by white space
               tokens = line.split()
               if len(line) < 2:
                    continue
# take the first token as the image id, the rest as the
description
               image_id, image_desc = tokens[0], tokens[1:]
               # remove filename from image id
               image_id = image_id.split('.')[0]
               # convert description tokens back to string
               image_desc = ' '.join(image_desc)
               # create the list if needed
               if image_id not in mapping:
                    mapping[image_id] = list()
               # store description
               mapping[image_id].append(image_desc)
          return mapping


def clean_descriptions(self, descriptions):
     # prepare translation table for removing punctuation
```

```
    table = str.maketrans('', '', string.punctuation)
    for key, desc_list in descriptions.items():
        for i in range(len(desc_list)):
            desc = desc_list[i]
            # tokenize
            desc = desc.split()
            # convert to lower case
            desc = [word.lower() for word in desc]
            # remove punctuation from each token
            desc = [w.translate(table) for w in desc]
            # remove hanging 's' and 'a'
            desc = [word for word in desc if len(word)>1]
            # remove tokens with numbers in them
        desc = [word for word in desc if word.isalpha()]
            # store as string
            desc_list[i] =  ' '.join(desc)
```

The above code is used to load the descriptions of all the images and remove punctuation, unwanted articles, remove hanging letters like ‘s’ and ‘a’, convert to lower case and remove tokens with numbers in them and store as a string.

Extracting vgg16_weights from keras

```
input_layer = Input(shape=(224, 224, 3))
model = VGG16(include_top=False, input_tensor=in_layer, pooling='avg')
print(model.summary())
```

The above code snippet is to extract the vgg16 model weights from the keras library for feature extraction.

Feature Extraction from images

```
def prepare_image_data(self):
          # Prepare the Image Data
          if os.path.exists('features.pkl'):
               print('Features already extracted into
\'features.plk\' file.')
          # re-structure the model
          model.layers.pop()
          model = Model(inputs=model.inputs)
          # summarize
          # print(model.summary())
          # extract features from each photo
          if os.path.isdir(source):
                    feature = self.extract_feature(filename,
model)
          elif os.path.isfile(source):
               return self.extract_feature(source, model)
          else:
               raise Exception("Source for images needs to be
a file or directory.")


     # extract a feature for a single photo
     def extract_feature(self, filename, model):
```

Above snippet is used to load all the images to VGG16 model and extract features from them and save the features in pickle file format.

Preparation of variables for caption generation

```
     prepare_image_data()
          prepare_text_data()
          training_features, training_desc =
     prepare_training_data()
          test_data_features, test_data_descriptions =
     load_test_data()
          tokenizer = prepare_tokenizer(training_desc)
          vocab_size = summarize_vocab(tokenizer)
          max_length = max_length_desc(training_desc)
     pretrained_model =
load_pretrained_model('D:/Study/Caption-
Generator/model_18.h5')
```

The above code snippet is used for storing the functions and dataset into variables for easier loading of data.

Caption Generation

```
def defining_model(vocabulary_size, maximum_length):
          input_image = Input(shape=(4096,))
          fimage1 = Dropout(0.5)(input_image)
          fimage2 = Dense(256, activation='relu')(fimage1)
          input_image_2 = Input(shape=(maximum_length,))
          sequence1 = Embedding(vocabulary_size, 256,
mask_zero=True)(input_image_2)
          sequence2 = Dropout(0.5)(sequence1)
          sequence3 = LSTM(256)(sequence2)
```

```
          decoder_1 = add([fimage2, sequence3])

          decoder_2 = Dense(256, activation='relu')(decoder_1)

          outputs_of_the_process = Dense(vocabulary_size,
activation='softmax')(decoder2)

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


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

          model_generated.summary()

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

          return model_generated
```

The above code snippet is used to generate the captions for the images by loading the pickle file which consists of all the features of the images extracted from the dataset.

Rendering of model into web application using flask

```
UPLOAD_FOLDER = os.path.join('D:', 'Study', 'Dataset',
'Flickr8k_Dataset', 'Flicker8k_Dataset')

    return redirect(url_for('get_caption',

                                    filename=filename))

    imgcptgen = ImageCaptionGenerator()


    model, tokenizer, max_length = imgcptgen.testing_params()

    caption = imgcptgen.test(model, tokenizer, max_length,
full_filename)
```

```
    return render_template("caption.html", captioned_image =
'Flicker8k_Dataset/' + filename, caption = caption)


if __name__ == '__main__':
    app.run()
```

The above code snippet helps in binding the ML model with HTML code to provide a User Interface to the users.

6.2 Screenshots

Home Screen

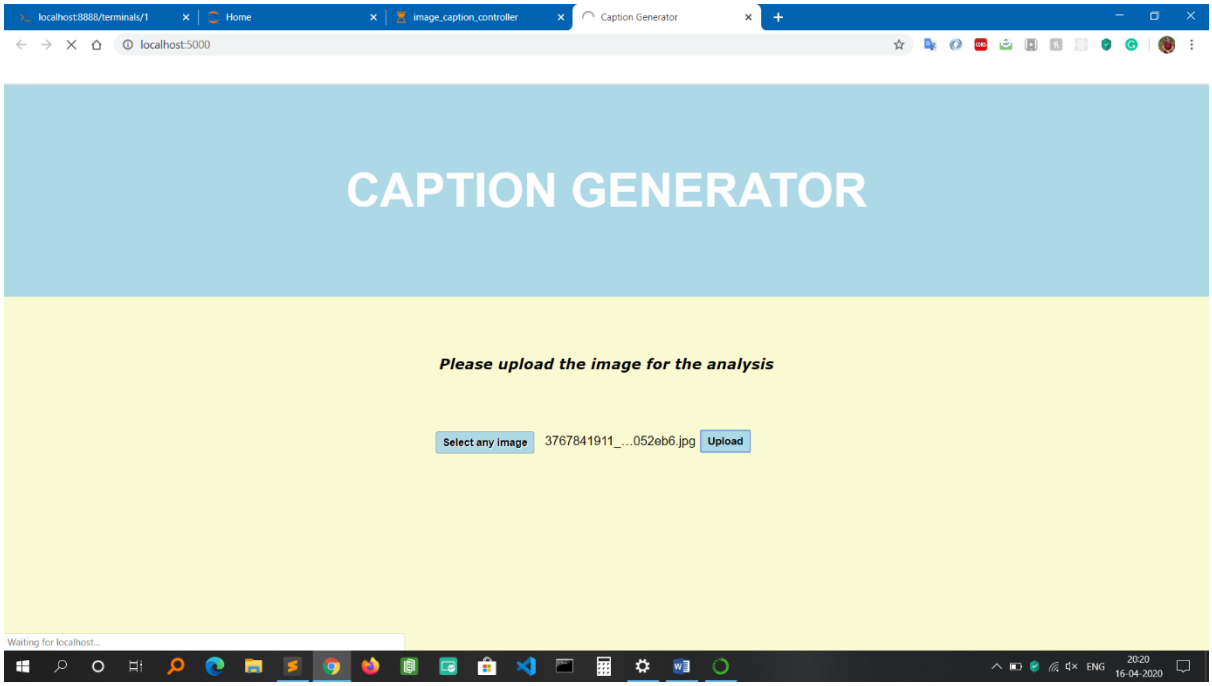

Fig 6.1 Home Screen

Above image tells shows the home page of the caption generator application.

User has to input a jpg, jpeg or png image files only as the application accepts the following formats only. Once the image has been selected and uploaded, the application will start processing the image and will generate the caption for the same.

In case the user accidentally enters wrong files such as non-image file format like .html, .txt etc the application won't process the file and will redirect back to the home page indicating the user that the person has selected the wrong file for generating caption.

Result Page

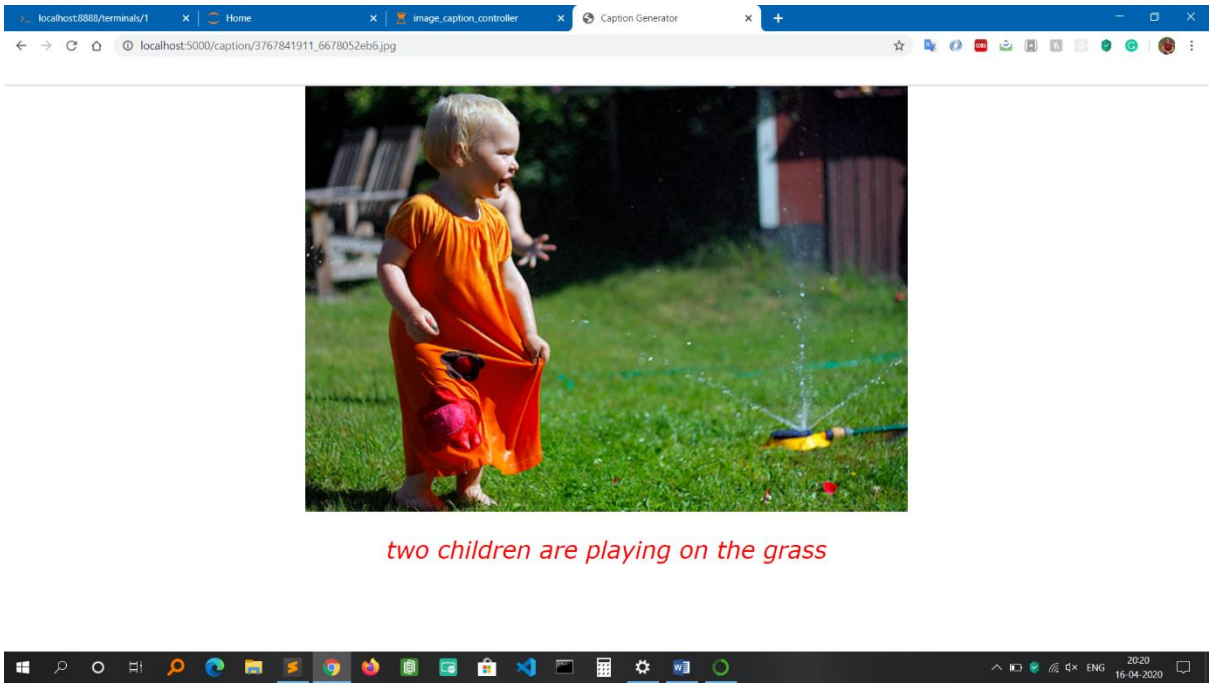

Fig 6.2 Result Page Screen

Above image shows the result once the image has been uploaded by the user. The result consists of the image along with the caption generated by the ML model. Caption has been successfully generated for the above image only when image formats like jpg, jpeg and png has been successfully uploaded in the application.

MODEL EVALUATION AND PERFORMANCE

7. MODEL EVALUATION AND PERFORMANCE

As all the machine-learning applications have two major parts, one is training for the model generation and another is testing on the generated model. The model is trained with 6000 images. Once the model is generated, the model is evaluated using BLEU Score for the captions it has generated.

7.1 Model Testing

- **BLEU Score**

BLEU Score (Bilingual Evaluation Understudy) helps in comparing quality of text which has been translated by the machine from one language to another. Here the language used is English. It is mainly used to know the prediction accuracy whether the caption generated for the image matches with the textual descriptions in the corpus.

```
Dataset: 6000
Descriptions: train=6000
Vocabulary Size: 7579
Description Length: 34
Dataset: 1000
Descriptions: test=1000
Photos: test=1000
BLEU-1: 0.516172
BLEU-2: 0.272307
BLEU-3: 0.186117
BLEU-4: 0.087105
```

Fig 7.1 BLEU Score for Model Evaluation

50 % and above is considered as a very good score.

BLEU Score allows to get the different n grams weights specified for the calculation.

Very helpful in calculating different BLEU Score like individual and cumulative ngram scores such as single gram (one-gram) or word pairs (bigram).

7.2 Manual Test Cases

Table 7.1 Valid Input Test Case (a)

Test Case Description	Testing the functionality of uploading the image for analysis		
Date Of Testing	16-April-2020	**Test Case Status**	Pass

Serial no	**Pre-requisites:**	**Serial no**	**Test Data**
1	Access to Chrome Browser	1	Choose file = Upload images of any format like jpg, jpeg, png

Test Scenario	Verify on the ability to upload the images of any format for analysis after selecting the desired image

Step no	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Navigate to http://localhost:5000/	Caption generator homepage should open	Successfully redirected	Pass
2	Click on Choose File to select the image for uploading	File Explorer should open for selecting the image	File Explorer Window successfully opened	Pass
3	Select the desired image	Should display the name of the image selected on the home page of the application	The image name is being displayed on the home page of the application	Pass
4	Click on Upload button to upload the image for analysis	ML processing should begin in the background and should redirect to the result page for displaying the caption with the respective image	Page redirected and caption generated for the uploaded image.	Pass

Table 7.2 Valid Input Test Case (b)

Test Case Description	Testing the functionality by trying to upload non-image file		
Date Of Testing	16-April-2020	**Test Case Status**	Pass

Serial no	**Pre-requisites:**	**Serial no**	**Test Data**
1	Access to Chrome Browser	1	Choose file = .txt

Test Scenario	Verify on the ability to upload non-image files like .txt

Step no	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Navigate to http://localhost:5000/	Caption generator homepage should open	Successfully redirected	Pass
2	Click on Choose File to select non-image file like .txt	File Explorer should open for selecting the file	File Explorer Window successfully opened	Pass
3	Select the txt file	Should display the name of the file selected on the home page of the application	The file name is being displayed on the home page of the application	Pass
4	Click on Upload button to upload the image for analysis	Page should not redirect because the application accepts only jpg, jpeg and png format files.	Page not redirected because format not supported	Pass

Table 7.3 Invalid Input Test Case (a)

Test Case Description	Testing the functionality of prompting the user to upload image file format only.		
Date of Testing	16-April-2020	**Test Case Status**	Fail

Serial no	**Pre-requisites:**	**Serial no**	**Test Data**
1	Access to Chrome Browser	1	Choose file = .html
		2	Click on the upload button

Test Scenario	Verify on whether user gets a prompt informing to upload only image file format

Step no	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Navigate to http://localhost:5000/	Caption generator homepage should open	Successfully redirected	Pass
2	Click on Choose File to select non-image file like .html	File Explorer should open for selecting the file	File Explorer Window successfully opened	Pass
3	Select any .html file	Should display the name of the file selected on the home page of the application	File name is displayed on the home page of the application	Pass
4	Click on Upload button to upload the file for analysis	Application should not process the file and should prompt a message saying file type not allowed	No prompt message displayed	Fail

Table 7.4 Invalid Input Test Case (b)

Test Case Description	Testing the functionality of uploading the image for analysis		
Date of Testing	16-April-2020	**Test Case Status**	Pass

Serial no	**Pre-requisites:**	**Serial no**	**Test Data**
1	Access to Chrome Browser	1	Click the upload button

Test Scenario	Verify on whether the home page gets redirected to result page without selecting any image

Step no	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Navigate to http://localhost:5000/	Caption generator homepage should open	Successfully redirected	Pass
2	Click on Upload button to upload the image for analysis	The Validator should display a message saying 'Please select a file' as no image is selected for uploading	Page not redirected and Validator displays a message saying 'Please select a file'	Pass

RESULTS AND DISCUSSION

8. RESULTS AND DISCUSSION

8.1 Correct Classification of Caption Generation

Fig 8.1 Black and white dog is running through the grass

The image above gives a precise description of the image during the testing of the model with one of the image.

Fig 8.2 Two children are playing on the grass

The image above gives a precise description of the image during the testing of the model with one of the image.

Fig 8.3 Man in red shirt is standing on the edge of the water

The image above gives a precise description of the image during the testing of the model with one of the image.

8.2 Misclassified Caption Generation

Fig 8.4 Wrong generation of caption

Above image tells about the caption generated wrongly during testing of the trained model on real-time images.

8.3 Discussion

The reason behind the wrong generation of the caption is that the accuracy of the model is 52%. This is because the dataset had only 8092 images with which it was able to achieve this accuracy. According to statistics, Flickr30k or MSCOCO dataset with 30000 images was able to achieve 76% accuracy. Therefore, with 8000 images of the dataset, the model has achieved good accuracy.

CONCLUSION

9. CONCLUSION

The objective of the project is to identify the action portrayed in the given image. Almost 52% of the time the application gives the correct output. The generated caption describes the image in the same way as out human brain recognizes any visual. The caption generator is a useful application for visually impaired people as they might not know what kind of image is it and the kind of action taking place in it. Fine Tuning this application could ease their work. The application also helps in visual media for auto-subtitling. Besides, the whole application has been saved in the GitHub repository. So in the future, if the user requests for any changes, it can be easily done through git version control.

FUTURE ENHANCEMENTS

10. FUTURE ENHANCEMENTS

The project opens up the scope for further enhancements and that are listed as below.

- Distorted and blurred images can also be considered.
- Caption Generation should work for video also.
- A Customized Mobile Application can be developed based on the user requirements.
- With the help of some hybrid algorithms in the future, it will be possible to achieve a higher accuracy that will help in better generation of captions for all the images.

APPENDIX

APPENDIX A: BIBLIOGRAPHY

Book References

- Jason Brownlee (2007), Deep Learning for Natural Language, Edition v1.1

Paper References

- Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, Barbara Plank, Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures, Retrieved from journal of Artificial Intelligence Research submitted on 15th Jan 2016 - https://arxiv.org/abs/1601.03896

- Jiuxiang Gu, Gang Wang, Jianfei Cai, Tsuhan Chen, An Empirical Study of Language CNN for Image Captioning, Retrieved from 2017 IEEE International Conference on Computer Vision - https://ieeexplore.ieee.org/

- MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, Hamid Laga, A Comprehensive survey of Deep Learning for Image Captioning, Retrieved from journal on Deep Learning for Image Captioning submitted on 14th October 2018 - https://arxiv.org/pdf/1810.04020.pdf

- Kenneth P. Camilleri Marc Tanti, Albert Gatt, What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?, Retrieved from Journal on Computer Vision and Pattern Recognition submitted on 7th August 2017 - https://arxiv.org/abs/1708.02043

Web References

- Flickr8k Dataset - https://www.kaggle.com/flickr8k
- Sumit Saha - A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way - https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

- Rohit Thakur - Step by step VGG16 implementation in Keras for beginners - https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c
- Jason Brownee – Deep Learning for Natural Language Processing - https://machinelearningmastery.com/deep-learning-for-nlp/
- Shuang Bai – A survey on automatic image caption generation - https://www.sciencedirect.com/science/article/abs/pii/S0925231218306659?via%3Dihub#!
- Anonymous (2018) – VGG16 – Convolutional Network for Classification and Detection - https://neurohive.io/en/popular-networks/vgg16/

Appendix B: USER MANUAL

Step 1: The user must first launch the application.

Step 2: Select an image file.

Step 3: Click on the submit button.

Step 4: Within a matter of seconds, the ML model recognizes and process the content of an image.

Step 5: The result will be displayed on the next page.