Image Caption Generator using deep learning with Flickr Dataset

[1]Mr. Dhirendra Parate, [2]Mrs. Minu Choudhary

[1]M.Tech Scholar, Rungta College of Engineering & Technology, Bhilai, Chhattisgarh, India,
2 Assistant Professor, Department of Computer Science, Rungta College of Engineering & Technology, Bhilai, India

***Abstract*: The Image Caption Generator is a process of identifying the context of an image and annotating it with appropriate captions using deep learning and the computer. To create an image caption generator that could load a random image and output a few captions that described it. Long Short-Term Memory Network (LSTM) and Convolutional Neural Network (CNN) will be used for Natural Language Processing and picture feature extraction, respectively (NLP).**

***Index Terms*: Image Captioning, deep learning, Convolutional Neural Network, Natural Language Processing, Long Short-Term Memory Network (LSTM)**

I. INTRODUCTION

Computer vision in the field of image processing has significantly advanced in recent years, including image classification and object recognition. The issue of image captioning, which involves automatically generating one or more phrases to comprehend an image's visual information, has benefited from advancements in image categorization and object detection. Automatically creating thorough and natural image descriptions offers a wide range of possible applications, including adding titles to news photos, adding descriptions to medical images, text-based image retrieval, information access for blind users, and human-robot interaction. These captioning-related applications have significant theoretical and real-world research significance. In the age of artificial intelligence, image captioning is a trickier but important work. An image captioning algorithm should produce a semantic description of a new image when given one. Based on an image we've provided or uploaded, this tool generates captions for the pictures. A trained model that has been trained with algorithms and a sizable dataset will produce the caption. When we utilize or apply it on social media or on other applications, the fundamental concept is that consumers will receive automated captions.

II. LITERATURE SURVEY

A written description must be provided for a given image as part of the difficult artificial intelligence challenge known as caption creation. It takes both approaches from computer vision to understand the content of the image and a language model from the field of natural language processing to transfer the comprehension of the image into words in the appropriate order. On applications of this problem, deep learning techniques recently produced state-of-the-art results.

Deep learning techniques have delivered cutting-edge outcomes for caption generating issues. The most amazing aspect of these methods is that, rather than requiring complex data preparation or a pipeline of specially created models, a single end-to-end model can be developed to predict a caption given a photo.
RNNs are now quite potent especially for modelling sequential data. In his blog post The Unreasonable Effectiveness of Recurrent Neural Networks, Andrej Karapathy explains the application of RNNs in a very clear and concise manner. RNNs often fall into one of four categories.

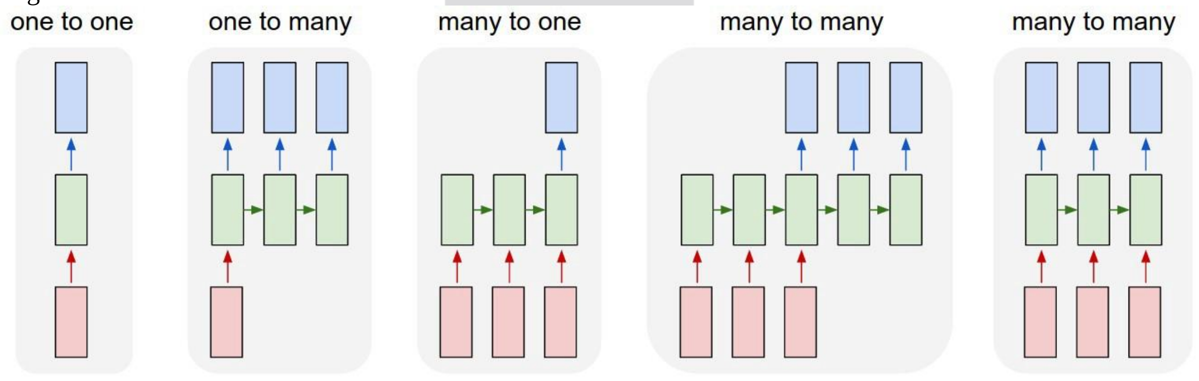


Figure1-Types of RNNs

CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional Neural Network (CNN) is a Deep Learning method that takes in an input image and gives priority (learnable weights and biases) to different characteristics and objects in the image to help it distinguish between distinct images.

The output of the first convolution layer becomes the input for the second layer after the image has gone through one convolution layer. For each layer after that, this process is repeated.

It is important to attach a completely connected layer following a series of convolutional, nonlinear, and pooling layers. The output data from convolutional networks is used in this layer. An N-dimensional vector, where N is the number of classes from which the model chooses the desired class, is produced by attaching a fully linked layer to the end of the network.

RECURRENT NEURAL NETWORKS (RNN)

The nodes in a directed graph which constitute a recurrent neural network (RNN) will be connected in the form of a series. It facilitates RNN's management of series operations like time sequence, handwriting recognition, and sequence expression. RNNs have the capacity to retain key details about the input they receive, enabling them to anticipate the subsequent item in a sequence.

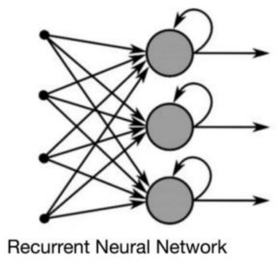


Figure 2: Sample Recurrent Neural Network

RNN usually features a STM and hence cannot handle very long sequences. Long STM (LSTM) network extends RNN which extends the memory of RNN. Therefore, LSTM are often employed in problems with sequences having long gaps. LSTMs can remember the previous inputs for an extended duration because it stores all those data into a memory. In image captioning problems, captions are generated from image features using RNN alongside LSTM.

LONG SHORT-TERM MEMORY (LSTM)

Recurrent neural networks (RNNs) of the Long Short-Term Memory (LSTM) type are able to recognize order dependence in sequence prediction issues. The most common applications of this are in difficult issues like speech recognition, machine translation, and other issues.

When training conventional RNNs, this issue was observed because as we go further into a neural network, if the gradients are very small or zero, little to no training can occur, resulting in poor predicting performance. Since there may be lags of uncertain length between significant occurrences in a time series, LSTM networks are well-suited for categorizing, processing, and making predictions based on time series data.
As it overcomes the short term memory constraints of the RNN, LSTM is significantly more efficient and superior to the regular RNN. The LSTM can process inputs while processing pertinent information, and it can ignore irrelevant information.

CNN LSTM ARCHITECTURE

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE and SI do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.
The CNN-LSTM architecture combines LSTMs to facilitate sequence prediction with CNN layers for feature extraction on input data. This approach is especially made for problems involving the sequence prediction of spatial inputs, such as photos or videos. They are frequently utilized in activities including activity recognition, image and video description, and many others.The general architecture of the CNN-LSTM Model is shown in Fig-2:

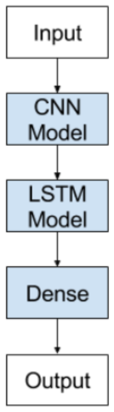


Figure 3 -General Architecture of CNN- LSTM Model

III. TECHNOLOGY USED

A. PYTHON

Python is a general-purpose, interpreted programming language. Python was developed by Guido van Rossum and originally made available in 1991. Its design philosophy places a strong emphasis on code readability and makes remarkable use of substantial whitespace. Its language constructs and object-oriented methodology are designed to aid programmers in creating clean, comprehensible code for both little and big projects. Python uses garbage collection and has dynamic typing. Procedural, object-oriented, and functional programming are just a few of the programming paradigms it supports. Due to its extensive standard library, Python is frequently referred to as a "batteries included" language.

B. JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

C. GOOGLE COLAB

Colaboratory, sometimes known as "Colab," is a Google Research product. Colab is particularly well suited to machine learning, data analysis, and education. It enables anyone to create and execute arbitrary Python code through the browser. Technically speaking, Colab is a hosted Jupyter notebook service that offers free access to computer resources, including GPUs, and requires no setup to use.

D. KAGGLE KERNELS

Jupyter notebooks may be run instantly before your eyes with Kaggle Kernels, which are completely free! This is extremely fantastic, a free platform for running Jupyter notebooks in the browser is called Kaggle Kernels. This implies that whenever you have an internet connection, you can use a Jupyter notebook environment without having to deal with the complexity of setting up a local environment. Additionally, since the notebook's computing power originates from cloud servers rather than your local system, you can perform a lot of data science and machine learning without depleting your laptop's battery!

E. PYTHON LIBRARIES

1) **Pandas-** Pandas is an open-source library designed primarily for working with relational or labelled data in an easy and intuitive manner. It offers a number of data structures and operations for manipulating numerical data and time series. This library is based on the NumPy library. Pandas is quick, with high performance and productivity for users.

2) **Numpy**- NumPy is a Python-based array-processing library. It includes a high-performance multidimensional array object as well as tools for manipulating these arrays. It is the fundamental Python package for scientific computing. It is free and open-source software.

3) **Matplotlib-** Matplotlib is a fantastic Python visualization library for 2D array plots. Matplotlib is a multi-platform data visualization library based on NumPy arrays and designed to work with the SciPy stack as a whole. It was first introduced in 2002 by John Hunter. One of the most significant advantages of visualization is that it provides us with visual access to massive amounts of data in easily digestible visuals. Matplotlib includes a variety of plots such as line, bar, scatter, histogram, and so on.

4) **Keras-** Keras is a Python-based open-source high-level Neural Network library that can run on Theano, TensorFlow, or CNTK.

5) **RegEx-** A Regular Expression (RegEx) is a special sequence of characters that searches for a string or set of strings using a search pattern. It can detect the presence or absence of text by matching it with a specific pattern, and it can also divide a pattern into one or more sub-patterns. Python includes a re module that allows you to use regex in Python. Its primary function is to perform a search using a regular expression and a string. It either returns the first match or none at all.

6) **NLTK** -The NLTK Library (Natural Language Toolkit) is a collection of libraries and programmes for statistical language processing. It is one of the most powerful NLP libraries, containing packages for making machines understand human language and respond appropriately.

7) **Pickle** -The pickle module in Python is used to serialize and de-serialize a Python object structure. In Python, any object can be pickled and saved to disc. Pickle begins by "serializing" the object before writing it to file. Pickling is a method for converting a Python object (list, dictionary, etc.) into a character stream. The idea is that this character stream contains all of the information required to reconstruct the object in a subsequent Python script.

8) **TensorFlow** -TensorFlow is a Google open-source library designed primarily for deep learning applications. Traditional machine learning is also supported. TensorFlow was originally designed for large numerical computations rather than deep learning. However, it proved to be very useful for deep learning development as well, so Google made it open source.

IV. DATASET SPECIFICATIONS

The project's goal is to predict the captions for the input image. The dataset contains 8k images with 5 captions per image. For input, the features are extracted from both the image and the text captions. The features will be concatenated to predict the caption's next word. CNN is used for image recognition, while LSTM is used for text recognition. The BLEU Score is a metric used to assess the performance of the trained model.

FLICKR8K DATASET

Flickr8k dataset [12] is a public benchmark dataset for image to sentence description. This dataset consists of 8000 images for each image, with five captions. The dataset depicts a variety of events and scenarios and doesn't include images containing well-known people and places which makes the dataset more generic. The dataset has 6000 images in training dataset, 1000 images in development dataset and 1000 images in test dataset. Features of the dataset making it suitable for this research are:

- Multiple captions mapped for a single image makes the model generic and avoids over fitting of the model.
- Diverse category of training images can make the image captioning model to work for multiple categories of images and hence can make the model more robust.

V. METHODOLOGY

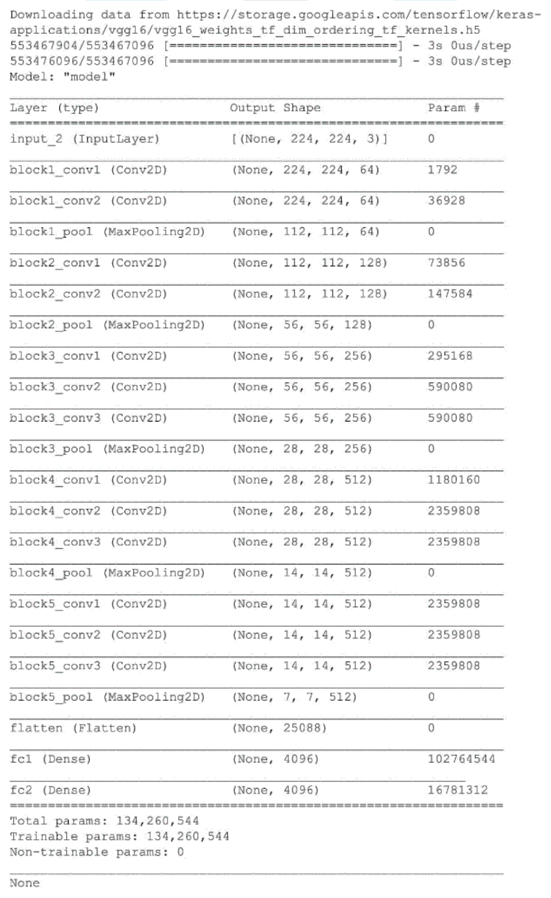
1. IMPORT MODULES

First, we have to import all the basic modules we will be needing for this project, such as-

a) **os -** used to handle files using system commands.
b) **pickle** - used to store numpy features extracted
c) **numpy -** a Python library that can perform a wide range of mathematical operations on arrays.
d) **tqdm** - progress bar decorator for iterators. Includes a default range iterator printing to stderr.
e) **VGG16, preprocess_input** - imported modules for feature extraction from the image data
f) **load_img, img_to_array** - used for loading the image and converting the image to a numpy array
g) **Tokenizer** - used for loading the text as convert them into a token
h) **pad_sequences -** used for equal distribution of words in sentences filling the remaining spaces with zeros
i) **plot_model** - used to visualize the architecture of the model through different images

2. EXTRACT IMAGE FEATURES

We have to load and restructure the model. In this method we have to load the VGG16 model by using keras and TensorFlow to restructure the model. Only the previous layers of the VGG16 model are required to extract feature results, not the fully connected layer.You may include more layers if you prefer, but for faster results, avoid adding unnecessary layers.



```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 3s 0us/step
553476096/553467096 [==============================] - 3s 0us/step
Model: "model"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
input_2 (InputLayer)        [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)           (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                 (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                 (None, 4096)              16781312
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 4: VGG16 Model

3. LOAD THE DATA FOR PREPROCESSING AND EXTRACT THE IMAGE FEATURES

a. The dictionary 'features' is created and will be loaded with image data extracted features.

b. **image path, target size=(224, 224))** - a custom dimension that will be used to resize the image when it is loaded into the array
c. **image.reshape((1, image.shape[0], image.shape[1], image.shape[2])** - reshaping image data for preprocessing in an RGB image.
d. **model.predict(image, verbose=0)** - Image feature extraction
e. **img name.split('.')[0]** - Removed the image name from the extension in order to load only the image name.

4. STORE FEATURES IN PICKLE

Since extracted features are not saved to disc, re-extraction of features can increase running time. To save time, dump and store your dictionary in a pickle for reloading.

.

5. LOAD FEATURES FROM PICKLE

Load all of your previously saved feature data into your project for faster execution.

6. LOAD THE CAPTIONS DATA, SPLIT AND APPEND THE CAPTIONS DATA WITH THE IMAGE

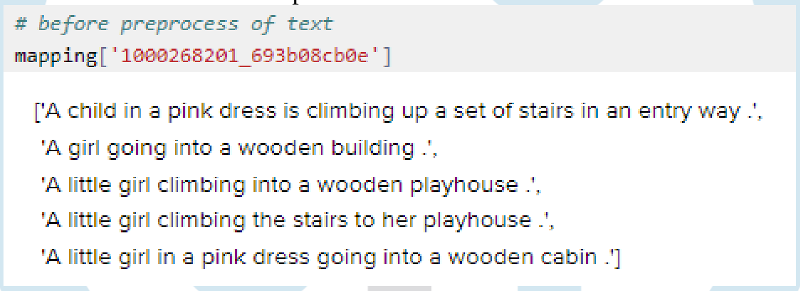
A dictionary 'mapping' is created with image id as the key and caption text as the values. If image id is not in mapping, the same image may have multiple captions: **mapping [image id] = []** generates a list of captions to append to the corresponding image.

```
8091
```

Figure 5: length of mapping

7. PREPROCESS TEXT DATA

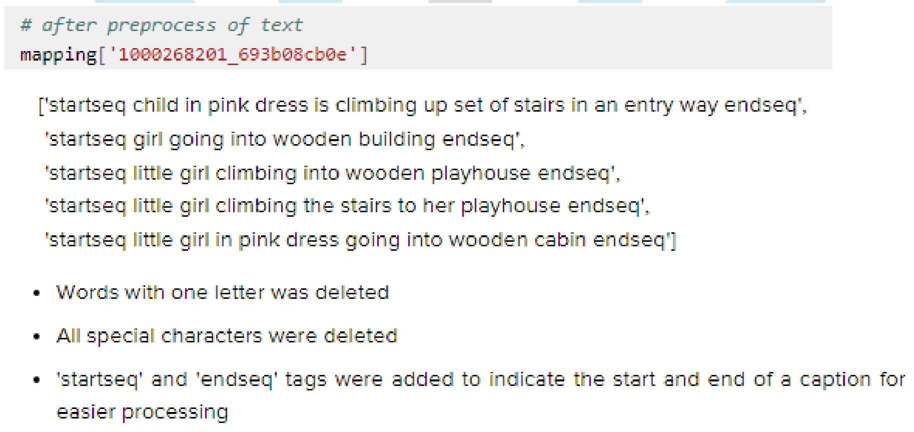
Defined to clean and convert text for a faster and better process.



```
# before preprocess of text
mapping['1000268201_693b08cb0e']
```

```
['A child in a pink dress is climbing up a set of stairs in an entry way .',
 'A girl going into a wooden building .',
 'A little girl climbing into a wooden playhouse .',
 'A little girl climbing the stairs to her playhouse .',
 'A little girl in a pink dress going into a wooden cabin .']
```

Figure 6 : before preprocess of text



```
# after preprocess of text
mapping['1000268201_693b08cb0e']
```

```
['startseq child in pink dress is climbing up set of stairs in an entry way endseq',
 'startseq girl going into wooden building endseq',
 'startseq little girl climbing into wooden playhouse endseq',
 'startseq little girl climbing the stairs to her playhouse endseq',
 'startseq little girl in pink dress going into wooden cabin endseq']
```

- Words with one letter was deleted
- All special characters were deleted
- 'startseq' and 'endseq' tags were added to indicate the start and end of a caption for easier processing

Figure 7: After preprocess of text

8. TOKENIZE THE TEXT AND CALCULATE LENGTH OF THE CAPTIONS

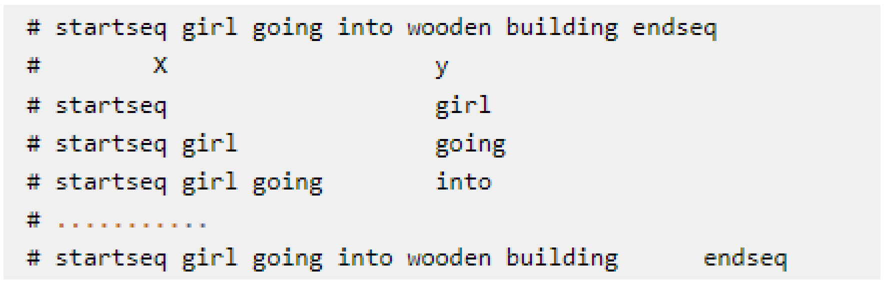
Vocabulary size and number of unique words finding the maximum length of the captions, which will be used as a guide for the padding sequence.

```
vocab_size    max_length

8485          35
```

9. TRAIN TEST SPLIT

If your system does not have enough memory, it may cause your session to crash depending on the size of the data. If you have less than 16 GB of memory, creating and loading data in batches is extremely useful.



```
# startseq girl going into wooden building endseq
#        X                          y
# startseq                          girl
# startseq girl                     going
# startseq girl going               into
# ...........
# startseq girl going into wooden building      endseq
```

Figure 8 : Explanatory example of the sequence split into pairs

10. DEFINING A BATCH AND INCLUDE THE PADDING SEQUENCE

For better results, the padding sequence normalises the size of all captions to the maximum size, filling them with zeros.

11. MODEL CREATION

a. **shape=(4096,)** - output length of the features from the VGG model
b. **Dense** - single dimension linear layer array
c. **Dropout()** - used to add regularization to the data, avoiding over fitting & dropping out a fraction of the data from the layers
d. **model.compile()** - compilation of the model
e. **loss='sparse_categorical_crossentropy'** - loss function for category outputs
f. **optimizer='adam'** - automatically adjust the learning rate for the model over the no. of epochs
g. Model plot shows the concatenation of the inputs and outputs into a single layer
h. Feature extraction of image was already done using VGG, no CNN model was needed in this step.

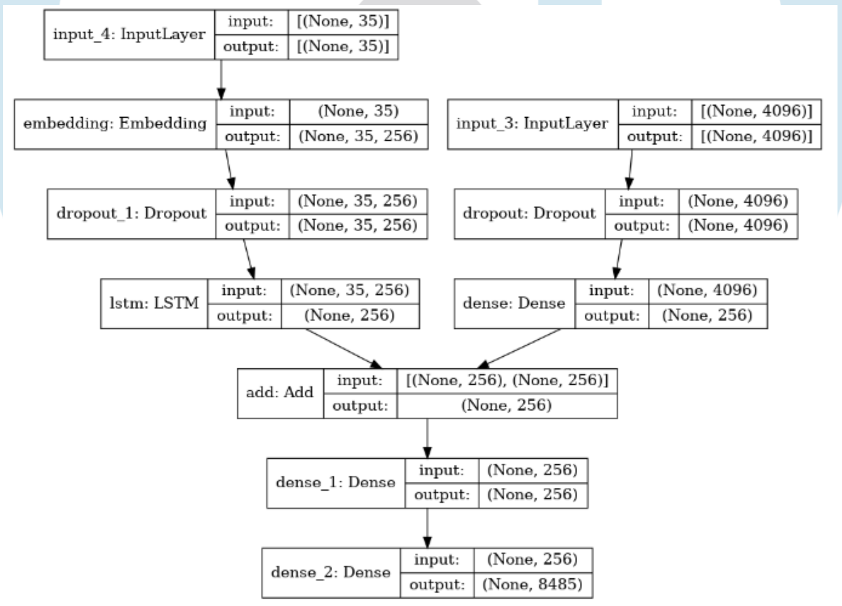
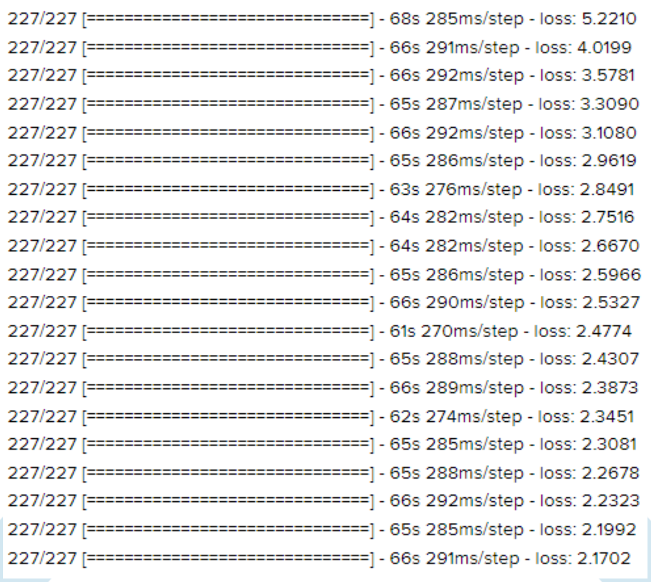

input_4: InputLayer | input: [(None, 35)] | output: [(None, 35)]
embedding: Embedding | input: (None, 35) | output: (None, 35, 256)
input_3: InputLayer | input: [(None, 4096)] | output: [(None, 4096)]
dropout_1: Dropout | input: (None, 35, 256) | output: (None, 35, 256)
dropout: Dropout | input: (None, 4096) | output: (None, 4096)
lstm: LSTM | input: (None, 35, 256) | output: (None, 256)
dense: Dense | input: (None, 4096) | output: (None, 256)
add: Add | input: [(None, 256), (None, 256)] | output: (None, 256)
dense_1: Dense | input: (None, 256) | output: (None, 256)
dense_2: Dense | input: (None, 256) | output: (None, 8485)

Figure 9: Model Created after feature extraction

12. TRAIN THE MODEL

a. **steps = len(train)** // batch_size - back propagation and fetch the next data
b. Loss decreases gradually over the iterations
c. Increase the no. of epochs for better results
d. Assign the no. of epochs and batch size accordingly for quicker results



```
227/227 [==============================] - 68s 285ms/step - loss: 5.2210
227/227 [==============================] - 66s 291ms/step - loss: 4.0199
227/227 [==============================] - 66s 292ms/step - loss: 3.5781
227/227 [==============================] - 65s 287ms/step - loss: 3.3090
227/227 [==============================] - 66s 292ms/step - loss: 3.1080
227/227 [==============================] - 65s 286ms/step - loss: 2.9619
227/227 [==============================] - 63s 276ms/step - loss: 2.8491
227/227 [==============================] - 64s 282ms/step - loss: 2.7516
227/227 [==============================] - 64s 282ms/step - loss: 2.6670
227/227 [==============================] - 65s 286ms/step - loss: 2.5966
227/227 [==============================] - 66s 290ms/step - loss: 2.5327
227/227 [==============================] - 61s 270ms/step - loss: 2.4774
227/227 [==============================] - 65s 288ms/step - loss: 2.4307
227/227 [==============================] - 66s 289ms/step - loss: 2.3873
227/227 [==============================] - 62s 274ms/step - loss: 2.3451
227/227 [==============================] - 65s 285ms/step - loss: 2.3081
227/227 [==============================] - 65s 288ms/step - loss: 2.2678
227/227 [==============================] - 66s 292ms/step - loss: 2.2323
227/227 [==============================] - 65s 285ms/step - loss: 2.1992
227/227 [==============================] - 66s 291ms/step - loss: 2.1702
```

Figure 10 : 20 no. of epochs

13. SAVE THE MODEL

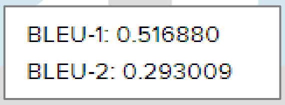
We can save the trained model in the working directory for reuse

14. GENERATE CAPTIONS FOR THE IMAGE

Convert the model's predicted index into a word Caption generator, appending all the words for an image.
The caption begins with 'startseq,' and the model predicts the caption until 'endseq' appears.

15. VALIDATE THE DATA USING BLEU SCORE

Bilingual Evaluation Understudy or BLEU score is used to evaluate the descriptions generated from translation and other Natural Language Processing (NLP) applications. In a list of tokens, the BLEU Score is used to compare the predicted text to a reference text. All of the words appended from the captions data (actual captions) are included in the reference text. A BLEU Score greater than 0.4 is considered good; for a higher score, increase the number of epochs accordingly.

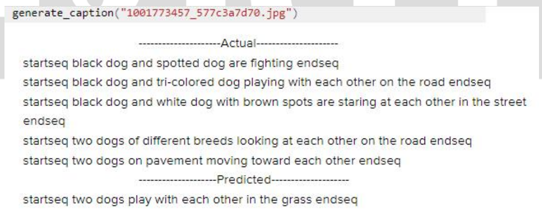


```
BLEU-1: 0.516880
BLEU-2: 0.293009
```

Figure 11: BLEU Score for input image

16. VISUALIZE THE RESULTS

Image caption generator formed. The actual captions of the image are printed first, followed by a predicted caption of the image.



```
generate_caption("1001773457_577c3a7d70.jpg")

--------------------Actual--------------------
startseq black dog and spotted dog are fighting endseq
startseq black dog and tri-colored dog playing with each other on the road endseq
startseq black dog and white dog with brown spots are staring at each other in the street
endseq
startseq two dogs of different breeds looking at each other on the road endseq
startseq two dogs on pavement moving toward each other endseq
-------------------Predicted-------------------
startseq two dogs play with each other in the grass endseq
```

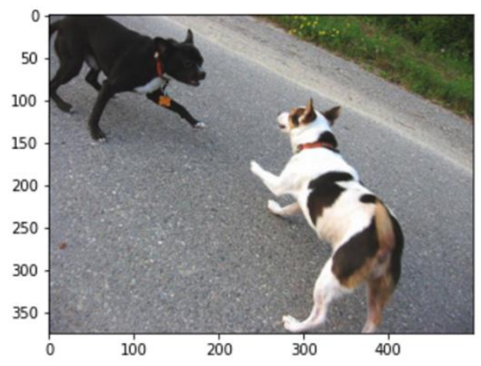


Figure 12: Captions generated of the given Image

VI. CONCLUSION

In this paper, we learned and designed an Image Caption Generator technique that will respond to the user with captions or descriptions based on an image. The Image Based Model extracts image features, and the Language Based Model translates the image features and objects into natural sentences. CNN is used in the image-based model, whereas LSTM is used in the language-based model. Training the model by increasing the no. of epochs can give better and more accurate results. Processing large amounts of data can consume a significant amount of time and system resources. If we want to process large datasets like flickr32k, we can increase the number of layers in the model. In addition to VGG16, we can use the CNN model to extract image features.
The workflow is as follows: data collection, pre-processing, training model, and prediction. The ultimate goal of an image caption generator is to improve social media platforms, image indexing, and accessibility for visually impaired people through automated generated captions or descriptions.

References

[1] https://research.google.com/colaboratory/faq.html
[2] https://towardsdatascience.com/introduction-to-kaggle-kernels-2ad754ebf77
[3] https://www.geeksforgeeks.org/introduction-to-pandas-in-python/
[4] https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/
[5] https://www.geeksforgeeks.org/python-introduction-matplotlib/
[6] https://www.javatpoint.com/keras
[7] https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/
[8] https://www.analyticsvidhya.com/blog/2021/07/getting-started-with-nlp-using-nltk-library/
[9] https://www.geeksforgeeks.org/understanding-python-pickling-example/
[10] https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-tensorflow
[11] https://medium.com/@raman.shinde15/image-captioning-with-flickr8k-dataset-bleu-4bcba0b52926
[12] M. Hodosh, P. Young, J. Hockenmaier, "Framing image description as a ranking task: Data models and evaluation metrics". *Journal of Artificial Intelligence Research*, pp. 853-899, 2013.
[13] https://ieeexplore.ieee.org/document/8276124