INTRODUCTION

1. INTRODUCTION

A Human being can see an image and your brain can easily tell what the image is about, but a computer cannot tell what the image is representing. Ability to automatically describe the content of an image using properly formed English sentences is a very fundamental and challenging task. However, it could have greater potential impact, for instance by helping visually impaired people had better understand the content of the images on the web or locally saved in their system. It could also provide more accurate and compact information of images in scenarios such as image sharing in social network or video surveillance systems. Moreover, a description must capture not only the objects contained in an image, but it must also express how these objects relate to each other as well as their attributes and the activities they are involved in.

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 help visually impaired people to make them better understand the content of the images, Auto-subtitling.

1.3 SCOPE

The application will 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 that helps in 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 the scientific study of algorithms and statistical models that the computer system uses to do a specific task without using explicit instructions. With the advancement in Deep learning techniques and availability of huge datasets and computer power, it is possible to build models that can generate captions for an image. These algorithms can build a mathematical model for sample data which is known as training data, then uses to achieve some objective that is done on 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 features are extracted from the images 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 corpus is pre-processed to remove stop words, punctuation marks and digits from the textual description?
2. How objects are identified from the images?
3. How features are extracted from the images like actions performed by the person or animal in the images etc?

Any image will have 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 once 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. 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.

After importing images from the dataset, consequently all 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. “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.

Various CNN algorithms can be used for feature extraction. Some of the algorithms are VGG16, VGG19, InceptionV3. Feature extraction was done in this project with the help of VGG16 model. The model achieved 92% accuracy with 14 million images dataset.

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.

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 it is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. The system is considered feasible when the positives nominate the negatives.

- **TECHNICAL FEASIBILITY**

  The 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. Hence 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. Workforce 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. So 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. The paper compares these two architectures. 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 open-source distribution of the Python programming language and R programming languages for 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**

  An open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Following are the uses: Data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning and much more.

- **VS Code**

  It is an editor developed by Microsoft for Windows, Linux and macOS. Benefit of this application is that it provides support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets and code refactoring.

- **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 for tracking changes in source code during software development. 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 speed, data integrity and support for distributed, non-linear workflows.

3.3.2 TECHNOLOGIES

- **Python 3.7**

  Python is an expressive, extensible, cross-platform, free, open-source and high-level programming language which is being used for general-purpose programming. Python 3.7 is an interpreted language that provides easy integration. Python supported Flask framework even helps to integrate the code with HTML5.

- **Flask**

  It is a micro web framework written in Python classified as a microframework because it does not require particular tools or libraries. Also it has no database abstraction layer, form validation or any other components where pre-existing third-party libraries provide common functions.

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

  It is the standard mark-up language for documents designed to be displayed in a web browser that can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

- **CSS (Cascading Style Sheets)**

  It is a style sheet language used for describing the presentation of a document written in a mark-up language like HTML. It is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

- **Bootstrap**

  It is a free and open-source CSS framework directed at responsive, mobile-first front-end web development that contains CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

- **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 TensorFlow, Microsoft Cognitive Toolkit, R, Theano or PlaidML. It is designed to enable fast experimentation with deep neural networks as it focuses on being user-friendly, modular and extensible.

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. Since normal human beings can see and tell what the image is representing, visually impaired people can be major users.

4.2 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the system

Model Development and Deployment

- **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.

- **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

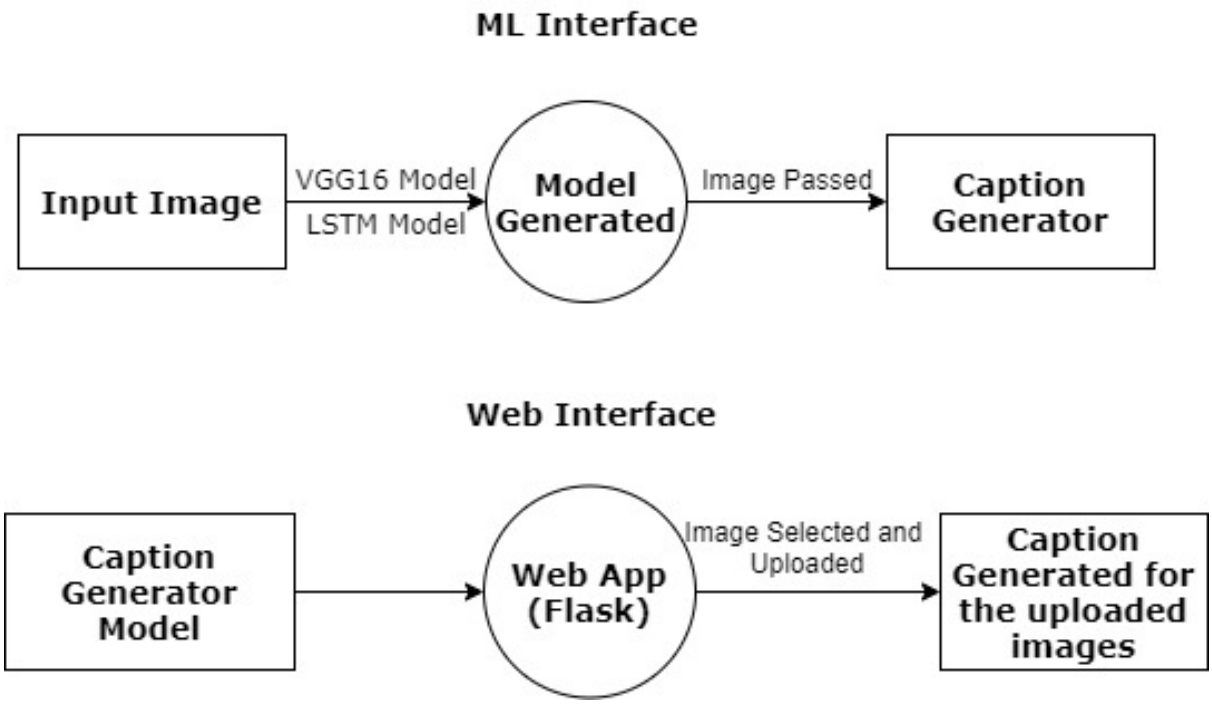

Fig 5.1 DFD Level 0

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

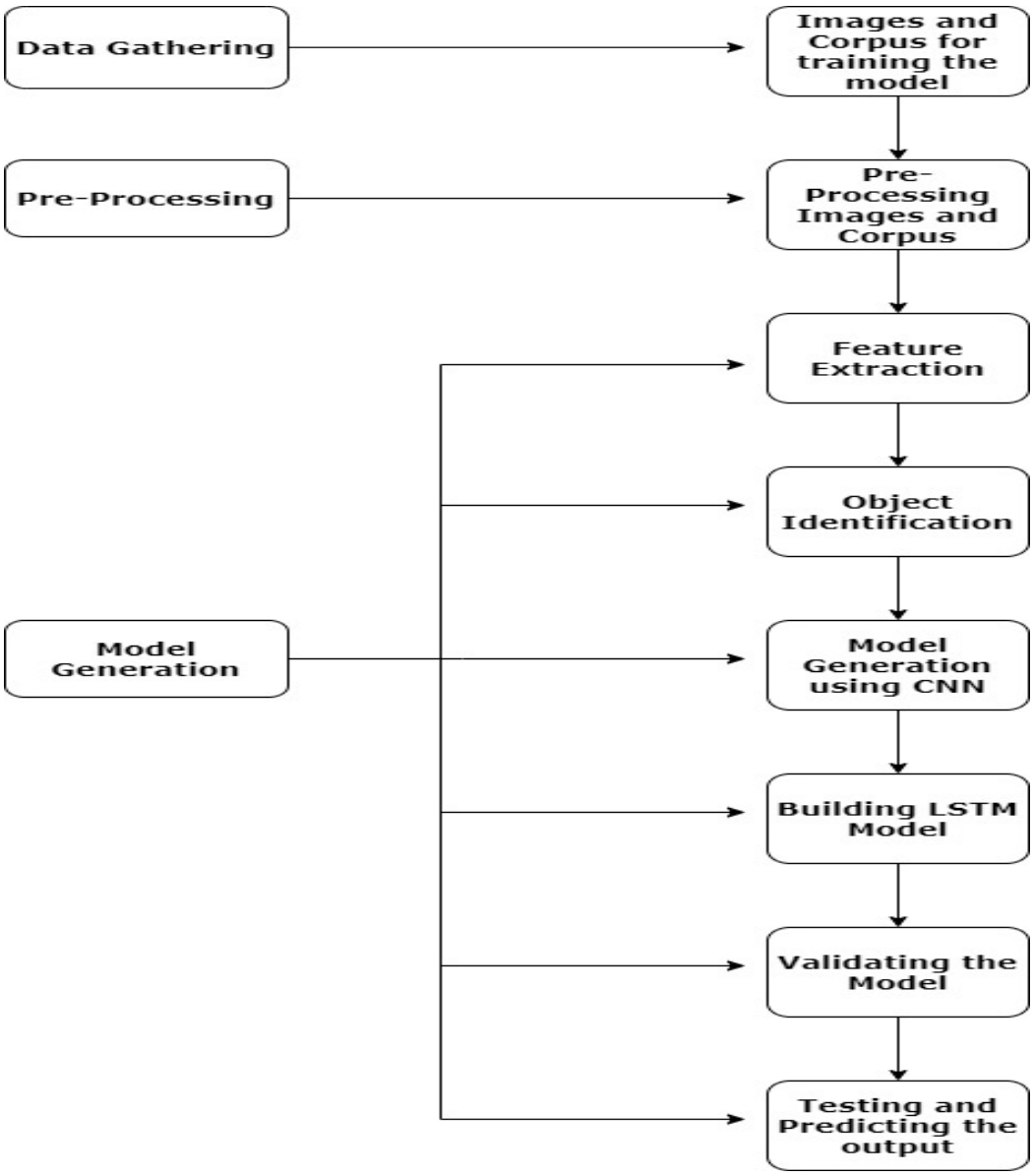

Fig 5.2 Process Flow Diagram

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

5.3 METHODOLOGY

Following are the system of methods used for the project.

- **How Images are Pre-Processed**

CNN model could be used directly as part of a broader image caption model. The problem is, it is a large model and running each photo through the network every time we want to test a new language model configuration is redundant. Instead, we can pre-compute the “photo features” using the pre-trained model and save them to file. We can load these features later and feed them into our model as the interpretation of a given photo in the dataset. It is no different from running the photo through the full VGG model.

This optimization will make training our models faster and consume less memory.

Once features are extracted, they are stored in pickle format as a file for future use.

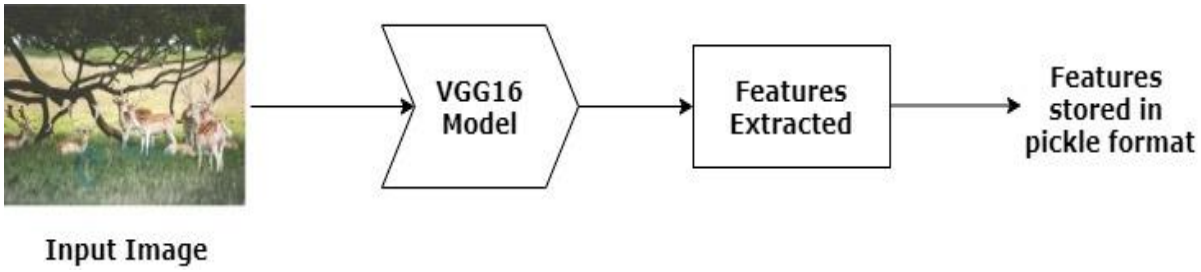

Fig 5.3 How Images are Pre-Processed

- **How Captions are Pre-Processed**

The Flickr8k dataset contains multiple descriptions for each photograph and the text of the descriptions requires some minimal cleaning. Once the file containing all of the descriptions of the images, the objective is to find the unique identifier of each image. This identifier is used on the photo file name and in the text file of descriptions.

Once the list of photo descriptions are loaded, it will return a dictionary of photo identifiers to descriptions. Every photo identifier maps to a list of one or more textual descriptions. Then the descriptions are needed to be cleaned. Then the cleaned descriptions are tokenized so that they are easier to work with.

The text is cleansed to reduce the size of the vocabulary of words. It is done to convert all words to lowercase, remove all punctuations, remove all words that are one character or less in length and remove all words with numbers in them. Once cleaned, we can summarize the size of the vocabulary.

Ideally, a small and expressive vocabulary is expected.

Finally, the dictionary of image identifiers and descriptions are saved to a new file called descriptions.txt with one image identifier and description per line.

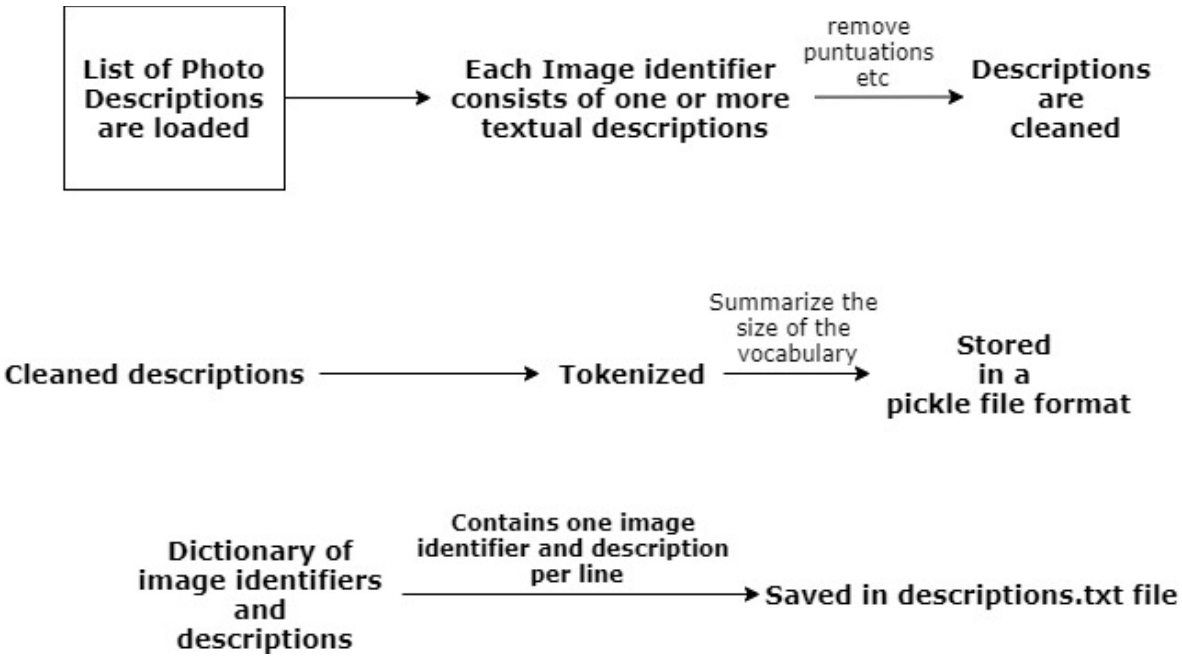

Fig 5.4 How Captions are Pre-Processed

- **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 specialized deep neural networks which can process the data that has input shape like a 2D matrix. The images are easily represented as a 2D matrix and CNN is very useful in working with images.

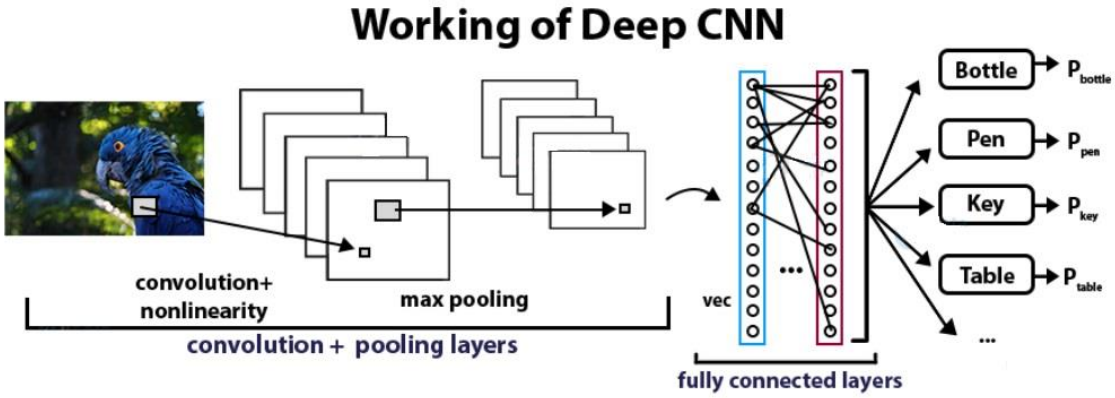

Fig 5.5 Working of CNN Algorithm

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

LSTM Algorithm – With the features being extracted from the images with the help of CNN Algorithm, they are now fed into the LSTM model that will be responsible for generating the image captions.

LSTM (Long Short Term Memory) is a type of RNN that is well suited for sequence prediction problems. With the help of LSTM, it can predict what the next word will be. LSTM will use the information from CNN to help generate a description of the image.

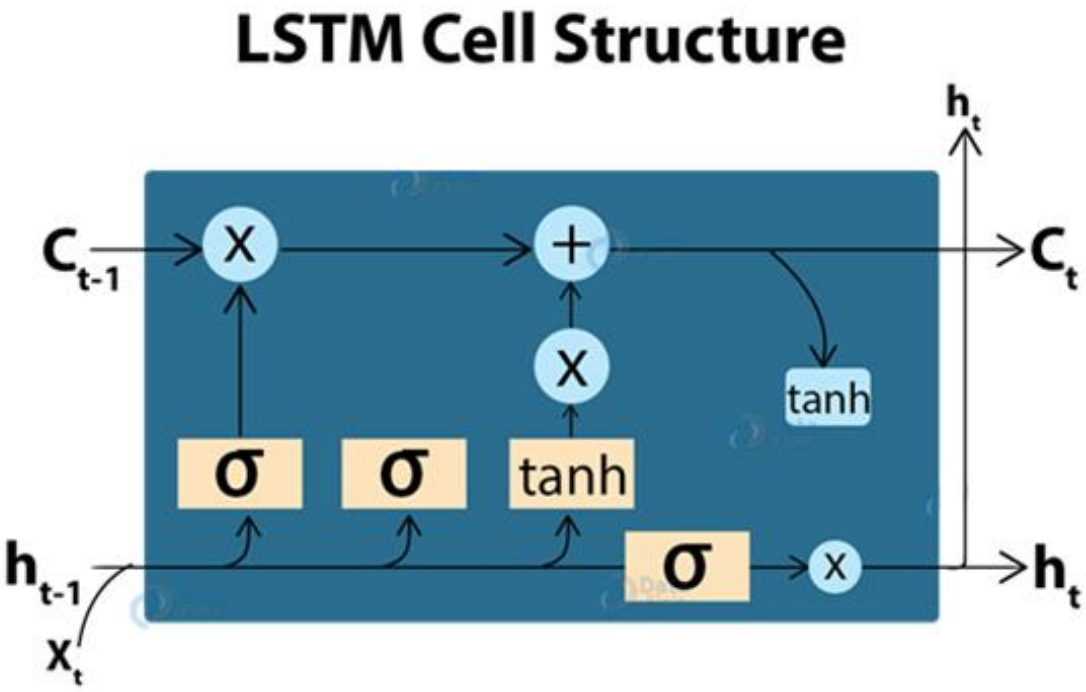

Fig 5.6 LSTM Cell Structure

- **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 extracted from the images and stored in a pickle format file called 'features.pkl'. 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.

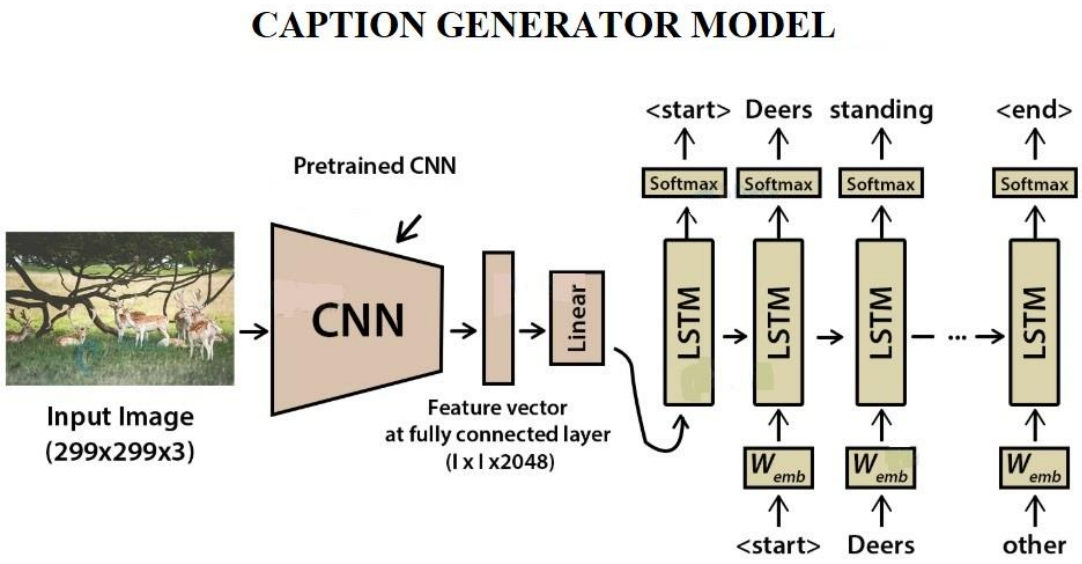

Fig 5.7 Working of Caption Generator

IMPLEMENTATION

6. IMPLEMENTATION

6.1 SOURCE CODE

Source code is the one that tells about what are all the important functions used and what are all the important steps in the project.

- **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.

- **LOADING ALL IMAGES FROM DATASET AND EXTRACTING FEATURES FROM THEM**

```
def prepare_image_data(self):
          # Prepare the Image Data
          # extract features from all images
          directory =
'D:/Study/Dataset/Flickr8k_Dataset/Flicker8k_Dataset'
          if os.path.exists('features.pkl'):
               print('Features already extracted into
\'features.plk\' file.')
          else:
               features = self.extract_features(directory)
               print('Extracted Features: %d' % len(features))
               # save to file
               dump(features, open('features.pkl', 'wb'))
def extract_features(self, source):
          # load the model
          model = VGG16()
          print('Using %s model to extract features ...' %
model.name)
          # re-structure the model
          model.layers.pop()
          model = Model(inputs=model.inputs,
outputs=model.layers[-1].output)
          # summarize
          # print(model.summary())
          # extract features from each photo
```

```
            if os.path.isdir(source):
                features = dict()
                for name in listdir(source):
                    # load an image from file
                    filename = source + '/' + name
                    feature = self.extract_feature(filename,
model)
                    # get image id
                    image_id = name.split('.')[0]
                    # store feature
                    features[image_id] = feature
                    print('>%s' % name)
                return features
            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):
            # load the photo
            image = load_img(filename, target_size=(224, 224))
            # convert the image pixels to a numpy array
            image = img_to_array(image)
            # reshape data for the model
```

```
            image = image.reshape((1, image.shape[0],
image.shape[1], image.shape[2]))
            # prepare the image for the VGG model
            image = preprocess_input(image)
            # get features
            feature = model.predict(image, verbose=0)
            return feature
```

The 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.

- **DEFINE THE CAPTIONING MODEL**

```
def define_model(self, vocab_size, max_length):
            inputs1 = Input(shape=(4096,))
            fe1 = Dropout(0.5)(inputs1)
            fe2 = Dense(256, activation='relu')(fe1)
            inputs2 = Input(shape=(max_length,))
            se1 = Embedding(vocab_size, 256,
mask_zero=True)(inputs2)
            se2 = Dropout(0.5)(se1)
            se3 = LSTM(256)(se2)
            decoder1 = add([fe2, se3])
            decoder2 = Dense(256, activation='relu')(decoder1)
            outputs = Dense(vocab_size,
activation='softmax')(decoder2)
            model = Model(inputs=[inputs1, inputs2],
outputs=outputs)
            model.compile(loss='categorical_crossentropy',
optimizer='adam')
```

```
          model.summary()
          plot_model(model, to_file='model.png',
show_shapes=True)
          return model
```

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.

- **Index.html**

```
<!DOCTYPE html>
<html>
<head>
     <title>Caption Generator</title>
</head>
<div align="center">
<form method=post enctype=multipart/form-data>
     <body bgcolor="LightGoldenRodYellow" id="body">
          <div class="header">
               <h1>CAPTION GENERATOR</h1>
          </div>
          <br>
          <br>
          <br>
          <h3 class="info">Please upload the image for the
analysis</h3>
          <br>
          <br>
          <br>
          <table>
```

```
                <tr>
                    <td><input type="file" name="file"
class="custom-file-input" required></td></div>
                    <td><input type="submit" value="Upload"
class="custom-file-upload"></td>
                </tr>
            </table>
            <br>
            <br>
            <p id="display"></p>
</body>
</form>
</div>
</html>
```

The above code snippet is used to build the home screen of the application which will allow the users to upload the image for generating caption.

- **Caption.html**

```
<html>
    <head>
        <title>Caption Generator</title>
    </head>
    <body>
        <img src="{{ url_for('static', filename=
captioned_image) }}" alt="Captioned Image" class="caption-
img">
        <p class="caption-txt" align="center">{{ caption
}}</p>
```

```
    </body>
</html>
```

The above code snippet helps in displaying the image with their caption generated once the ML model has processed them.

- **Image_caption_controller.py**

```
UPLOAD_FOLDER = os.path.join('D:', 'Study', 'Dataset',
'Flickr8k_Dataset', 'Flicker8k_Dataset')
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS
@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
```

```
            return redirect(url_for('get_caption',
                                    filename=filename))
    return render_template("index.html")
@app.route('/caption/<path:filename>', methods=['GET'])
def get_caption(filename):
    imgcptgen = ImageCaptionGenerator()
    full_filename = os.path.join(app.config['UPLOAD_FOLDER'],
filename)
    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

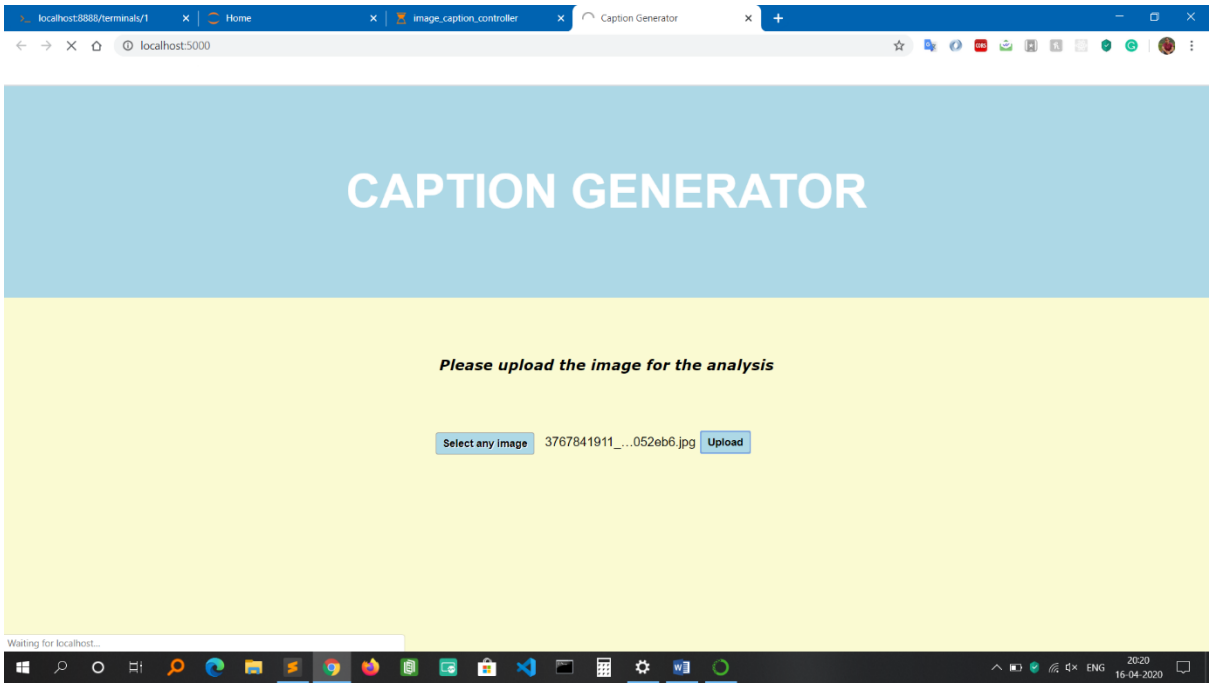

Fig 6.1 Home Screen

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

RESULT PAGE

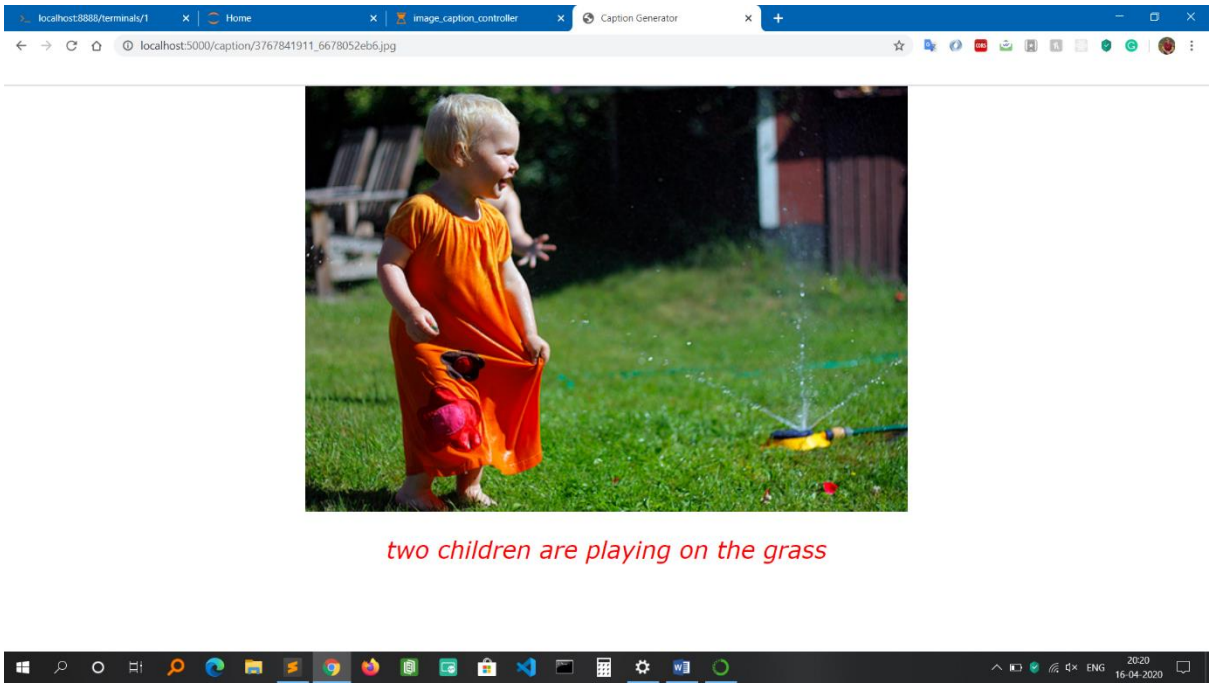

Fig 6.2 Result Page Screen

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.

HOME SCREEN 2

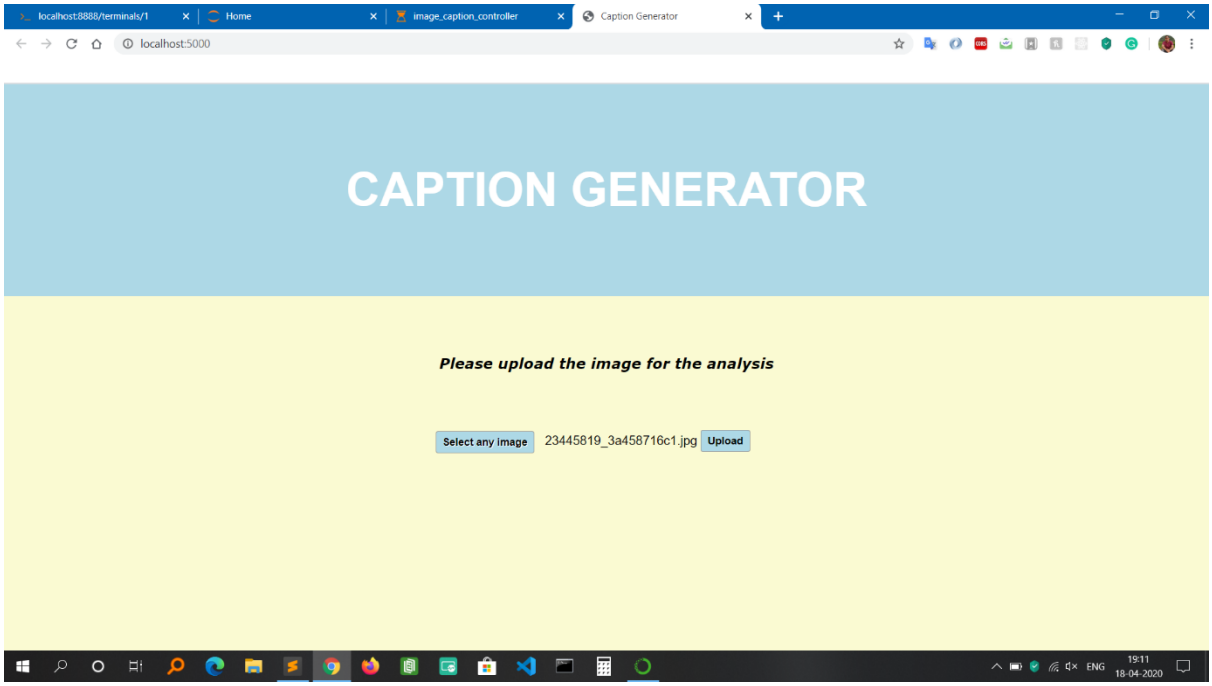

Fig 6.3 Home Screen 2

RESULT PAGE 2

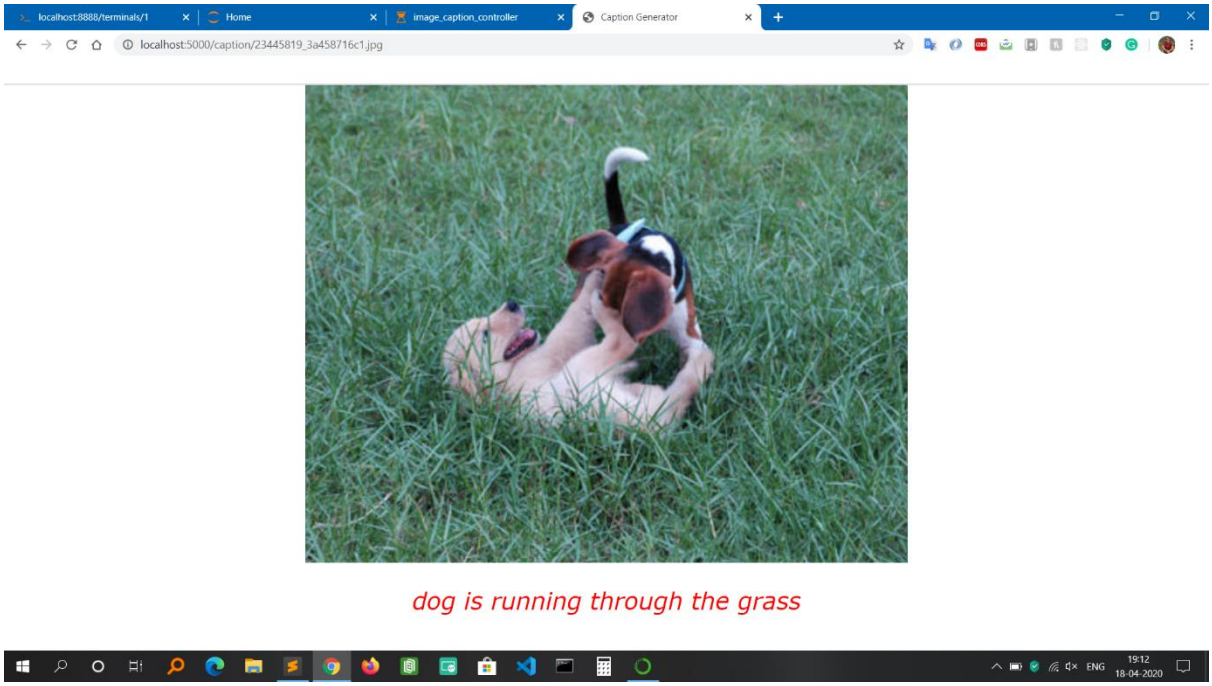

Fig 6.4 Result Page Screen 2

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. This application is done using 8092 different images. Once the model is generated, to find the accuracy of the caption, BLEU Score was used.

7.1 MODEL TESTING

- **BLEU SCORE**

Bilingual Evaluation Understudy (BLEU) is a score for comparing a candidate translation of the text to one or more reference translations. It is a metric for evaluating a generated sentence to a reference sentence.

```
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

If the score is 50 % and above, it is a very good score.

The BLEU score calculations in NLTK allow you to specify the weighting of different n-grams in the calculation of the BLEU score.

The BLEU score calculations in NLTK allow you to specify the weighting of different n-grams in the calculation of the BLEU score. Helps in the ability to calculate different types of BLEU score, such as individual and cumulative n-gram scores.

An individual N-gram score is an evaluation of just matching grams of a specific order, such as single words (1-gram) or word pairs (2-gram or bigram).

7.2 MANUAL TEST CASES

Table 7.1 Valid Input Test Case

Test Case ID	PES_001	**Test Case Description**	Test the Functionality of uploading the image for analysis		
Created By		**Reviewed By**		**Version**	1
QA Tester's Log					
Tester's Name		**Date Tested**	16-April-2020	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**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 #	**Step Details**	**Expected**	**Actual Results**	**Pass / Fail / Not**
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

Test Case ID	PES_002	**Test Case Description**	Test the Functionality by trying to upload non-image file		
Created By		**Reviewed By**		**Version**	1
QA Tester's Log					
Tester's Name		**Date Tested**	16-April-2020	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**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 #	**Step Details**	**Expected**	**Actual Results**	**Pass / Fail / Not**
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

Test Case ID	PES_003	**Test Case Description**	Test the Functionality of uploading the image for analysis without selecting any image		
Created By		**Reviewed By**		**Version**	1
QA Tester's Log					
Tester's Name		**Date Tested**	16-April-2020	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**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 #	**Step Details**	**Expected**	**Actual Results**	**Pass / Fail / Not**
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

RESULT AND DISCUSSION

8. RESULT AND DISCUSSION

8.1 RESULTS

Fig 8.1 Black and white dog is running through the grass

Above image 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

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

Fig 8.3 Dog is running through the grass

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

8.2 RESULTS OF WRONG GENERATION OF CAPTIONS

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 was to identify the action portrayed in the given image. Almost 52% of the time the project gives the correct output. The generated caption will describe the image that will say what kind of actions is taking place in it. The caption from the image will be helpful for visually impaired people as they will know what kind of image is it and the kind of action taking place in it. This has been solved with my application as it provides a service to users that helps in generating captions for any images of their choice once they upload it in my application. This study also helps in visual media as this application could be used 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 Mobile Application can be developed that will make the user more convenient to use.
- In the future, some hybrid algorithms can be used to achieve a higher accuracy that will help in better generation of captions for the images.