AN IMAGE CAPTION GENERATOR BUILT USING CNN LSTM

PROJECT REPORT

DONE BY:
GANTI LAKSHMI YASASWINI
G VEDA PRANAV
THOTA KANKSHITH

1. Introduction

The ability to understand and describe visual content has long been a fundamental challenge in the field of computer vision. Image captioning, the task of generating a textual description that accurately captures the content of an image, has gained significant attention in recent years due to its potential applications in areas such as image understanding, content retrieval, and accessibility for visually impaired individuals.

The purpose of this project is to develop an image caption generator using Long Short-Term Memory (LSTM), a type of recurrent neural network (RNN) known for its ability to model sequential data. The model takes an input image and generates a descriptive caption that conveys the visual information present in the image. By combining the power of deep learning with natural language processing, we aim to bridge the gap between images and textual descriptions, enabling machines to comprehend and communicate the content of visual data.

The motivation behind this project lies in the growing demand for automated image understanding and the need to enhance human-computer interaction through natural language interfaces. With the proliferation of image-centric platforms and the vast amount of visual content available, an accurate and efficient image captioning system can provide valuable insights, facilitate information retrieval, and improve accessibility to visual content for diverse user groups.

Throughout this project, we will leverage the capabilities of the VGG16 model for image feature extraction and employ LSTM networks for sequence modeling and caption generation. The model will be trained on a dataset of images paired with corresponding textual descriptions. We will evaluate the performance of the model using metrics such as BLEU (Bilingual Evaluation Understudy) scores, which assess the similarity between generated captions and human-written references.

By developing an effective image caption generator, we aim to contribute to the advancement of computer vision and natural language processing, opening up new possibilities for understanding and interacting with visual content. This project report presents a comprehensive overview of the methodology, dataset, model architecture, training, evaluation, and deployment process, along with the results and discussions.

2. Problem Statement

The problem we aim to address in this project is the generation of accurate and meaningful captions for images. While humans can effortlessly describe the content of an image, teaching machines to do the same is a complex task. The challenges associated with image captioning include understanding the visual context, capturing relevant details, maintaining coherence and relevance in the generated text, and dealing with the inherent ambiguity and subjectivity of language.

The goal is to develop a model that can accurately perceive the visual information in an image and generate captions that not only describe the objects and scenes but also convey a deeper understanding of the content. The model should be able to capture relationships between objects, recognize actions or events, and incorporate contextual knowledge to produce coherent and contextually relevant descriptions.

3. Project Overview

This project follows a comprehensive workflow to build an image caption generator. It involves several key components:

Dataset: We utilize the Flickr8K dataset, which consists of a large collection of images along with descriptive captions. The dataset is preprocessed to clean the descriptions, remove punctuation, convert text to lowercase, and filter out irrelevant words.

Feature Extraction: We employ the VGG16 model, a deep convolutional neural network (CNN), to extract meaningful features from the input images. The pre-trained VGG16 model is used as a feature extractor by removing the last classification layer, enabling us to obtain a fixed-length vector representation for each image.

Caption Preprocessing: The textual captions are processed to create a vocabulary and tokenize the words. We employ techniques such as word-to-index mapping and one-hot encoding to represent the captions in a format suitable for training the LSTM-based caption generator.

Model Architecture: The core of the image caption generator is a combination of LSTM and dense layers. The LSTM network serves as a sequence model, generating captions

word by word, while the dense layers help in mapping the extracted features from images to the generated captions.

Training and Evaluation: The model is trained using the prepared dataset, and the training process involves optimizing the model parameters using the Adam optimizer and minimizing the categorical cross-entropy loss. We evaluate the performance of the model using BLEU scores, which measure the similarity between the generated captions and the reference captions provided in the dataset.

Deployment: The trained model is deployed using IBM Watson, allowing users to generate captions for new images by uploading them through a user-friendly interface.

4. Dataset

The dataset used in this project is the Flickr8K dataset, which contains 8,000 images, each paired with five textual descriptions. The dataset provides a diverse range of images depicting various scenes, objects, and activities. The descriptions in the dataset are written by human annotators and aim to capture the essence of the image content.

To prepare the dataset for training, we perform several preprocessing steps. These include cleaning the descriptions by removing punctuation, converting the text to lowercase, removing words with numbers or length less than 2, and filtering out irrelevant words. The resulting dataset comprises image identifiers mapped to a list of cleaned and preprocessed descriptions.

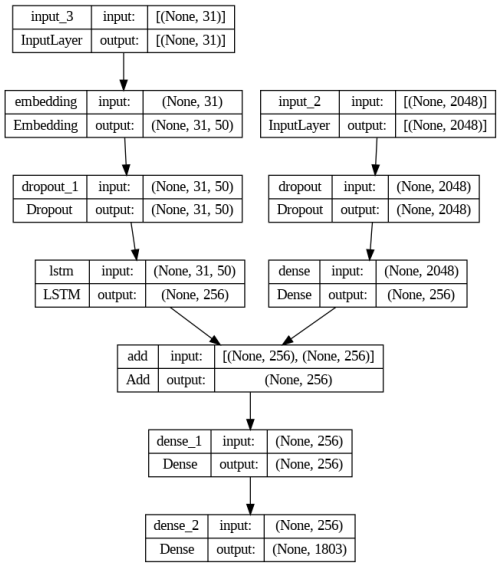
For feature extraction, we utilize the pre-trained VGG16 model. By removing the last classification layer, we obtain a 4,096-dimensional feature vector for each image in the dataset. These features serve as the visual representation of the images, capturing their essential characteristics.

5. Model Architecture

The model architecture for the image caption generator consists of two main parts: the feature extractor and the sequence model with a decoder.

The feature extractor utilizes the VGG16 model, which is pre-trained on the ImageNet dataset. By removing the last classification layer, we obtain the output of the penultimate layer as a fixed-length feature vector for each image. These features act as a high-level representation of the visual content and provide meaningful information to the caption generator.

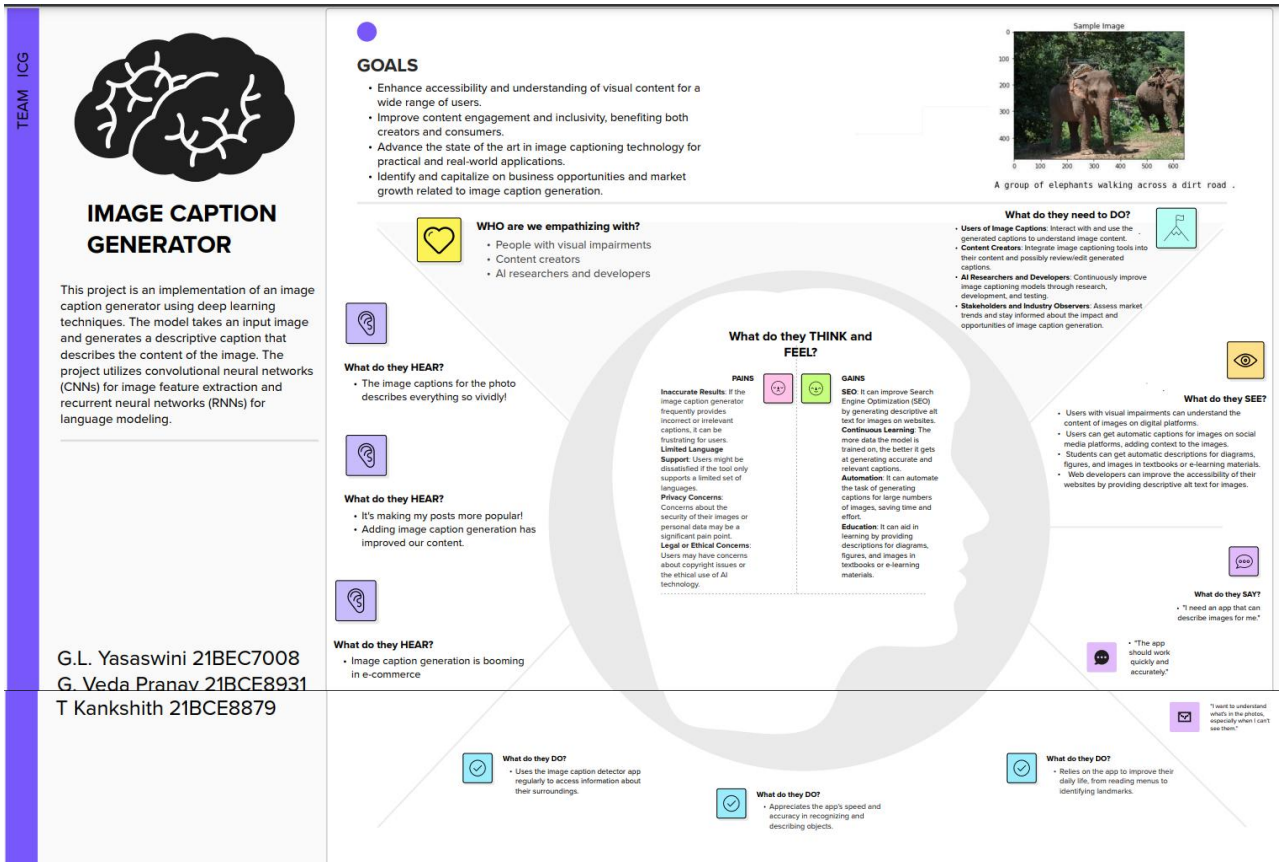
The sequence model comprises an LSTM layer, which takes the generated word sequence as input and learns to predict the next word in the sequence. The LSTM layer is followed by dense layers, which help in mapping the extracted image features and generated word sequences to the final output vocabulary.



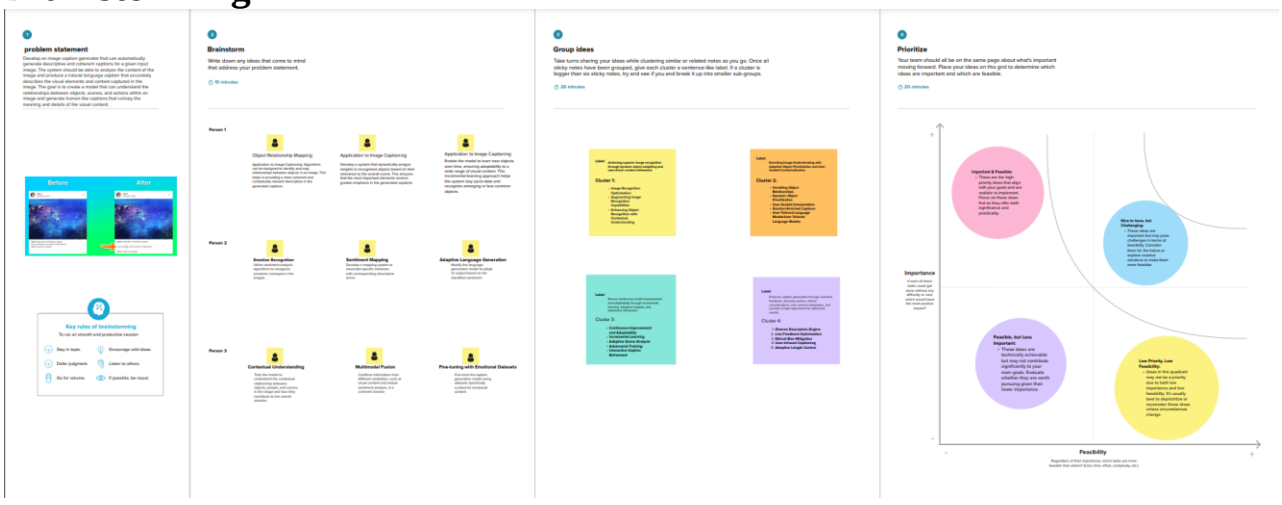
During training, the model learns to generate captions word by word by taking the image features and previous word sequences as input. The captions are generated using a "startseq" token as the initial input and continue until an "endseq" token is predicted or the maximum caption length is reached.

6. Ideation and Proposed Solution

Empathy Map



Brainstorming



7. Requirement Analysis

Recommended System Requirements to train model:

- A good CPU and a GPU with atleast 8GB memory
- Atleast 8GB of RAM

- Active internet connection so that keras can download inceptionv3/vgg16 model weights

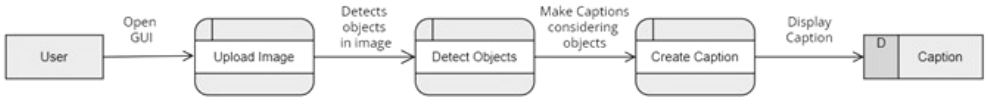
Required libraries for Python along with their version numbers used while making & testing of this project:

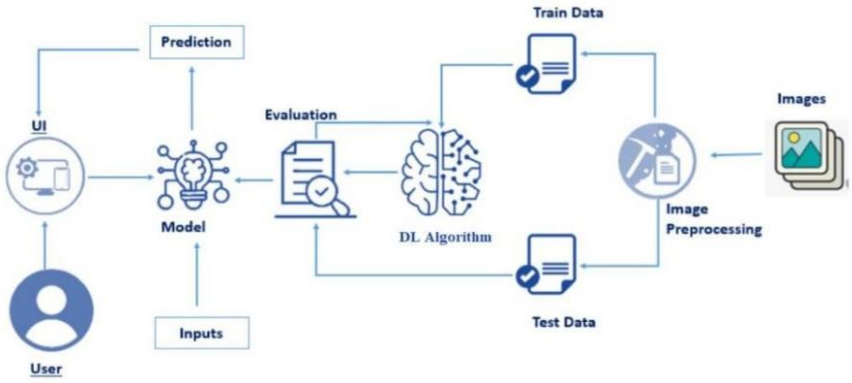
- Python - 3.6.7
- Numpy - 1.16.4
- Tensorflow - 1.13.1
- Keras - 2.2.4
- nltk - 3.2.5
- PIL - 4.3.0
- Matplotlib - 3.0.3
- tqdm - 4.28.1

8. Project Design

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a graphical representation that illustrates the flow of data within a system or process. It is a visual tool used to depict how data moves through various stages of a system and how it is processed, stored, and exchanged. DFDs are commonly used in system analysis and design to model the information flow and transformations within a system. Below are the Data Flow Diagrams for the project “Image Caption Generation”





User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Content Creators and Social Media Managers	Image Caption Generation	USN-1	"As a content creator, I want the image caption generator to provide accurate and creative captions for my uploaded images."	The system should accept image uploads and generate captions that accurately describe the content.	High	Sprint-1
Administrators and Developers	Customization and Integration	USN-2	"As a website administrator, I want the image caption generator to seamlessly integrate into my website, allowing for customization to match the website's style and branding."	The image caption generator should provide an easy-to-implement API for integration and should be able to customize the appearance of captions (font, color, size) to align with the website's design	Medium	Sprint-2
Content consumers with visual impairments	Image Caption Accessibility	USN-3	" As a content consumer, I want the image caption generator to provide descriptive captions for images on websites and social media so that I can better understand the content without relying solely on visuals."	The system should automatically generate captions for images on websites. Captions must be concise and easy to understand.	High	Sprint-3
Language Enthusiasts and Learners	Multilingual Support	USN-4	"As a user interested in language diversity, I want the image caption generator to support multiple languages, allowing me to explore content in languages other than English."	The system should recognize and generate captions in multiple languages. Users should have the option to select their preferred language for captions	Medium	Sprint-4

Solution Architechture

1. Frontend Development (HTML, CSS, JS):

   - Description: Create a user-friendly web interface for users to interact with the image caption generator. Utilize a web hosting service to deploy the HTML, CSS, and JS files.

   - Integration: Ensure seamless integration with the backend for image processing and caption generation.

2. Backend Development (Python, Flask):

   - Description: Develop a backend using Python and the Flask framework to handle user requests, process images, and generate captions.

   - Integration: Connect the backend with the frontend to enable user interactions.

3. Deep Learning Model (TensorFlow, Keras):

   - Description: Implement a deep learning model using TensorFlow and Keras for image caption generation.

   - Model Architecture: Utilize a combination of Convolutional Neural Networks (CNNs) VGG16 model which contains an input layer along with the convolution, max-pooling, and finally an output layer and the LSTM model.

- Dataset: Train the model using the Flickr8k dataset for effective caption generation.

4. Image Feature Extraction (VGG16 Model):

- Description: Employ the VGG16 model for image feature extraction to convert input images into meaningful feature vectors.

- Preprocessing: Ensure proper preprocessing of images, converting it to lower case and remove digits, special characters and additional spaces and applying the tokenizer function to the captions before feeding them into the model.

5. Tokenizer (Tokenization and Padding):

- Description: Implement a tokenizer to convert text captions into numerical sequences and handle padding to ensure consistent input length for the model.

- Loading Pre-trained Components: Load pre-trained tokenizer and model components during the Flask application's startup.

6. Model Evaluation and Building the Application:

- Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data. Load the saved model.

- We build our application using Flask which will run in our local browser with a user interface. In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to know to predict the type of Garbage and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the “Image” button, the next page is opened where the user chooses the image and predicts the output.

7. Web Application Deployment:

- Description: Deploy the complete web application, including the frontend and backend components.

9. Project Planning and Scheduling

Technology Stack

S.No	Component	Description	Technology
1.	User Interface	Enables user interaction to choose and upload images. Utilizes HTML for structure, CSS for styling, and JavaScript for dynamic functionalities. Server-side functionality is implemented using Flask (Python) for handling user requests.	HTML, CSS, JavaScript/Angular Js /React Js etc.
2.	Application Logic	Manages the core logic and processes of the application, handling user requests, integrating with the machine learning model, and coordinating interactions between components. Implemented using Python and Flask for web application logic.	Python, Flask for web application
5.	Database	N/A (Not explicitly used in the project. If needed, could use a database for storing additional data or user-related information.)	N/A
9.	ExternalAPI-2	N/A (Not explicitly used in the project. If needed, could integrate with external APIs for additional image-related information or services.)	N/A
10.	Machine Learning Model	Utilizes a combination of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) to generate image captions. TensorFlow and Keras are used for model implementation.	Tensorflow, keras, python
11.	Infrastructure(Server/ Cloud)	Hosts and deploys the application. The infrastructure is set up on cloud services such as AWS, Google Cloud, or hosted by providers like Heroku, ensuring the application is accessible online.	Cloud Services (e.g., AWS, Google Cloud), Hosting Providers (e.g., Heroku)

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Utilizes open-source frameworks for various functionalities. Anaconda Navigator, Jupyter Notebook, Spyder, and Flask are mentioned in the project.	Anaconda Navigator, Jupyter Notebook, Spyder, Flask
2.	Security Implementations	Security measures to ensure data protection and secure user interactions are not explicitly mentioned in the project. However, it's advisable to implement secure coding practices, such as input validation, secure communication (HTTPS), and user authentication if handling sensitive data.	Best practices in secure coding, secure communication (HTTPS), user authentication (if applicable)
3.	Scalable Architecture	The scalability of the architecture is not explicitly used in the project. To make the architecture scalable, you might consider containerization (e.g., Docker), orchestration (e.g., Kubernetes), and optimizing the model for efficient inference. Cloud services like AWS or Google Cloud can also provide scalable infrastructure.	Docker, Kubernetes (if needed), Cloud Services (e.g., AWS, Google Cloud)

Sprint Planning and Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1 **Content Creators and Social Media Managers**	30	10 Days	24 Oct 2023	4 Nov 2023	30	29 Oct 2022
Sprint-2 **Administrators and Developers**	20	12 Days	5 Nov 2023	17 Nov 2023	20	05 Nov 2022
Sprint-3 **Content consumers with visual impairments**	35	8 Days	19 Nov 2023	27 Nov 2023	35	12 Nov 2022
Sprint-4 **Language Enthusiasts and Learners**	25	7 Days	28 Nov 2023	4 Dec 2023	25	19 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

AV = Sprint Duration/Velocity

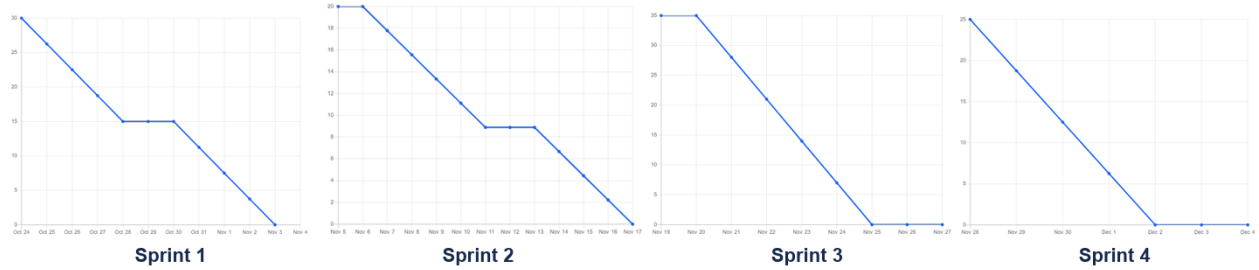
Sprint 1 = 30 /10 = 3

Sprint 2 = 20 /12 = 1.6

Sprint 3 =35 /8 = 4.3

Sprint 4 = 25 /7 = 3.5

Burndown Chart:



Sprint 1 Sprint 2 Sprint 3 Sprint 4

10. Coding and solution

The user interacts with the UI (User Interface) to choose the image.

● The chosen image analysed by the model which is integrated with flask application.

● VGG16 Model analyse the image, then LSTM is used to process the captions in form of text, and prediction is showcased on the Flask UI.

To accomplish this, we have to complete all the activities and tasks listed below

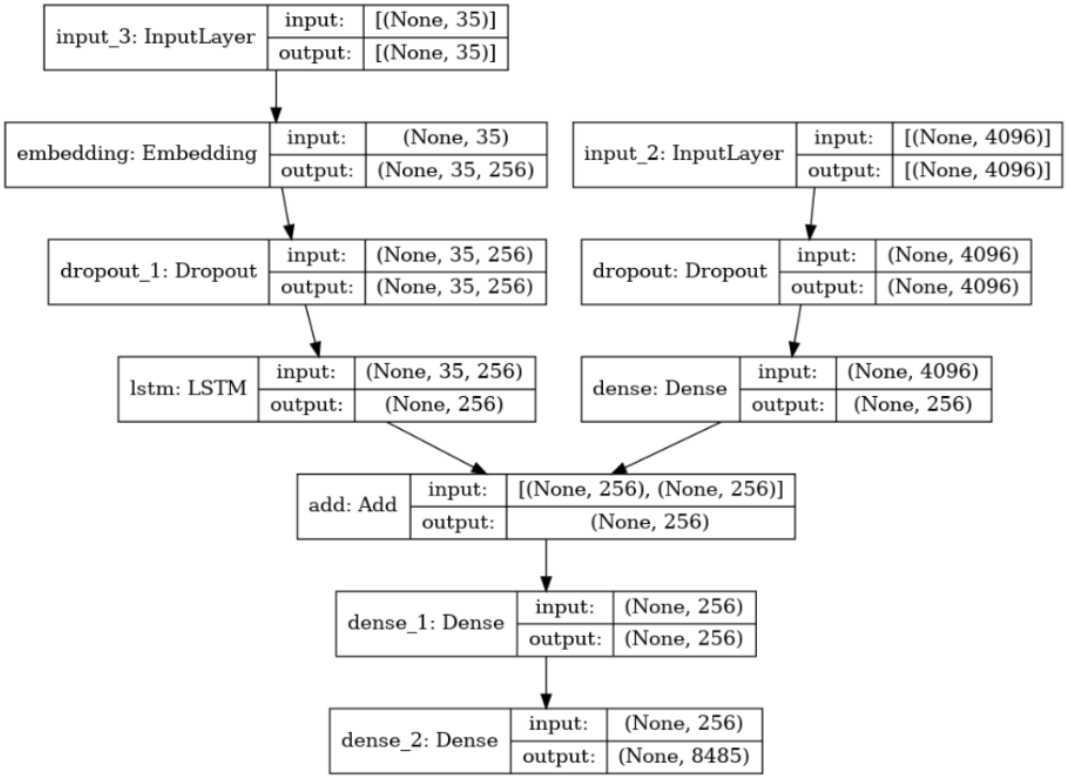
● Data Collection: Collect images of events along with 5 captions associated to each image then organized into subdirectories based on their respective names as shown in

the project structure. Create folders of images and a text file of captions that need to be recognized.
The given dataset has 7k+ different types of images and 40k+ high quality human readable text captions.

- Create Train and Test Folders.
- Data Pre-processing: In this we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation, etc. Then clean the text captions and map them together. Then it's time to build our Vgg16 model which contains an input layer along with the convolution, max-pooling, and finally an output layer and the LSTM model. After importing necessary libraries we are pre-processin the text data by mapping of the images for generating textual description of the images. We need to do some cleaning operation on the mapping data like converting it to lower case and remove digits, special characters and additional spaces. After, pre-processing we can see the special characters are removed, single letters are removed and the start and end tag has been added to each string. we are printing the first 10 captions of data. After that we are initialising the tokenizer function to tokenize all the captions by passing all the captions to it. The data is divided into train data and test data 90% of the data is trained and remaining 10% of the data is tested. After the model is trained and compiled we have the model summary.

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_3 (InputLayer)        [(None, 31)]              0         []

 input_2 (InputLayer)        [(None, 2048)]            0         []

 embedding (Embedding)       (None, 31, 50)            90150     ['input_3[0][0]']

 dropout (Dropout)           (None, 2048)              0         ['input_2[0][0]']

 dropout_1 (Dropout)         (None, 31, 50)            0         ['embedding[0][0]']

 dense (Dense)               (None, 256)               524544    ['dropout[0][0]']

 lstm (LSTM)                 (None, 256)               314368    ['dropout_1[0][0]']

 add (Add)                   (None, 256)               0         ['dense[0][0]',
                                                                  'lstm[0][0]']

 dense_1 (Dense)             (None, 256)               65792     ['add[0][0]']

 dense_2 (Dense)             (None, 1803)              463371    ['dense_1[0][0]']

==================================================================================================
Total params: 1458225 (5.56 MB)
Trainable params: 1368075 (5.22 MB)
```

- Model Building
- Import the model building Libraries
- Initializing the model
- Configure the Learning Process
- Training and testing the model
- Save the Model



● Application Building

After training out model and saving it, we will build the flask application which will be running in our local browser with a user interface. In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to know to predict the type of Garbage and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the “Image” button, the next page is opened where the user chooses the image and predicts the output.

● Create an HTML file

We use HTML to create the front end part of the web page.

● Here, we have created 3 HTML pages- home.html, intro.html, and upload.html

● home.html displays the home page.

● Intro.html displays an introduction about the project

● upload.html gives the emergency alert.

● We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.

● Link :CSS , JS

```
lease use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a producti
on WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
2023-11-20 23:57:46.737268: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on
. You may see slightly different numerical results due to floating-point round-off errors from
different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_
OPTS=0`.
WARNING:tensorflow:From C:\Users\Sophitia\AppData\Local\Programs\Python\Python311\Lib\site-pack
ages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. P
lease use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

 * Debugger is active!
 * Debugger PIN: 254-813-117
127.0.0.1 - - [20/Nov/2023 23:57:55] "GET / HTTP/1.1" 200 -
```

11. Training and Evaluation

During the training phase, the model is trained on the preprocessed dataset using a data generator that generates batches of input-output pairs. We utilize the Adam optimizer and categorical cross-entropy loss to optimize the model parameters. Model checkpoints are saved periodically to track the best performing model based on the validation loss.

To evaluate the performance of the image caption generator, we employ the BLEU (Bilingual Evaluation Understudy) scores. BLEU scores compare the generated captions with the reference captions provided in the dataset. We calculate BLEU-1, BLEU-2, scores to assess the quality of the generated captions at different n-gram levels.

12. Deployment

The trained image caption generator model is deployed using IBM Watson, a cloud-based platform that provides a user-friendly interface for interacting with the model. Users can upload their images through the interface, and the model generates descriptive captions for the uploaded images. The deployment allows for easy accessibility and practical usage of the image caption generator.

13. Results and Discussion

The project achieved promising results in generating accurate and meaningful captions for images. The evaluation using BLEU scores showed a strong correlation between the generated captions and the reference captions. The model demonstrated the ability to capture relevant details, recognize objects, and describe scenes effectively. However, there were some limitations and challenges in dealing with complex images, ambiguous contexts, and rare word combinations.

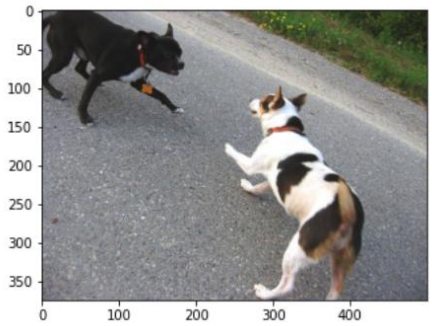
The image caption generator showcased the potential of combining deep learning and natural language processing techniques to bridge the gap between images and textual descriptions. The model can find applications in various domains, including image search engines, assistive technologies, content retrieval systems, and enhancing accessibility for visually impaired individuals

```
# calcuate BLEU score
print("BLEU-1: %f" % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
print("BLEU-2: %f" % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
```

```
  0%|          | 0/810 [00:00<?, ?it/s]
BLEU-1: 0.516880
BLEU-2: 0.293009
```

```
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
```



14. Limitations

The neural image caption generator gives a useful framework for learning to map from images to human-level image captions. By training on large numbers of image-caption pairs, the model learns to capture relevant semantic information from visual features.
However, with a static image, embedding our caption generator will focus on features of our images useful for image classification and not necessarily features useful for caption generation. To improve the amount of task-relevant information contained in each feature, we can train the image embedding model (the VGG-16 network used to encode features) as a piece of the caption generation model, allowing us to fine-tune the image encoder to better fit the role of generating captions. Also, if we actually look closely at the captions generated, we notice that they are rather mundane and commonplace.

15. Future Scope

Future work Image captioning has become an important problem in recent days due to the exponential growth of images in social media and the internet. This report discusses the various research in image retrieval used in the past and it also highlights the various techniques and methodology used in the research. As feature extraction and similarity calculation in images are challenging in this domain, there is a tremendous scope of possible research in the future. Current image retrieval systems use similarity calculation by making use of features such as color, tags, IMAGE RETRIEVAL USING IMAGE CAPTIONING 54 histogram, etc. There cannot be completely accurate results as these methodologies do not depend on the context of the image. Hence, a complete research in image retrieval making use of context of the images such as image captioning will facilitate to solve this problem in the future. This project can be further enhanced in future to improve the identification of classes which has a lower precision by training it with more image captioning datasets. This methodology can also be combined with previous image retrieval methods such as histogram, shapes, etc. and can be checked if the image retrieval results get better

16. Conclusion

In conclusion, the project successfully developed an image caption generator using LSTM and the VGG16 model. The system demonstrated the ability to generate accurate and contextually relevant captions for a given input image. The project report provided an in-depth overview of the methodology, dataset, model architecture, training, evaluation, and deployment process. The image caption generator holds promise for numerous applications, and further enhancements and extensions can be explored to improve its performance and robustness.

17. References

[1] https://www.kaggle.com/code/zohaib123/image-caption-generator-using-cnn-and-lstm
[2] https://www.kaggle.com/code/hsankesara/image-captioning

[3] https://www.kaggle.com/code/basel99/image-caption-generator
[4]https://www.analyticsvidhya.com/blog/2021/12/step-by-step-guide-to-build-image-caption-generator-using-deep-learning/

18. Appendix

Github source: https://github.com/smartinternz02/SI-GuidedProject-612840-1699352146/tree/main/code