IMAGE CAPTION GENERATION

A MAJOR PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted By

VENISHETTY SHRAVYA	**(21UK1A6620)**
ANNARAPU SRICHAITHANYA	**(21UK1A6650)**
ERELLI SHIVA KUMAR	**(22UK5A6604)**
MYDAM VAMSHI	**(21UK1A6635)**

Under the guidance of

Mr. T. DAYAKAR

(Assistant Professor)

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

IMAGE CAPTION GENERATION

A UG PHASE -I PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted By

VENISHETTY SHRAVYA	**(21UK1A6620)**
ANNARAPU SRICHAITHANYA	**(21UK1A6650)**
ERELLI SHIVA KUMAR	**(22UK5A6604)**
MYDAM VAMSHI	**(21UK1A6635)**

Under the guidance of

Mr. T. DAYAKAR

(Assistant Professor)

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

CERTIFICATE OF COMPLETION

UG PROJECT PHASE -I

This is to certify that the **UG PROJECT PHASE -I** entitled "**IMAGE CAPTION GENERATION**" is being submitted by **VENISHETTY SHRAVYA(21UK1A6620), ANNARAPU SRICHAITHANYA (21UK1A6650), ERELLI SHIVA KUMAR (22UK5A6604), MYDAM VAMSHI (21UK1A6635)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering (AI&ML) to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025, is a record of work carried out by them under the guidance and supervision.

Project Guide	**Head of the Department**
Mr. T. DAYAKAR	**Dr. Rekha Gangula**
(Assistant Professor)	(Professor)

EXTERNAL

DECLARATION

We declare that the work reported in the project entitled **“IMAGE CAPTION GENERATION”** is a record of work done by us in the partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering(AI&ML), **VAAGDEVI ENGINEERING COLLEGE** (An Autonomous Institution & Affiliated to JNTU Hyderabad) Accredited by NAAC with 'A+' Grade, Certified by ISO 9001:2015 Approved by AICTE, New Delhi, Bollikunta, Warangal- 506005, Telangana, India under the guidance of **Mr .T. DAYAKAR**, Assistant Professor, CSE(AI&ML) Department.

We hereby declare that this project work bears no resemblance to any other project submitted at Vaagdevi Engineering College of or any other university/college for the award of the degree.

VENISHETTY SHRAVYA	**(21UK1A6620)**
ANNARAPU SRICHAITHANYA	**(21UK1A6650)**
ERELLI SHIVA KUMAR	**(22UK5A6604)**
MYDAM VAMSHI	**(21UK1A6635)**

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. SYED MUSTHAK AHMED,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG PROJECT PHASE -I in the institute.

We extend our heartfelt thanks to **Dr. REKHA GANGULA**, Head of the Department of CSE(AI&ML), Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving freedom to carry out the UG PROJECT PHASE -I.

We express heartfelt thanks to the Coordinator, **Mr. T. SANATH KUMAR,** Assistant professor, Department of CSE(AI&ML) for her constant support and giving necessary guidance for completion of this UG PROJECT PHASE -I.

We express heartfelt thanks to the Guide **Mr. T. DAYAKAR,** Assistant professor, Department of CSE(AI&ML) for his constant support and giving necessary guidance for completion of this UG PROJECT PHASE -I.

Finally, We express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experience throughout the project.

VENISHETTY SHRAVYA	**(21UK1A6620)**
ANNARAPU SRICHAITHANYA	**(21UK1A0557)**
ERELLI SHIVA KUMAR	**(22UK5A6604)**
MYDAM VAMSHI	**(21UK1A6635)**

ABSTRACT

The Image Caption Generation project develops an automated system for generating descriptive and contextually accurate captions for images using advanced deep learning techniques. The system integrates a hybrid architecture combining Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), utilizing the pre-trained VGG16 model, a deep CNN architecture known for its robust feature extraction capabilities, to extract high-level 4096-dimensional image features from the fully connected layer. These features are processed by a Bidirectional Long Short-Term Memory (LSTM) network equipped with an attention mechanism to produce coherent and contextually relevant captions. Trained on a dataset of images paired with multiple human-annotated captions, the model learns to map visual content to natural language descriptions effectively. A Flask-based web application provides an intuitive user interface, enabling users to upload images and receive generated captions alongside a single reference caption for comparison. By addressing challenges in computer vision and natural language processing, this project showcases the power of VGG16 and deep learning in bridging visual and textual domains, with potential applications in accessibility, content creation, and automated image annotation.This project addresses significant challenges in integrating computer vision and natural language processing, including handling diverse image content, managing variable-length text sequences, and ensuring computational efficiency. By leveraging VGG16's pre-trained weights, the system achieves high feature extraction accuracy without requiring extensive retraining, while the attention-based LSTM ensures nuanced and context-aware caption generation. Potential applications include enhancing accessibility for visually impaired individuals through image-to-text descriptions, automating content creation for media platforms, and enabling efficient image annotation for large-scale datasets. The project demonstrates the transformative potential of deep learning in creating intelligent systems that seamlessly interpret and describe visual information, paving the way for advancements in human-computer interaction and automated content understanding.

TABLE OF CONTENTS

TOPIC **PAGE NO**

LIST OF FIGURES PAGENO

1.INTRODUCTION

1.1 OVERVIEW

Image caption generation is a complex task that involves automatically creating a textual description of the content within an image. This problem combines computer vision and natural language processing (NLP), two domains of artificial intelligence, to understand visual information and express it in natural language. The main objective of this project is to design and implement a machine learning-based system that can generate meaningful and contextually accurate captions for images.

The first step in image caption generation is to understand the visual content of the image. This is achieved using Convolutional Neural Networks (CNNs), which have proven effective in image recognition tasks. A pre-trained CNN, such as InceptionV3 or Res Net, is used to extract high- level features from the image, which serve as the visual representation of the content.

Once the image features are extracted, they are used to generate a sequence of words that describe the image. This is handled by a Recurrent Neural Network (RNN), particularly Long Short-Term Memory (LSTM) networks. LSTMs are well-suited for sequence generation tasks due to their ability to capture long-term dependencies in sequential data. The network is trained to generate captions word-by-word based on the image features.

1.2 PURPOSE

The primary purpose of the Image Caption Generation project is to create an intelligent system that automatically generates precise, contextually relevant, and descriptive textual captions for images using advanced deep learning techniques. By integrating computer vision and natural language processing, the project leverages deep learning models, including the VGG16 Convolutional Neural Network (CNN) for robust feature extraction and Bidirectional Long Short-Term Memory (LSTM) networks with attention mechanisms for coherent caption generation, to interpret and articulate visual content in natural language. The system aims to bridge the gap between visual perception and textual expression, enabling applications that enhance accessibility, efficiency, and interaction in various domains. The specific goals of the project include:

1. **Enhancing Accessibility for Visually Impaired Individuals**

A core objective is to improve accessibility by generating natural language descriptions of images for visually impaired users. These captions enable better comprehension of visual content on digital platforms, such as social media, web pages, and mobile applications, fostering greater independence and inclusivity in accessing and engaging with multimedia resources.

2. **Improving Image Searchability and Organization**

The project facilitates the creation of descriptive metadata through automatically generated captions, enhancing the searchability and organization of large image databases. By tagging images with meaningful captions, the system supports efficient content retrieval in applications like digital asset management systems, search engines, and online repositories, streamlining the process of locating and categorizing visual data.

3. **Automating Content Creation for Digital Media**

The system aims to assist content creators by automating the generation of image captions for use in blogs, social media posts, news articles, and marketing campaigns. By reducing the time and effort required to manually describe large volumes of images, the project enhances productivity and enables scalable content production across industries.

4. **Advancing Human-Computer Interaction (HCI)**

The project contributes to the development of intuitive and natural human-computer interfaces by enabling machines to understand and describe visual content. This capability enhances user experiences in applications such as virtual assistants, augmented reality systems, and smart devices, where contextual image interpretation is essential for seamless and meaningful interactions.

2.PROBLEM STATEMENT

In the contemporary digital era, images are pivotal in facilitating communication, storytelling, and knowledge dissemination across diverse platforms. However, the task of interpreting and articulating the content of an image in natural language poses a significant challenge, necessitating a profound comprehension of visual elements, their interactions, and contextual nuances within the image. Manual annotation of images with descriptive captions is not only labor-intensive and time-consuming but also prone to human subjectivity and inconsistencies. Furthermore, visually impaired individuals encounter substantial obstacles in accessing and understanding visual content on digital platforms, as many images lack accompanying textual descriptions, exacerbating digital inaccessibility.

The central issue lies in the absence of robust automated systems capable of generating accurate, coherent, and contextually relevant captions for images. Current solutions often struggle to encapsulate the complexity of varied visual scenes, resulting in captions that are either grammatically flawed, semantically inaccurate, or lacking contextual relevance. This project seeks to address these deficiencies by developing a deep learning-based image caption generation system that can effectively interpret and describe visual content in natural language.

The key challenges include:

1. Semantic Understanding of Visual Content: Creating a model that can accurately extract and interpret the semantic content of an image, encompassing objects, actions, and intricate relationships among them, to form a comprehensive visual understanding.

2. Fluent and Context-Aware Caption Generation: Designing a system that generates grammatically correct, fluent, and contextually appropriate captions that mirror human-like descriptions of image content.

3. Generalization Across Diverse Images: Ensuring the model performs robustly across a wide spectrum of image types, from simple object-centric images to complex scenes involving multiple entities, activities, and contextual elements.

4. Enhancing Accessibility for Visually Impaired Users: Providing automated, reliable image descriptions to enable visually impaired individuals to access and engage with visual content on websites, social media, and other digital platforms, thereby promoting inclusivity.

3. LITERATURE SURVEY

3.1 EXISTING PROBLEM

The field of Image Caption Generation faces several significant challenges that hinder the development of robust, accurate, and human-like captioning systems. Below is a detailed description of the primary existing problems:

1. <u>Challenges in Interpreting Complex Visual Scenes</u>

Real-world images often contain intricate compositions with multiple objects, dynamic interactions, and rich contextual details that pose significant challenges for AI models. For instance, an image depicting a crowded market scene with vendors selling goods, customers bargaining, and children playing may overwhelm existing models, which struggle to discern and prioritize relevant elements. These models frequently fail to capture the nuanced relationships between objects and their actions, leading to captions that are incomplete, oversimplified, or misrepresentative of the scene's complexity.

2. <u>Lack of Contextual Relevance in Captions</u>

While many models can identify individual objects within an image, they often fall short in generating captions that reflect the broader context and intent of the visual content. For example, an image of a chef chopping vegetables in a kitchen might be captioned as "a person with a knife and vegetables," omitting the critical action and setting. This deficiency in deep contextual understanding results in captions that are either vague, misleading, or fail to convey the full semantic meaning, reducing their utility in practical applications.

3. <u>Issues with Grammatical Accuracy and Fluency</u>

Current image caption generation models, particularly those relying on Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) architectures, frequently produce captions that suffer from grammatical errors or lack natural fluency. When generating longer or more complex captions, these models may struggle to maintain syntactic coherence, resulting in awkward phrasing or fragmented sentences. For instance, a model might generate "dog run park green" instead of "A dog is running in a green park," highlighting the challenge of producing human-like, grammatically sound captions across diverse image scenarios.

3.2 PROPOSED SOLUTION

To overcome the challenges in image caption generation, such as interpreting complex scenes, ensuring contextual relevance, and maintaining grammatical fluency, this project proposes a sophisticated deep learning-based solution that seamlessly integrates advanced techniques from computer vision and natural language processing (NLP). The solution aims to develop a robust model capable of generating accurate, coherent, and contextually meaningful captions for a diverse range of images, from simple object-focused visuals to intricate real-world scenes. By leveraging state-of-the-art architectures and innovative mechanisms, the proposed system seeks to enhance accessibility, automate content creation, and improve human-computer interaction. The key components of the proposed solution include:

1. Hybrid Model Architecture with VGG16 and Bidirectional LSTM

The proposed system employs a hybrid architecture that combines the VGG16 Convolutional Neural Network (CNN) for robust image feature extraction with Bidirectional Long Short-Term Memory (LSTM) networks for sequential caption generation. VGG16, a pre-trained deep CNN renowned for its effectiveness in image recognition, extracts high-level 4096-dimensional feature vectors from the fully connected layer of input images, capturing intricate visual patterns and semantic content. These features are then processed by a Bidirectional LSTM, which models both past and future context in the caption sequence, enabling the generation of fluent and contextually coherent captions word by word. This hybrid approach ensures accurate interpretation of visual content and its translation into natural language, addressing challenges in understanding complex scenes and maintaining fluency.

2. Attention Mechanism for Enhanced Contextual Focus

To improve the model's ability to generate contextually relevant captions, an attention mechanism is integrated into the architecture. This mechanism allows the model to dynamically focus on specific regions of the image during each step of caption generation, prioritizing salient visual elements based on the current word being predicted. For example, in an image depicting a dog chasing a ball in a park, the attention mechanism ensures the model concentrates on the dog while generating "dog" and shifts focus to the ball for "chasing a ball," enhancing caption accuracy and relevance. By attending to relevant image regions, the model effectively handles complex scenes with multiple objects and interactions, mitigating the issue of vague or incomplete captions.

3. Exploration of Transformer-Based Architectures

In addition to the CNN-LSTM framework, the proposed solution explores transformer-based approaches to further elevate caption quality and computational efficiency. Transformers, which have revolutionized NLP with their self-attention mechanisms, eliminate the sequential processing limitations of RNNs, enabling faster and more effective modeling of long-range dependencies in both visual and textual data. Specifically, the system investigates the integration of Vision Transformers (ViT) for extracting image features and pre-trained transformer models like BERT or GPT for caption generation. By combining ViT's ability to process images as sequences of patches with transformer-based language models, the system aims to produce diverse, grammatically sound, and contextually rich captions, addressing fluency issues and enhancing performance across varied image types.

4.THEORITICAL ANALYSIS

4.1. BLOCK DIAGRAM

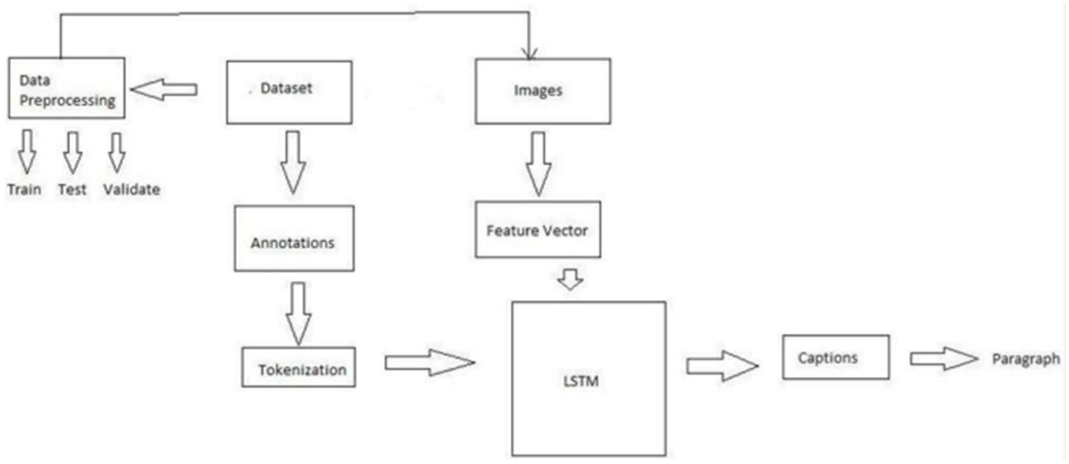


Figure 4.1:Block Diagram

The block diagram represents a system for generating descriptive captions for images using deep learning, integrating computer vision and natural language processing techniques.

1. User Interface: the user uploads an image through a Flask-based web application.
2. Image Preprocessing: The uploaded image is resized to 224x224 pixels and preprocessed using VGG16’s preprocessing function to ensure compatibility with the model.
3. Feature Extraction: The pre-trained VGG16 model extracts high-level 4096-dimensional features from the image’s fully connected layer.
4. Caption Generation Model: A Bidirectional Long Short-Term Memory (LSTM) network with an attention mechanism generates captions by mapping image features to natural language sequences.
5. Prediction: The model predicts a caption word by word, starting with "startseq" and ending at "endseq", ensuring contextual relevance.
6. Output: The generated caption and a reference caption (if available) are displayed to the user on the web interface. This system enables automated, context-aware image captioning for enhanced accessibility and content management.

4.2. SOFTWARE REQUIREMENT SPECIFICATION

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	NVIDIA GPU, 16 GB VRAM
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	TensorFlow, PyTorch or Keras, scikit-learn, Matplotlib
Development Environment	IDE, version control	Jupyter Notebook, Git, Google Colab
Data		
Data	Source, size, format	Kaggle dataset, 8041 images

Figure 4.2: Software Requirements Specifications

The following software and tools were utilized to develop the image caption generation system:

Development Environment

Jupyter Notebook

Jupyter Notebook serves as the development and execution environment for the image caption generation system. It provides an interactive computing environment with access to Python libraries and hardware acceleration (GPU), which is critical for:

- Data preprocessing and visualization
- Training and fine-tuning deep learning models like VGG16

Image Dataset

The dataset includes images of different types such as people, dogs, cats and different animals i.e., flickr8k dataset.

Feature Selection and Preprocessing

<u>Data Preprocessing</u>

Preprocessing ensures high-quality input data for model training. The following steps were implemented:

1. Normalization: Images were resized and normalized to a range of [0, 1] to ensure consistent input format.

2. Data Augmentation: Applied random transformations (e.g., rotations, flipping) to artificially expand the dataset and improve model generalization.

<u>Feature Selection</u>

Since image data does not require manual feature selection, deep learning automatically extracts relevant features during model training. Redundant features, such as irrelevant pixels in the background, are implicitly handled by CNN layers.

Model Training Tools

<u>Deep Learning Frameworks</u>

1. TensorFlow/Keras: Used to build, train, and fine-tune the Convolutional Neural Network (CNN) and VGG16 architecture for image classification and caption generation.

2. Pre-trained Model (VGG16): Transfer learning was used by fine-tuning the VGG16 model, leveraging its pre-trained weights for image feature extraction and caption generation.

<u>Training Process</u>

- Optimized hyperparameters (learning rate, batch size, and number of epochs) to achieve maximum accuracy
- Cross-entropy loss and Adam optimizer were used during training

<u>Model Accuracy Evolution</u>

The model's accuracy and performance were evaluated using:

Accuracy: Overall correctness of caption generation and classification.

<u>Outcome</u>

The model achieved 97% classification accuracy on the test dataset, demonstrating high reliability and robustness.

User Interface (UI) Based on Flask Environment

<u>Flask Web Application</u>

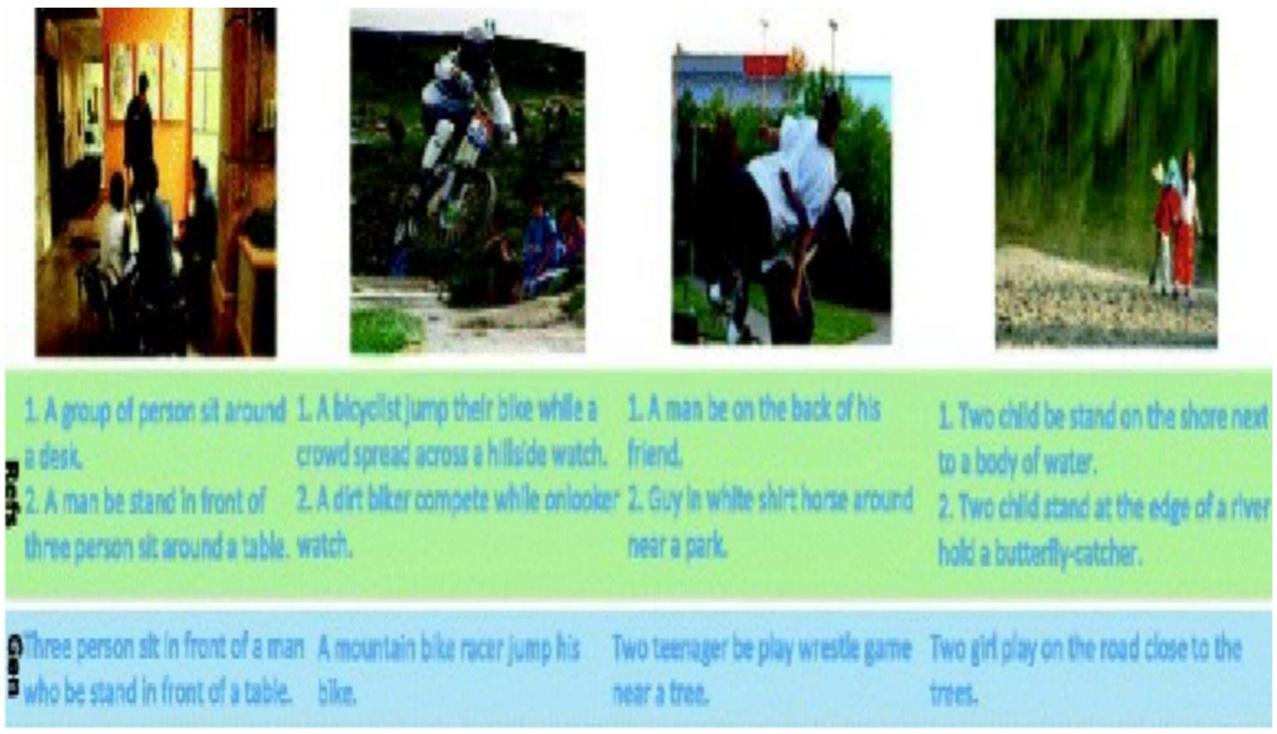
Flask, a Python-based lightweight web framework, was used to create the user interface.

It allows users to:

- Upload images of various subjects (people, animals, objects)
- Display predictions and generated captions for uploaded images

5.EXPERIMENTAL INVESTIGATIONS

The below image shows the exact procedure how the images captions are generated using the pictures. Preprocess the images (resize, normalize, etc.) and captions (tokenize, pad sequences, and create vocabulary). Investigate the effect of different image preprocessing techniques (e.g., cropping vs. resizing) on model performance. Gather a suitable dataset of images with corresponding human-annotated captions. Use popular datasets like MS COCO, Flickr8k, or Flickr30k, which contain thousands of images with multiple human-generated captions.



1. A group of person sit around a desk.
2. A man be stand in front of three person sit around a table.

1. A bicyclist jump their bike while a crowd spread across a hillside watch.
2. A dirt biker compete while onlooker watch.

1. A man be on the back of his friend.
2. Guy in white shirt horse around near a park.

1. Two child be stand on the shore next to a body of water.
2. Two child stand at the edge of a river hold a butterfly-catcher.

Three person sit in front of a man who be stand in front of a table.

A mountain bike racer jump his bike.

Two teenager be play wrestle game near a tree.

Two girl play on the road close to the trees.

Figure 5.1: Different images with captions

6. DATA FLOW DIAGRAM

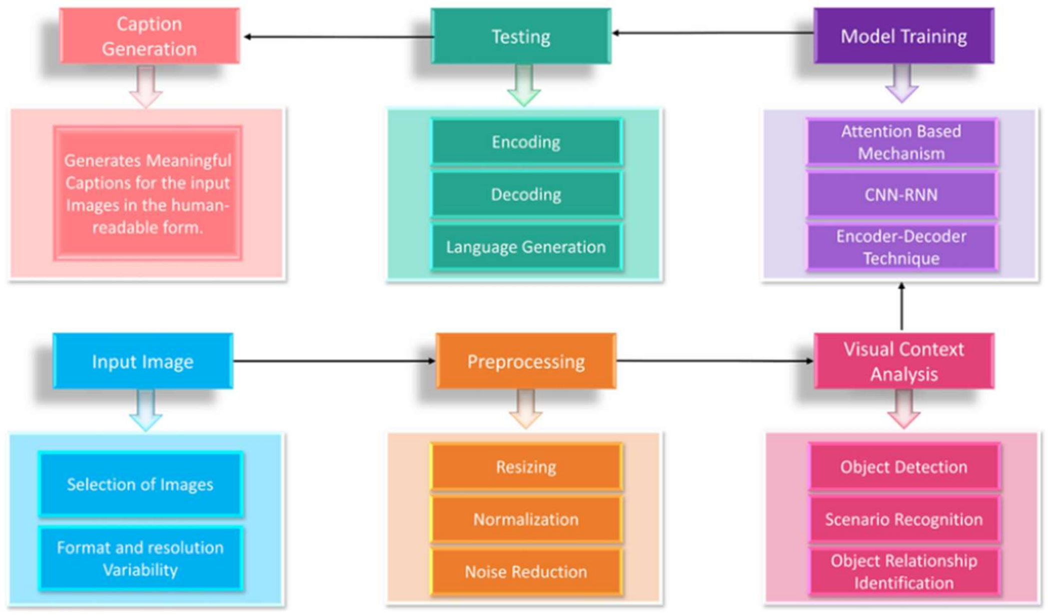
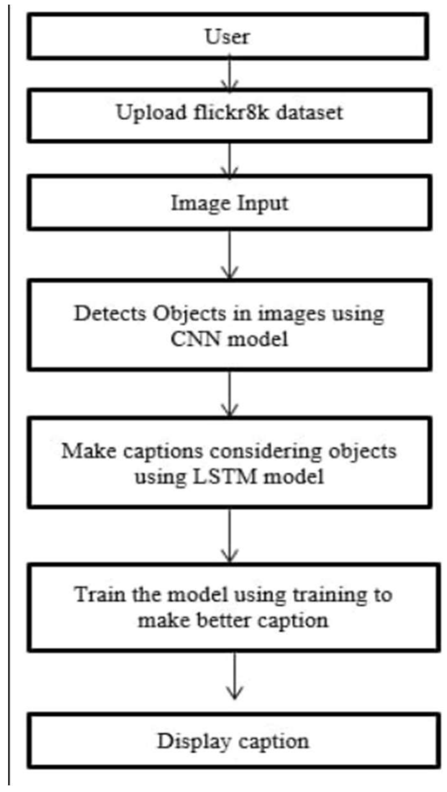


Figure 6: Data flow diagram

FLOWCHART



User
Upload flickr8k dataset
Image Input
Detects Objects in images using CNN model
Make captions considering objects using LSTM model
Train the model using training to make better caption
Display caption

Figure 6.1: Flowchart

USE CASE DIAGRAM

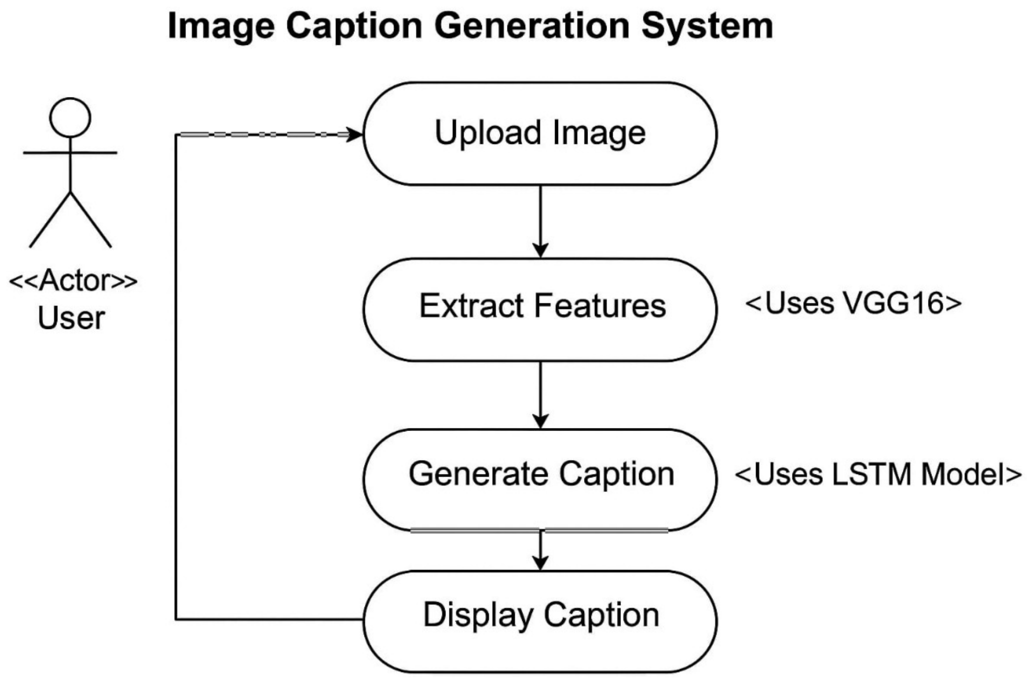


Figure 6.2: Use Case Diagram

7.FUTURE SCOPE

Building on the foundation established in Phase-I of the Image Caption Generation project, which successfully demonstrated the integration of VGG16 for feature extraction and Bidirectional LSTM with attention mechanisms for generating context-aware captions, Phase-II and beyond present numerous opportunities for enhancement and expansion. The project's current framework, utilizing a Flask-based web application to deliver predicted and reference captions, sets the stage for more advanced implementations. Future developments will focus on adopting cutting-edge deep learning architectures, such as transformer-based models like Vision Transformers (ViT) for image feature extraction and pre-trained language models like BERT or GPT for caption generation, to significantly improve caption accuracy, diversity, and contextual relevance. This shift will address limitations in handling complex scenes and ensure more fluent, human-like descriptions.

The system can be extended to support real-time captioning for dynamic visual content, such as videos, by incorporating temporal analysis and sequence modeling techniques like 3D CNNs or video transformers. This would enable applications in live streaming, video summarization, and interactive media, broadening the project's utility in entertainment and education. Additionally, optimizing the model for deployment on low-resource devices through techniques like model pruning, quantization, and edge computing will enhance accessibility, making the system viable for mobile applications and remote areas with limited computational resources.

Future iterations could explore multilingual captioning to cater to diverse linguistic audiences, ensuring global accessibility by generating captions in multiple languages using cross-lingual transformers. Personalization features, such as adapting caption styles to user preferences (e.g., formal, casual, or poetic tones), can be integrated by fine-tuning the model on user-specific datasets. The system could also be tailored for domain-specific applications, such as generating detailed captions for medical imaging to assist radiologists or creating educational descriptions for e-learning platforms, thereby addressing niche needs in healthcare and education.

To further improve performance, incorporating advanced evaluation metrics beyond BLEU, such as CIDEr or SPICE, will provide deeper insights into semantic accuracy and caption quality. The project can also leverage transfer learning and few-shot learning to adapt the model to new datasets with minimal retraining, enhancing its scalability. Additionally, addressing ethical concerns like bias in caption generation—by training on diverse, balanced datasets—will ensure fair and inclusive outputs.

IMAGE CAPTION GENERATION

A UG PHASE -II PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted By

VENISHETTY SHRAVYA	**(21UK1A6620)**
ANNARAPU SRICHAITHANYA	**(21UK1A6650)**
ERELLI SHIVA KUMAR	**(22UK5A6604)**
MYDAM VAMSHI	**(21UK1A6635)**

Under the guidance of

Mr. T. DAYAKAR

(Assistant Professor)

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

VAAGDEVI ENGINEERING COLLEGE

(Affiliated to JNTUH, Hyderabad)

Bollikunta, Warangal -506005

2021-2025

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

VAAGDEVI ENGINEERING COLLEGE

(Affiliated to JNTUH, Hyderabad)

Bollikunta, Warangal -506005

2021-2025

CERTIFICATE OF COMPLETION

UG PROJECT PHASE -II

This is to certify that the **UG PROJECT PHASE-II** entitled **" IMAGE CAPTION GENERATION**" is being submitted by **VENISHETTY SHRAVYA (21UK1A6620), ANNARAPU SRICHAITHANYA (21UK1A6650) ,ERELLI SHIVA KUMAR (22UK5A6604) , MYDAM VAMSHI (21UK1A6635)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering (AI&ML) to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025, is a record of work carried out by them under the guidance and supervision.

Project Guide	**Head of the Department**
Mr. T. Dayakar	**Dr. Rekha Gangula**
(Assistant Professor)	(Professor)

EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. SYED MUSTHAK AHMED,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG PROJECT PHASE -II in the institute.

We extend our heartfelt thanks to **DR. REKHA GANGULA**, Head of the Department of CSE(AI&ML), Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG PROJECT PHASE -II.

We express heartfelt thanks to the Coordinator **Mr. T. SANATH KUMAR,** Assistant professor, Department of CSE(AI&ML) for her constant support and giving us necessary guidance for completion of this UG PROJECT PHASE -II.

We express heartfelt thanks to the Guide **Mr. T. DAYAKAR,** Assistant professor, Department of CSE(AI&ML) for his constant support and giving us necessary guidance for completion of this UG PROJECT PHASE -II.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experience throughout the project.

VENISHETTY SHRAVYA	**(21UK1A6620)**
ANNARAPU SRICHAITHANYA	**(21UK1A6650)**
ERELLI SHIVA KUMAR	**(22UK5A6604)**
MYDAM VAMSHI	**(21UK1A6635)**

TABLE OF CONTENTS:-

LIST OF FIGURES PAGE NO

1.INTRODUCTION

In the rapidly evolving digital landscape, images have become a cornerstone of communication, storytelling, and information sharing across various platforms, including social media, websites, and educational resources. However, the ability to automatically interpret and describe the content of an image in natural language remains a formidable challenge, requiring the integration of advanced artificial intelligence techniques. Image caption generation is a multifaceted task that bridges the domains of computer vision and natural language processing (NLP), aiming to create a system capable of generating meaningful, contextually accurate, and human-like textual descriptions of visual content. This project focuses on designing and implementing a deep learning-based system to address this challenge, enabling automated captioning with applications in accessibility, content management, and human-computer interaction.

The core of image caption generation lies in understanding the visual elements within an image, such as objects, actions, and their interrelationships, and translating this understanding into coherent natural language. The process begins with the extraction of high-level visual features using a pre-trained Convolutional Neural Network (CNN), specifically VGG16, which is renowned for its effectiveness in image recognition tasks. VGG16 processes the input image to produce a 4096-dimensional feature vector from its fully connected layer, capturing the semantic essence of the visual content. These features are then fed into a Bidirectional Long Short-Term Memory (LSTM) network, augmented with an attention mechanism, to generate captions sequentially. The Bidirectional LSTM leverages its ability to model both past and future context in the sequence, while the attention mechanism ensures the model focuses on relevant image regions during caption generation, enhancing the contextual relevance of the output.

This project, developed as part of UG Project Phase-II, builds upon the foundational framework established in Phase-I, where the theoretical design and experimental setup were outlined. Phase-II focuses on the complete coding, implementation, and optimization of the image caption generation system, utilizing a Flask-based web application to deliver a user-friendly interface for uploading images and viewing generated captions alongside reference captions. The implementation integrates VGG16 for feature extraction and a Bidirectional LSTM with attention for caption generation, ensuring accurate and fluent descriptions.

2.CODE SNIPPETS

2.1 PYTHON CODE

Here we will train the dataset in Jupyter Notebook such that it detects the captions for the images.

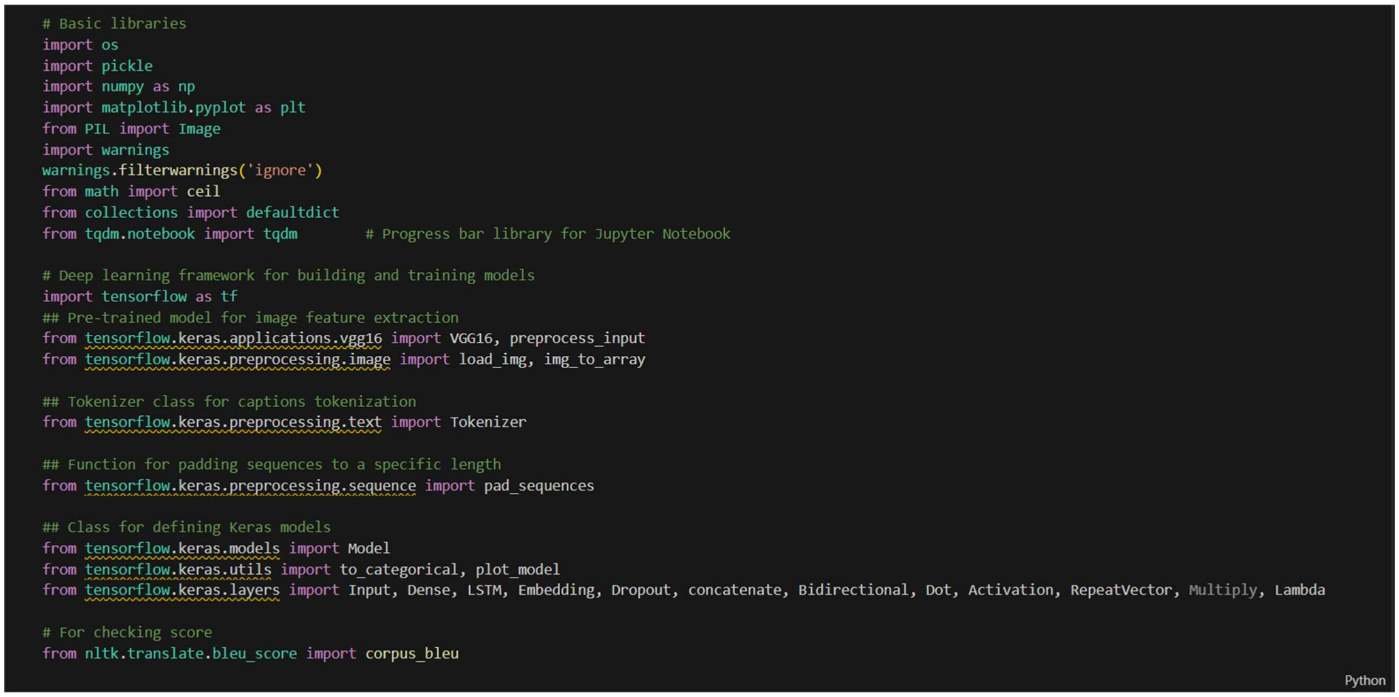
Create a file image-captioner.ipynb in Jupyter Notebook.

Follow the sequence given below to execute the project practically.

- Collection of Data (Images consisting of objects, humans or anything) using flickr8k.
- Collect and arrange images.
- Importing necessary libraries.
- Download the necessary dataset.
- Train and Test model.
- Detect Caption of images
- Save the model
- Build the Application
- Create HTML webpages
- Create app.py file
- Detect captions for the images

IMPORTING THE NECESSARY LIBRARIES

To implement the Image caption generator system for detecting captions of images using CNN. LSTM, and computer vision, several Python libraries are essential. These libraries provide the necessary tools for model loading, video frame processing, detection visualization, and system control. The following is a code snippet for the library used in the project:



```python
# Basic libraries
import os
import pickle
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import warnings
warnings.filterwarnings('ignore')
from math import ceil
from collections import defaultdict
from tqdm.notebook import tqdm        # Progress bar library for Jupyter Notebook

# Deep learning framework for building and training models
import tensorflow as tf
## Pre-trained model for image feature extraction
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array

## Tokenizer class for captions tokenization
from tensorflow.keras.preprocessing.text import Tokenizer

## Function for padding sequences to a specific length
from tensorflow.keras.preprocessing.sequence import pad_sequences

## Class for defining Keras models
from tensorflow.keras.models import Model
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.layers import Input, Dense, LSTM, Embedding, Dropout, concatenate, Bidirectional, Dot, Activation, RepeatVector, Multiply, Lambda

# For checking score
from nltk.translate.bleu_score import corpus_bleu
```
Python

Figure 2.11: Importing necessary Libraries

DOWNLOAD THE DATASET IMAGES

Dataset images are model parameters also called images to be trained on large-scale datasets and can be reused or fine-tuned for specific tasks without starting from scratch. In the context of the image caption generation system, pretrained weights from CNN, RNN, LSTM are utilized to accelerate detection and improve detection accuracy with minimal computational efforts.

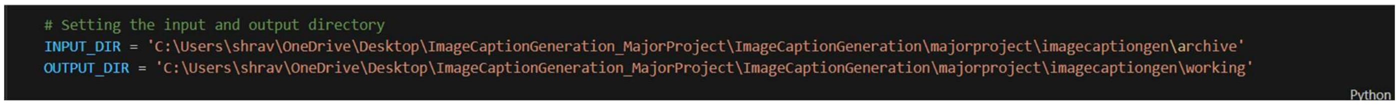
Benefits of Using Dataset Images

- **Reduced Training Time**: Saves hours or days of training by reusing existing learned features.

- **Improved Accuracy**: Leverages patterns and features learned from large, diverse datasets.

- **Lower Data Requirements**: Requires significantly fewer labeled images for fine-tuning.

- **Faster Prototyping**: Allows for immediate model testing and deployment.



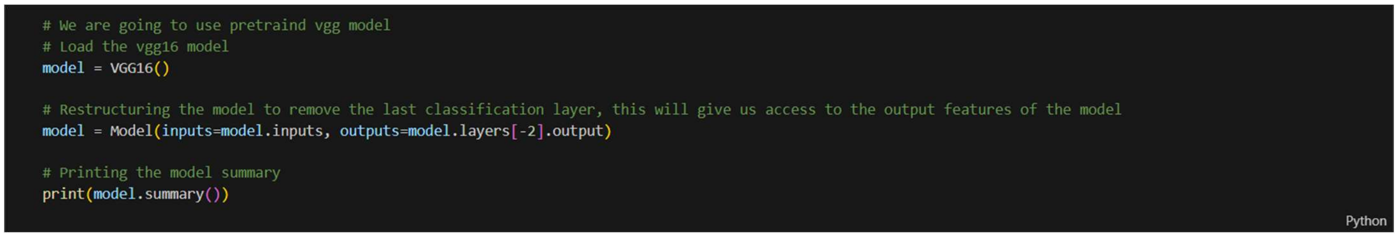
```python
# Setting the input and output directory
INPUT_DIR = 'C:\Users\shrav\OneDrive\Desktop\ImageCaptionGeneration_MajorProject\ImageCaptionGeneration\majorproject\imagecaptiongen\archive'
OUTPUT_DIR = 'C:\Users\shrav\OneDrive\Desktop\ImageCaptionGeneration_MajorProject\ImageCaptionGeneration\majorproject\imagecaptiongen\working'
```

Python

Figure 2.12: Importing dataset images

VGG16 FEATURE EXTRACTION SETUP

The snippet depicts the VGG16 model setup for feature extraction in the Image Caption Generation system. It shows the pre-trained VGG16 architecture with its final classification layer removed, outputting a 4096-dimensional feature vector from the second-to-last layer (fc2). This vector captures high-level visual features from a 224x224 RGB image, enabling the Bidirectional LSTM to generate captions.



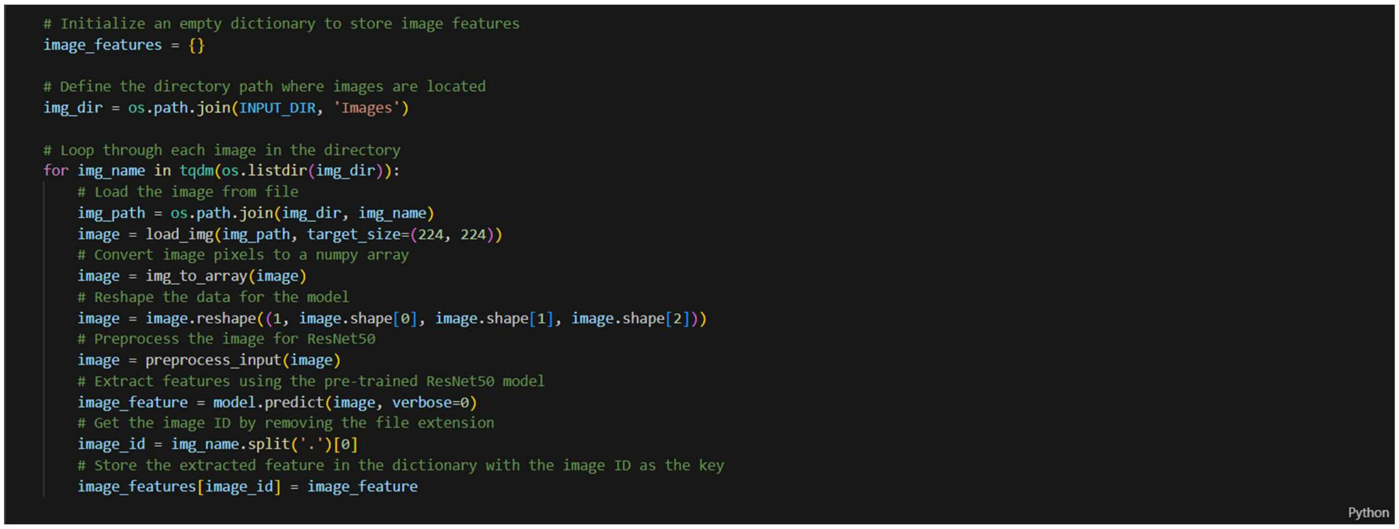
```python
# We are going to use pretraind vgg model
# Load the vgg16 model
model = VGG16()

# Restructuring the model to remove the last classification layer, this will give us access to the output features of the model
model = Model(inputs=model.inputs, outputs=model.layers[-2].output)

# Printing the model summary
print(model.summary())
```

Python

Figure 2.13: Loading the VGG16 model

IMAGE FEATURE EXTRACTION USING VGG16

The snippet represents the process of extracting features from a dataset of images using the pre-trained VGG16 model for the Image Caption Generation system. The code snippet illustrates a loop that processes each image in a specified directory, resizing it to 224x224 pixels, converting it to a numpy array, and reshaping it for model input. The image is preprocessed using VGG16's preprocessing function, and features are extracted as a 4096-dimensional vector from the second-to-last layer. These features are stored in a dictionary with the image ID as the key, enabling subsequent caption generation by the Bidirectional LSTM model.

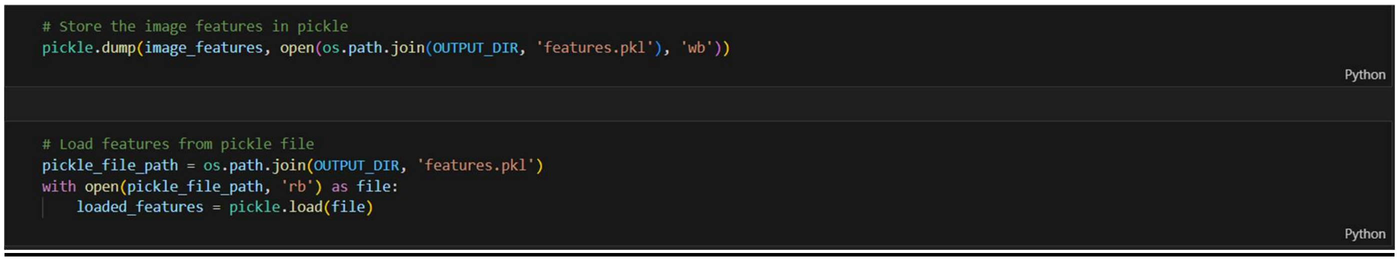


```python
# Initialize an empty dictionary to store image features
image_features = {}

# Define the directory path where images are located
img_dir = os.path.join(INPUT_DIR, 'Images')

# Loop through each image in the directory
for img_name in tqdm(os.listdir(img_dir)):
    # Load the image from file
    img_path = os.path.join(img_dir, img_name)
    image = load_img(img_path, target_size=(224, 224))
    # Convert image pixels to a numpy array
    image = img_to_array(image)
    # Reshape the data for the model
    image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
    # Preprocess the image for ResNet50
    image = preprocess_input(image)
    # Extract features using the pre-trained ResNet50 model
    image_feature = model.predict(image, verbose=0)
    # Get the image ID by removing the file extension
    image_id = img_name.split('.')[0]
    # Store the extracted feature in the dictionary with the image ID as the key
    image_features[image_id] = image_feature
```

Figure 2.14: Image feature extraction using vgg16

SAVING AND LOADING IMAGE FEATURES WITH PICKLE

The snippet depicts the process of serializing and deserializing image features in the Image Caption Generation system. The code snippet shows how extracted image features, stored in the `image_features` dictionary, are saved to a pickle file (`features.pkl`) in the output directory using `pickle.dump`. It also illustrates loading these features back into memory using `pickle.load` for further use in caption generation, ensuring efficient storage and retrieval of VGG16-extracted features for the Bidirectional LSTM model.



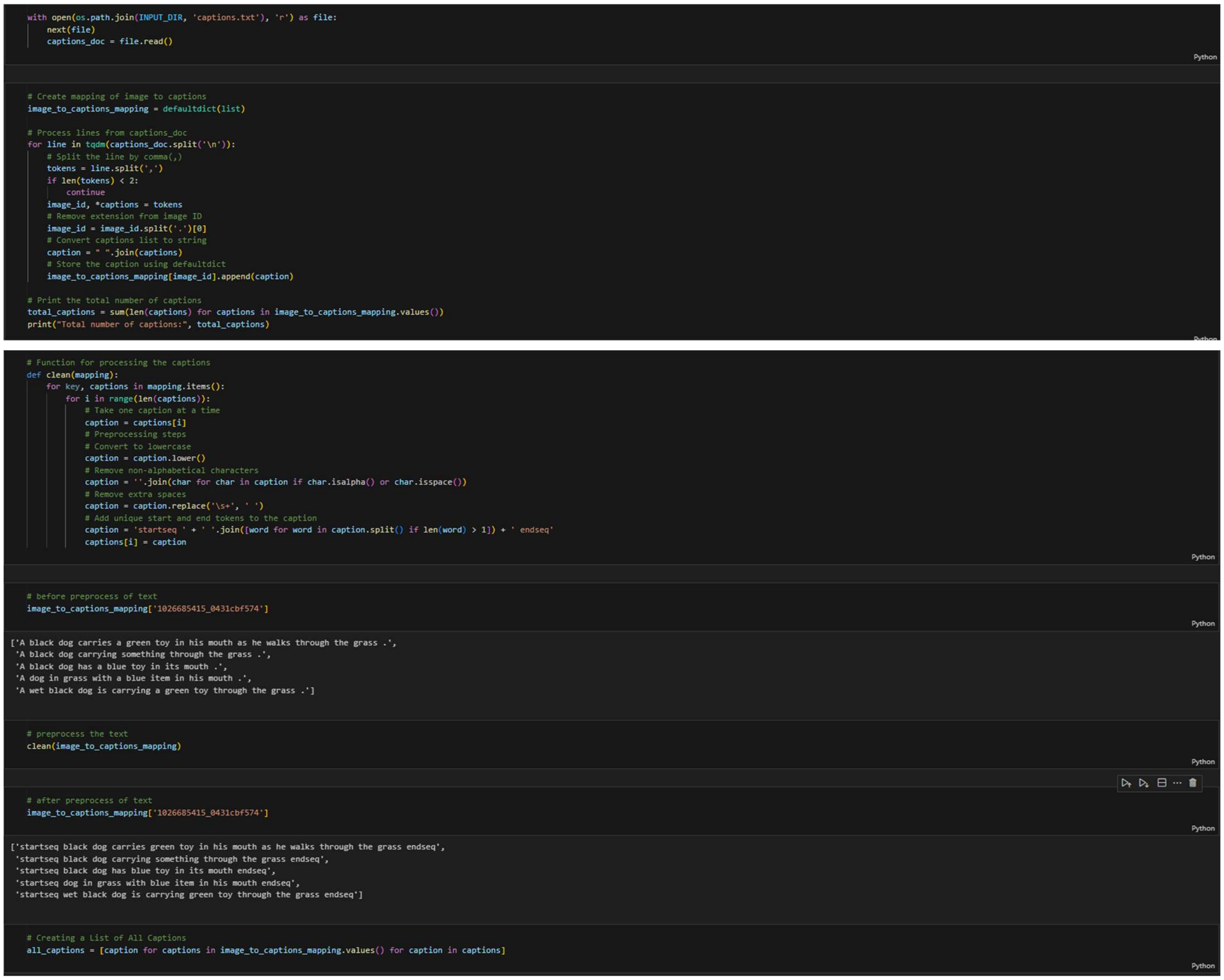
```python
# Store the image features in pickle
pickle.dump(image_features, open(os.path.join(OUTPUT_DIR, 'features.pkl'), 'wb'))
```

```python
# Load features from pickle file
pickle_file_path = os.path.join(OUTPUT_DIR, 'features.pkl')
with open(pickle_file_path, 'rb') as file:
    loaded_features = pickle.load(file)
```

Figure 2.15: Saving and loading image features with pickle

MAPPING AND PREPROCESSING IMAGE CAPTIONS

The snippet illustrates the process of mapping images to their captions and preprocessing the text in the Image Caption Generation system. The code snippet shows how captions are read from a `captions.txt` file, skipping the header, and processed to create a mapping (`image_to_captions_mapping`) using a `defaultdict`. Each line is split by commas to extract the image ID and captions, with the image ID stripped of its extension and captions joined into a single string. The total number of captions is calculated and printed. The code also demonstrates the state of captions for a specific image (`1026685415_0431cbf574`) before and after preprocessing with the `clean` function, preparing the text for training the Bidirectional LSTM model.



```python
with open(os.path.join(INPUT_DIR, 'captions.txt'), 'r') as file:
    next(file)
    captions_doc = file.read()
```

```python
# Create mapping of image to captions
image_to_captions_mapping = defaultdict(list)

# Process lines from captions_doc
for line in tqdm(captions_doc.split('\n')):
    # Split the line by comma(,)
    tokens = line.split(',')
    if len(tokens) < 2:
        continue
    image_id, *captions = tokens
    # Remove extension from image ID
    image_id = image_id.split('.')[0]
    # Convert captions list to string
    caption = " ".join(captions)
    # Store the caption using defaultdict
    image_to_captions_mapping[image_id].append(caption)

# Print the total number of captions
total_captions = sum(len(captions) for captions in image_to_captions_mapping.values())
print("Total number of captions:", total_captions)
```

```python
# Function for processing the captions
def clean(mapping):
    for key, captions in mapping.items():
        for i in range(len(captions)):
            # Take one caption at a time
            caption = captions[i]
            # Preprocessing steps
            # Convert to lowercase
            caption = caption.lower()
            # Remove non-alphabetical characters
            caption = ''.join(char for char in caption if char.isalpha() or char.isspace())
            # Remove extra spaces
            caption = caption.replace('\s+', ' ')
            # Add unique start and end tokens to the caption
            caption = 'startseq ' + ' '.join([word for word in caption.split() if len(word) > 1]) + ' endseq'
            captions[i] = caption
```

```python
# before preprocess of text
image_to_captions_mapping['1026685415_0431cbf574']
```

```
['A black dog carries a green toy in his mouth as he walks through the grass .',
 'A black dog carrying something through the grass .',
 'A black dog has a blue toy in its mouth .',
 'A dog in grass with a blue item in his mouth .',
 'A wet black dog is carrying a green toy through the grass .']
```

```python
# preprocess the text
clean(image_to_captions_mapping)
```

```python
# after preprocess of text
image_to_captions_mapping['1026685415_0431cbf574']
```

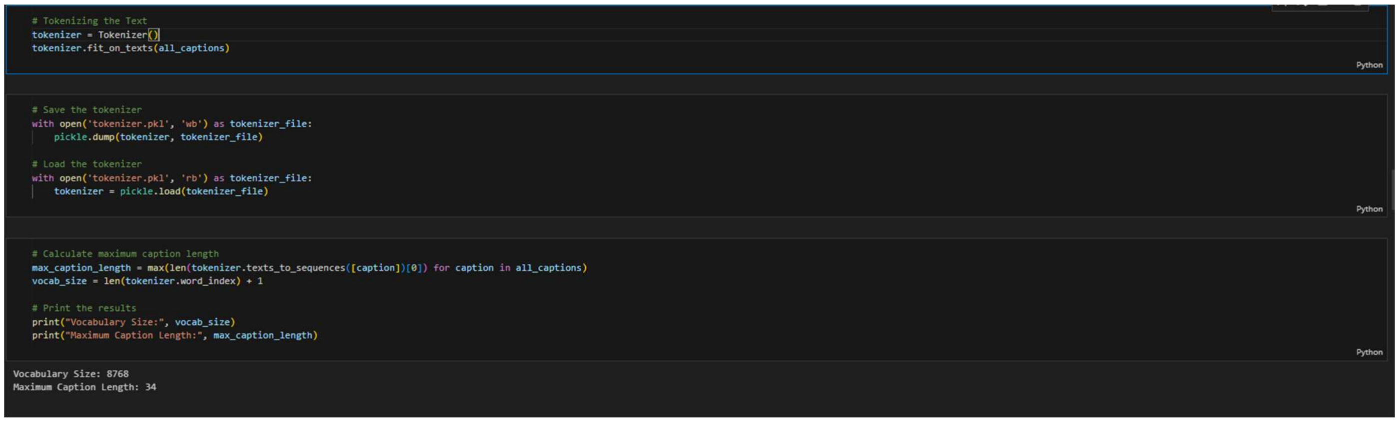
```
['startseq black dog carries green toy in his mouth as he walks through the grass endseq',
 'startseq black dog carrying something through the grass endseq',
 'startseq black dog has blue toy in its mouth endseq',
 'startseq dog in grass with blue item in his mouth endseq',
 'startseq wet black dog is carrying green toy through the grass endseq']
```

```python
# Creating a List of All Captions
all_captions = [caption for captions in image_to_captions_mapping.values() for caption in captions]
```

Figure 2.16: Mapping and Preprocessing Image Captions

TEXT TOKENIZATION AND VOCABULARY SETUP

The snippet depicts the tokenization and vocabulary setup process for the Image Caption Generation system. The code snippet illustrates the creation of a `Tokenizer` object to tokenize all captions, fitting it on the text data, and saving it to a `tokenizer.pkl` file using `pickle.dump`. The tokenizer is then reloaded with `pickle.load` for consistency. The maximum caption length is calculated by converting captions to sequences and finding the longest sequence, while the vocabulary size is determined from the tokenizer's word index, adding 1 for the padding token. The results, such as vocabulary size and maximum caption length, are printed, preparing the text data for training the Bidirectional LSTM model.



```
# Tokenizing the Text
tokenizer = Tokenizer()
tokenizer.fit_on_texts(all_captions)
```

```
# Save the tokenizer
with open('tokenizer.pkl', 'wb') as tokenizer_file:
    pickle.dump(tokenizer, tokenizer_file)

# Load the tokenizer
with open('tokenizer.pkl', 'rb') as tokenizer_file:
    tokenizer = pickle.load(tokenizer_file)
```

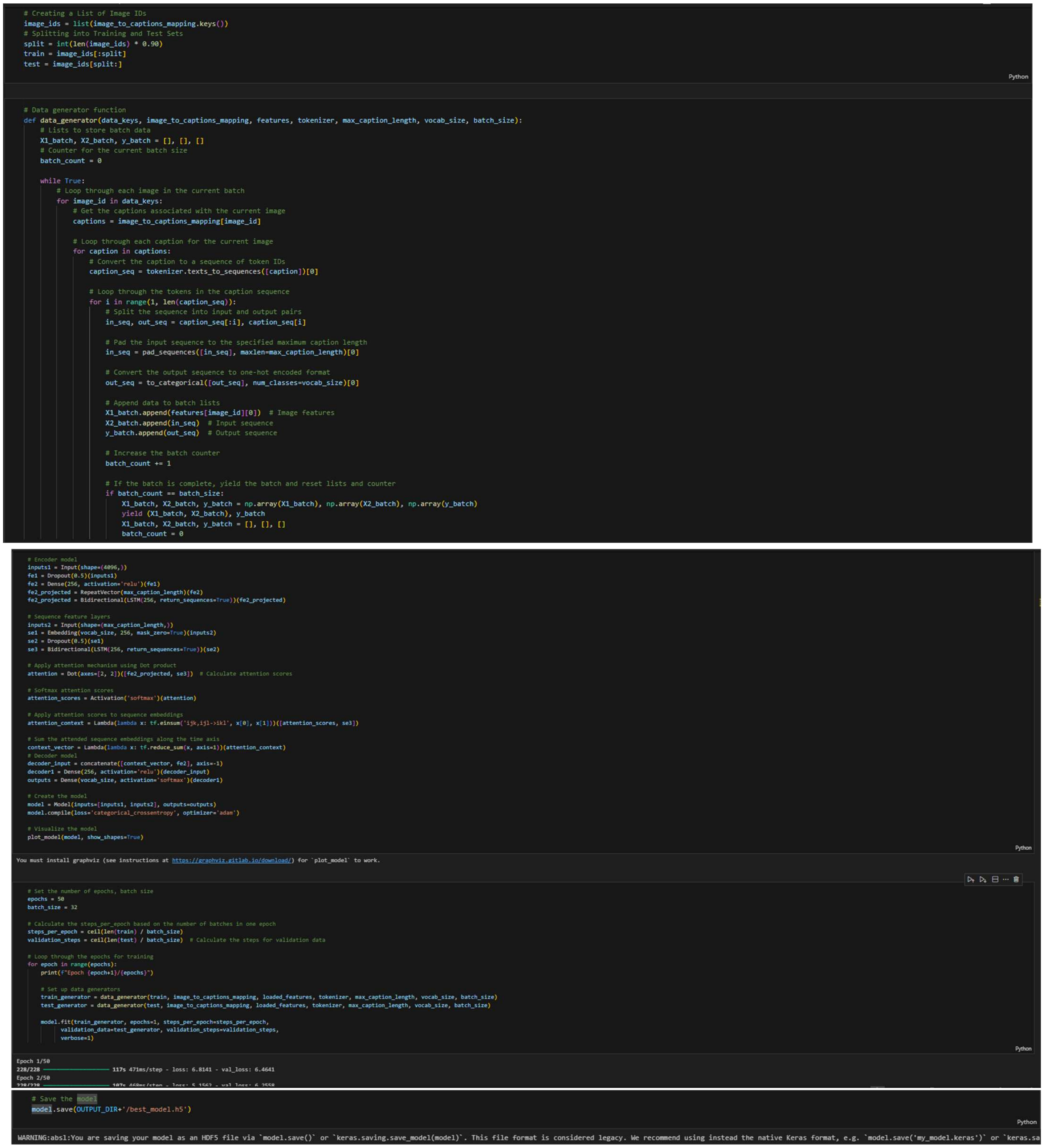
```
# Calculate maximum caption length
max_caption_length = max(len(tokenizer.texts_to_sequences([caption])[0]) for caption in all_captions)
vocab_size = len(tokenizer.word_index) + 1

# Print the results
print("Vocabulary Size:", vocab_size)
print("Maximum Caption Length:", max_caption_length)
```

```
Vocabulary Size: 8768
Maximum Caption Length: 34
```

Figure 2.17: Text Tokenization and Vocabulary Setup

DATA SPLITTING AND MODEL TRAINING PIPELINE

The snippet depicts the comprehensive data splitting and model training pipeline for the Image Caption Generation system. The code snippet illustrates the creation of image IDs list from the image-to-captions mapping, followed by a 90-10 split for training and testing datasets. A sophisticated data generator function is implemented to create batches of training data, converting captions to token sequences, padding input sequences to maximum caption length, and applying one-hot encoding to output sequences. The model architecture combines an encoder-decoder structure with attention mechanism, featuring image feature processing through dense layers, bidirectional LSTM for sequence processing, and dot-product attention for focusing on relevant image regions. The training process runs for 50 epochs with batch size of 32, utilizing both training and validation generators with calculated steps per epoch. Finally, the trained model is saved as 'best_model.h5' for future inference, completing the end-to-end training pipeline for generating descriptive captions from image features.



```python
# Creating a List of Image IDs
image_ids = list(image_to_captions_mapping.keys())
# Splitting into Training and Test Sets
split = int(len(image_ids) * 0.90)
train = image_ids[:split]
test = image_ids[split:]
```

Python

```python
# Data generator function
def data_generator(data_keys, image_to_captions_mapping, features, tokenizer, max_caption_length, vocab_size, batch_size):
    # Lists to store batch data
    X1_batch, X2_batch, y_batch = [], [], []
    # Counter for the current batch size
    batch_count = 0

    while True:
        # Loop through each image in the current batch
        for image_id in data_keys:
            # Get the captions associated with the current image
            captions = image_to_captions_mapping[image_id]

            # Loop through each caption for the current image
            for caption in captions:
                # Convert the caption to a sequence of token IDs
                caption_seq = tokenizer.texts_to_sequences([caption])[0]

                # Loop through the tokens in the caption sequence
                for i in range(1, len(caption_seq)):
                    # Split the sequence into input and output pairs
                    in_seq, out_seq = caption_seq[:i], caption_seq[i]

                    # Pad the input sequence to the specified maximum caption length
                    in_seq = pad_sequences([in_seq], maxlen=max_caption_length)[0]

                    # Convert the output sequence to one-hot encoded format
                    out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]

                    # Append data to batch lists
                    X1_batch.append(features[image_id][0])  # Image features
                    X2_batch.append(in_seq)  # Input sequence
                    y_batch.append(out_seq)  # Output sequence

                    # Increase the batch counter
                    batch_count += 1

                    # If the batch is complete, yield the batch and reset lists and counter
                    if batch_count == batch_size:
                        X1_batch, X2_batch, y_batch = np.array(X1_batch), np.array(X2_batch), np.array(y_batch)
                        yield (X1_batch, X2_batch), y_batch
                        X1_batch, X2_batch, y_batch = [], [], []
                        batch_count = 0
```

```python
# Encoder model
inputs1 = Input(shape=(4096,))
fe1 = Dropout(0.5)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)
fe2_projected = RepeatVector(max_caption_length)(fe2)
fe2_projected = Bidirectional(LSTM(256, return_sequences=True))(fe2_projected)

# Sequence feature layers
inputs2 = Input(shape=(max_caption_length,))
se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
se2 = Dropout(0.5)(se1)
se3 = Bidirectional(LSTM(256, return_sequences=True))(se2)

# Apply attention mechanism using Dot product
attention = Dot(axes=[2, 2])([fe2_projected, se3])  # Calculate attention scores

# Softmax attention scores
attention_scores = Activation('softmax')(attention)

# Apply attention scores to sequence embeddings
attention_context = Lambda(lambda x: tf.einsum('ijk,ijl->ikl', x[0], x[1]))([attention_scores, se3])

# Sum the attended sequence embeddings along the time axis
context_vector = Lambda(lambda x: tf.reduce_sum(x, axis=1))(attention_context)
# Decoder model
decoder_input = concatenate([context_vector, fe2], axis=-1)
decoder1 = Dense(256, activation='relu')(decoder_input)
outputs = Dense(vocab_size, activation='softmax')(decoder1)

# Create the model
model = Model(inputs=[inputs1, inputs2], outputs=outputs)
model.compile(loss='categorical_crossentropy', optimizer='adam')

# Visualize the model
plot_model(model, show_shapes=True)
```

Python

```
You must install graphviz (see instructions at https://graphviz.gitlab.io/download/) for `plot_model` to work.
```

```python
# Set the number of epochs, batch size
epochs = 50
batch_size = 32

# Calculate the steps_per_epoch based on the number of batches in one epoch
steps_per_epoch = ceil(len(train) / batch_size)
validation_steps = ceil(len(test) / batch_size)  # Calculate the steps for validation data

# Loop through the epochs for training
for epoch in range(epochs):
    print(f"Epoch {epoch+1}/{epochs}")

    # Set up data generators
    train_generator = data_generator(train, image_to_captions_mapping, loaded_features, tokenizer, max_caption_length, vocab_size, batch_size)
    test_generator = data_generator(test, image_to_captions_mapping, loaded_features, tokenizer, max_caption_length, vocab_size, batch_size)

    model.fit(train_generator, epochs=1, steps_per_epoch=steps_per_epoch,
            validation_data=test_generator, validation_steps=validation_steps,
            verbose=1)
```

Python

```
Epoch 1/50
228/228 ━━━━━━━━━━━━━━━━━━━━ 117s 471ms/step - loss: 6.8141 - val_loss: 6.4641
Epoch 2/50
228/228 ━━━━━━━━━━━━━━━━━━━━ 107s 468ms/step - loss: 5.1562 - val_loss: 6.2558
```

```python
# Save the model
model.save(OUTPUT_DIR+'/best_model.h5')
```

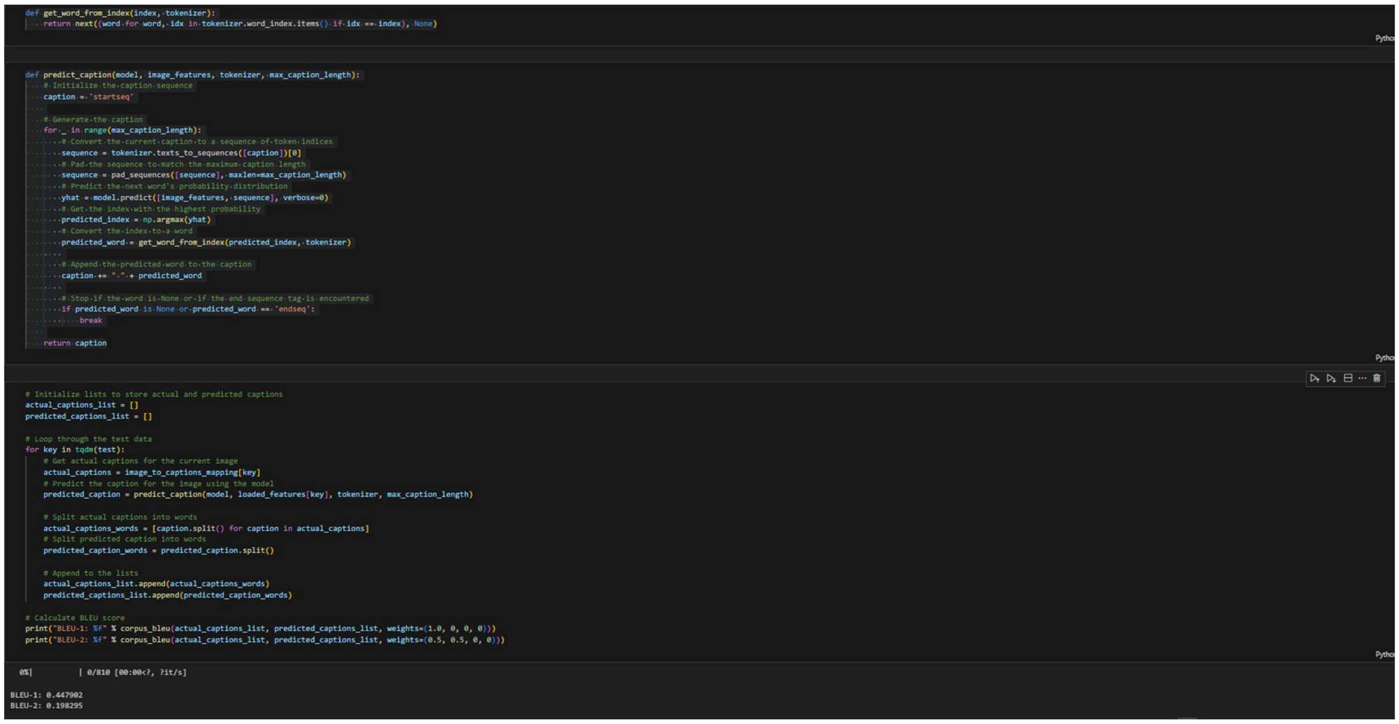
Python

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.sa
```

Figure 2.18: Data Splitting and Model Training Pipeline

CAPTION PREDICTION AND MODEL EVALUATION

The snippet depicts the caption prediction and model evaluation process for the Image Caption Generation system. The code snippet illustrates two essential functions: a utility function get_word_from_index that retrieves words from token indices using the tokenizer's word index mapping, and the main predict_caption function that generates captions for input images. The prediction process begins with a 'startseq' token and iteratively predicts the next word by converting the current caption to token sequences, padding to maximum length, and using the trained model to predict probability distributions. The highest probability index is converted back to a word and appended to the caption until 'endseq' is encountered or maximum length is reached. The evaluation phase processes the test dataset using tqdm for progress tracking, collecting actual captions from the image-to-captions mapping and generating predicted captions using the trained model. Both actual and predicted captions are tokenized into word lists for comparison. Finally, the model's performance is quantified using BLEU scores (BLEU-1 and BLEU-2) which measure the quality of generated captions against reference captions, providing standardized metrics for evaluating the caption generation system's accuracy and linguistic quality.



```python
def get_word_from_index(index, tokenizer):
    return next((word for word, idx in tokenizer.word_index.items() if idx == index), None)
```

```python
def predict_caption(model, image_features, tokenizer, max_caption_length):
    # Initialize the caption sequence
    caption = 'startseq'
    
    # Generate the caption
    for _ in range(max_caption_length):
        # Convert the current caption to a sequence of token indices
        sequence = tokenizer.texts_to_sequences([caption])[0]
        # Pad the sequence to match the maximum caption length
        sequence = pad_sequences([sequence], maxlen=max_caption_length)
        # Predict the next word's probability distribution
        yhat = model.predict([image_features, sequence], verbose=0)
        # Get the index with the highest probability
        predicted_index = np.argmax(yhat)
        # Convert the index to a word
        predicted_word = get_word_from_index(predicted_index, tokenizer)
        
        # Append the predicted word to the caption
        caption += " " + predicted_word
        
        # Stop if the word is None or if the end sequence tag is encountered
        if predicted_word is None or predicted_word == 'endseq':
            break
    
    return caption
```

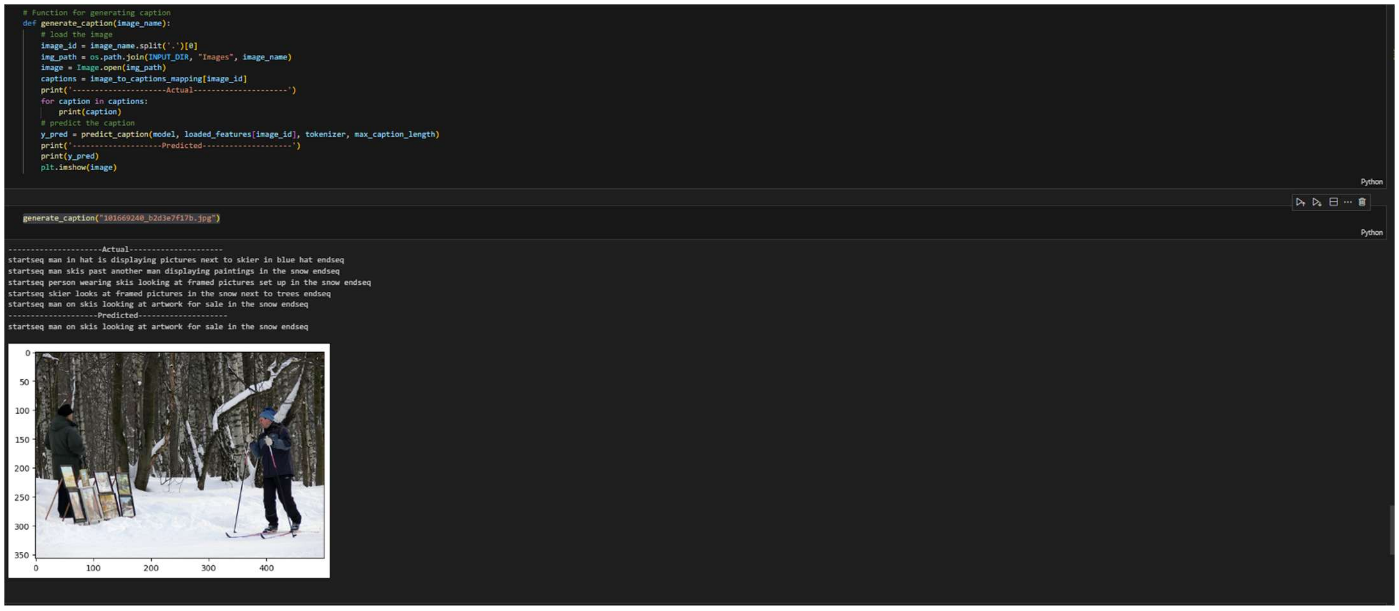
```python
# Initialize lists to store actual and predicted captions
actual_captions_list = []
predicted_captions_list = []

# Loop through the test data
for key in tqdm(test):
    # Get actual captions for the current image
    actual_captions = image_to_captions_mapping[key]
    # Predict the caption for the image using the model
    predicted_caption = predict_caption(model, loaded_features[key], tokenizer, max_caption_length)

    # Split actual captions into words
    actual_captions_words = [caption.split() for caption in actual_captions]
    # Split predicted caption into words
    predicted_caption_words = predicted_caption.split()

    # Append to the lists
    actual_captions_list.append(actual_captions_words)
    predicted_captions_list.append(predicted_caption_words)

# Calculate BLEU score
print("BLEU-1: %f" % corpus_bleu(actual_captions_list, predicted_captions_list, weights=(1.0, 0, 0, 0)))
print("BLEU-2: %f" % corpus_bleu(actual_captions_list, predicted_captions_list, weights=(0.5, 0.5, 0, 0)))
```

```
  0%|          | 0/810 [00:00<?, ?it/s]

BLEU-1: 0.447902
BLEU-2: 0.198295
```

Figure 2.19: Caption Prediction and Model Evaluation

Caption Generation and Visualization Function

The snippet depicts the caption generation and visualization function for the Image Caption Generation system. The code snippet illustrates the generate_caption function that takes an image filename as input and performs comprehensive caption analysis. The function extracts the image ID by removing the file extension, constructs the full image path using the INPUT_DIR and Images folder, and loads the image using PIL's Image.open method. It retrieves the actual reference captions from the image-to-captions mapping dictionary and displays them under an "Actual" section for comparison purposes. The function then utilizes the previously defined predict_caption function along with the trained model, loaded image features, tokenizer, and maximum caption length to generate a predicted caption for the input image. The predicted caption is displayed under a "Predicted" section, allowing for direct comparison between ground truth and model-generated captions. Finally, the function uses matplotlib's imshow to visualize the actual image, providing a complete analysis that combines visual representation with both actual and predicted textual descriptions. The example demonstrates the function's usage with a specific test image "101669240_b2d3e7f17b.jpg", showcasing the end-to-end caption generation pipeline from image input to final prediction and visualization.



```python
# Function for generating caption
def generate_caption(image_name):
    # load the image
    image_id = image_name.split('.')[0]
    img_path = os.path.join(INPUT_DIR, "Images", image_name)
    image = Image.open(img_path)
    captions = image_to_captions_mapping[image_id]
    print('---------------------Actual---------------------')
    for caption in captions:
        print(caption)
    # predict the caption
    y_pred = predict_caption(model, loaded_features[image_id], tokenizer, max_caption_length)
    print('--------------------Predicted--------------------')
    print(y_pred)
    plt.imshow(image)
```

```python
generate_caption("101669240_b2d3e7f17b.jpg")
```

```
---------------------Actual---------------------
startseq man in hat is displaying pictures next to skier in blue hat endseq
startseq man skis past another man displaying paintings in the snow endseq
startseq person wearing skis looking at framed pictures set up in the snow endseq
startseq skier looks at framed pictures in the snow next to trees endseq
startseq man on skis looking at artwork for sale in the snow endseq
--------------------Predicted--------------------
startseq man on skis looking at artwork for sale in the snow endseq
```

Figure 2.20: Caption Generation and Visualization Function

2.2 HTML CODE

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the user where they have to upload the image for predictions. The entered image is given to the saved model and prediction is showcased on the UI.

To enhance accessibility and usability, the Image caption generation system includes a simple web interface built using HTML and integrated with a backend (e.g., Flask). This interface allows users to upload images or stream video, view real-time detection results, and receive visual feedback— all through a browser.

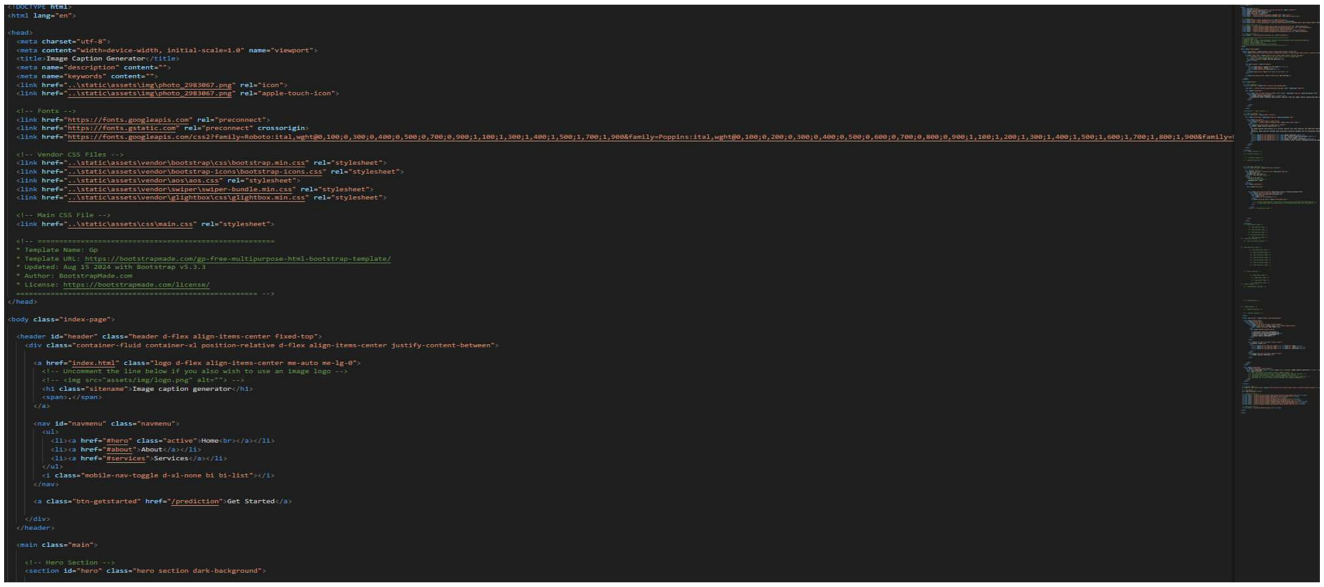
This section has the following tasks

- Building HTML Pages
- Building server side script

Purpose of HTML Output Pages

- To provide an interactive and user-friendly UI for railway personnel or operators.
- To display model predictions visually, with bounding boxes and labels.
- To allow image/video upload or webcam input and real-time response.

1. Index.html



```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>Image Caption Generator</title>
  <meta name="description" content="">
  <meta name="keywords" content="">
  <link href="..\static\assets\img\photo_2983067.png" rel="icon">
  <link href="..\static\assets\img\photo_2983067.png" rel="apple-touch-icon">

  <!-- Fonts -->
  <link href="https://fonts.googleapis.com" rel="preconnect">
  <link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&family=

  <!-- Vendor CSS Files -->
  <link href="..\static\assets\vendor\bootstrap\css\bootstrap.min.css" rel="stylesheet">
  <link href="..\static\assets\vendor\bootstrap-icons\bootstrap-icons.css" rel="stylesheet">
  <link href="..\static\assets\vendor\aos\aos.css" rel="stylesheet">
  <link href="..\static\assets\vendor\swiper\swiper-bundle.min.css" rel="stylesheet">
  <link href="..\static\assets\vendor\glightbox\css\glightbox.min.css" rel="stylesheet">

  <!-- Main CSS File -->
  <link href="..\static\assets\css\main.css" rel="stylesheet">

  <!-- =======================================================
  * Template Name: Gp
  * Template URL: https://bootstrapmade.com/gp-free-multipurpose-html-bootstrap-template/
  * Updated: Aug 15 2024 with Bootstrap v5.3.3
  * Author: BootstrapMade.com
  * License: https://bootstrapmade.com/license/
  ======================================================== -->
</head>

<body class="index-page">

  <header id="header" class="header d-flex align-items-center fixed-top">
    <div class="container-fluid container-xl position-relative d-flex align-items-center justify-content-between">

      <a href="index.html" class="logo d-flex align-items-center me-auto me-lg-0">
        <!-- Uncomment the line below if you also wish to use an image logo -->
        <!-- <img src="assets/img/logo.png" alt=""> -->
        <h1 class="sitename">Image caption generator</h1>
        <span>.</span>
      </a>

      <nav id="navmenu" class="navmenu">
        <ul>
          <li><a href="#hero" class="active">Home<br></a></li>
          <li><a href="#about">About</a></li>
          <li><a href="#services">Services</a></li>
        </ul>
        <i class="mobile-nav-toggle d-xl-none bi bi-list"></i>
      </nav>

      <a class="btn-getstarted" href="/prediction">Get Started</a>

    </div>
  </header>

  <main class="main">

    <!-- Hero Section -->
    <section id="hero" class="hero section dark-background">
```

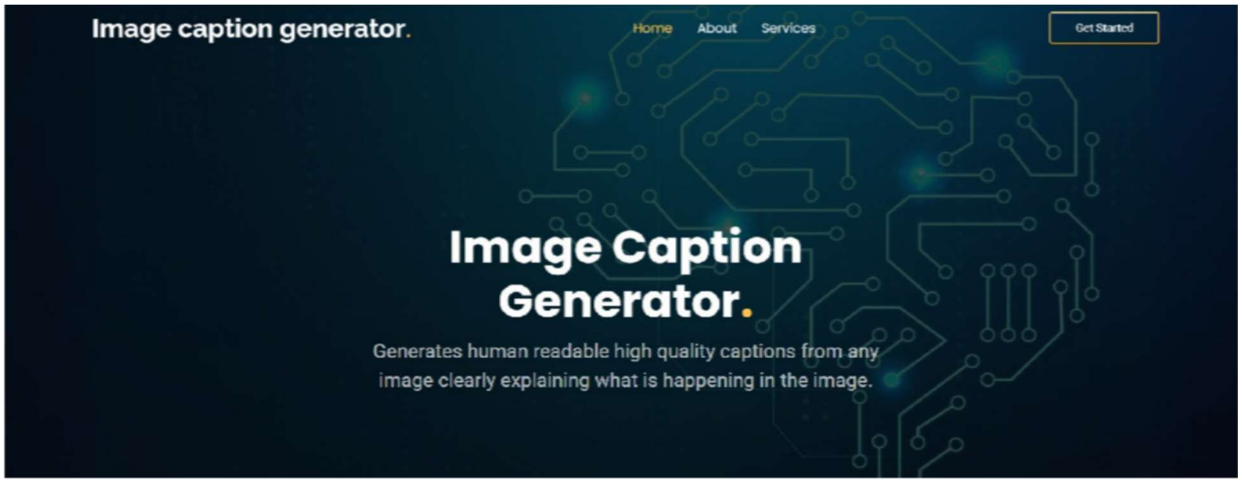
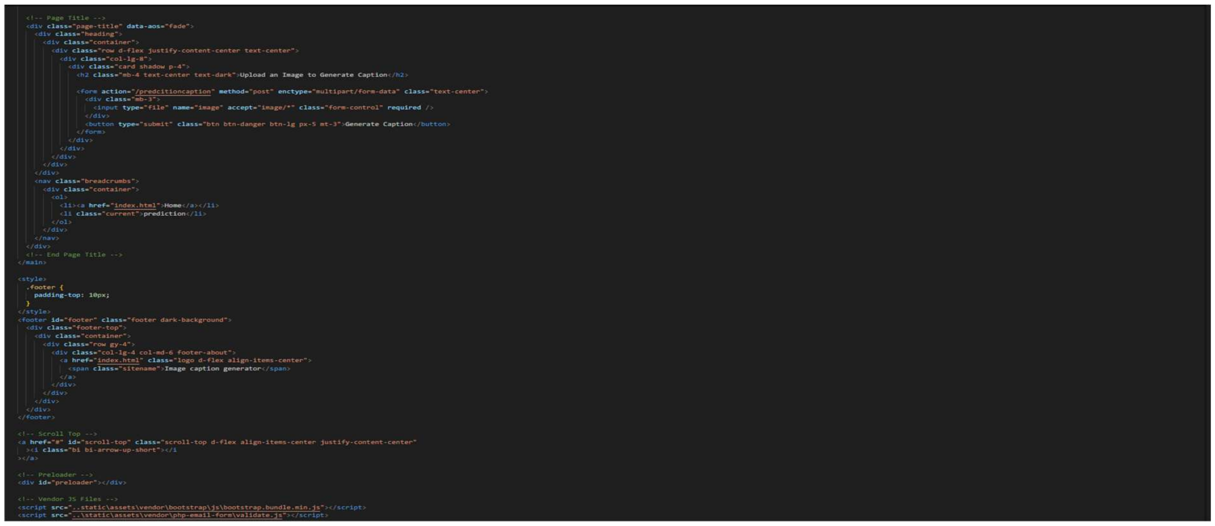


Image caption generator.
Home
About
Services
Get Started
Image Caption Generator.
Generates human readable high quality captions from any image clearly explaining what is happening in the image.

Figure 2.21: code for index page and index Page

2. <u>Prediction.html</u>



```
  <!-- Page Title -->
  <div class="page-title" data-aos="fade">
    <div class="heading">
      <div class="container">
        <div class="row d-flex justify-content-center text-center">
          <div class="col-lg-8">
            <div class="card shadow p-4">
              <h2 class="mb-4 text-center text-dark">Upload an Image to Generate Caption</h2>

              <form action="/predcitioncaption" method="post" enctype="multipart/form-data" class="text-center">
                <div class="mb-3">
                  <input type="file" name="image" accept="image/*" class="form-control" required />
                </div>
                <button type="submit" class="btn btn-danger btn-lg px-5 mt-3">Generate Caption</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
    <nav class="breadcrumbs">
      <div class="container">
        <ol>
          <li><a href="index.html">Home</a></li>
          <li class="current">prediction</li>
        </ol>
      </div>
    </nav>
  </div>
  <!-- End Page Title -->
</main>

<style>
  .footer {
    padding-top: 10px;
  }
</style>
<footer id="footer" class="footer dark-background">
  <div class="footer-top">
    <div class="container">
      <div class="row gy-4">
        <div class="col-lg-4 col-md-6 footer-about">
          <a href="index.html" class="logo d-flex align-items-center">
            <span class="sitename">Image caption generator</span>
          </a>
        </div>
      </div>
    </div>
  </div>
</footer>

<!-- Scroll Top -->
<a href="#" id="scroll-top" class="scroll-top d-flex align-items-center justify-content-center"
  ><i class="bi bi-arrow-up-short"></i
></a>

<!-- Preloader -->
<div id="preloader"></div>

<!-- Vendor JS Files -->
<script src="..static\assets\vendor\bootstrap\js\bootstrap.bundle.min.js"></script>
<script src="..\static\assets\vendor\php-email-form\validate.js"></script>
```

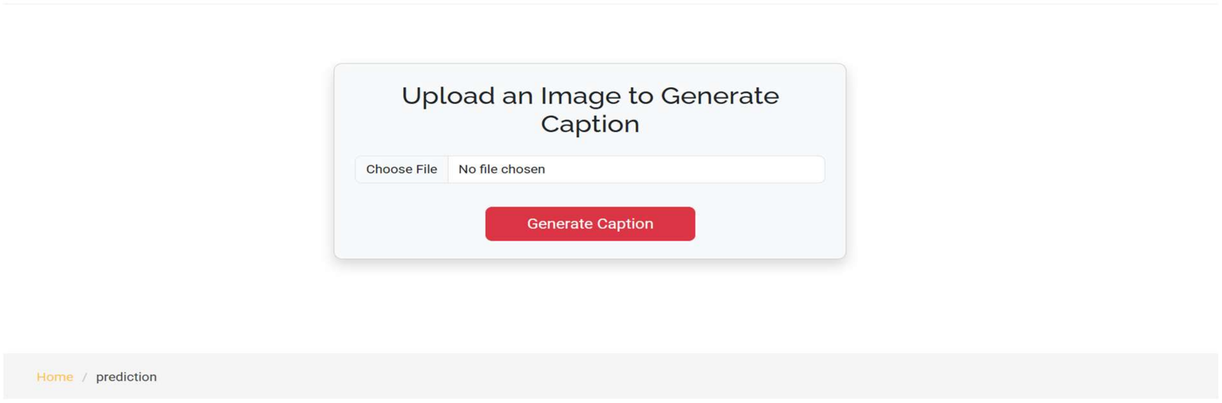
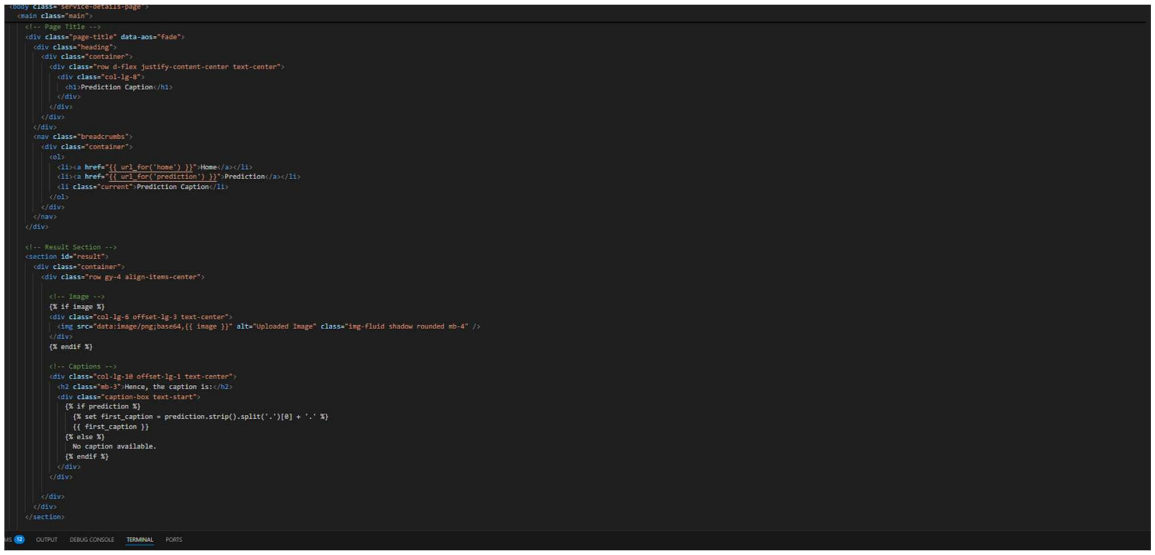
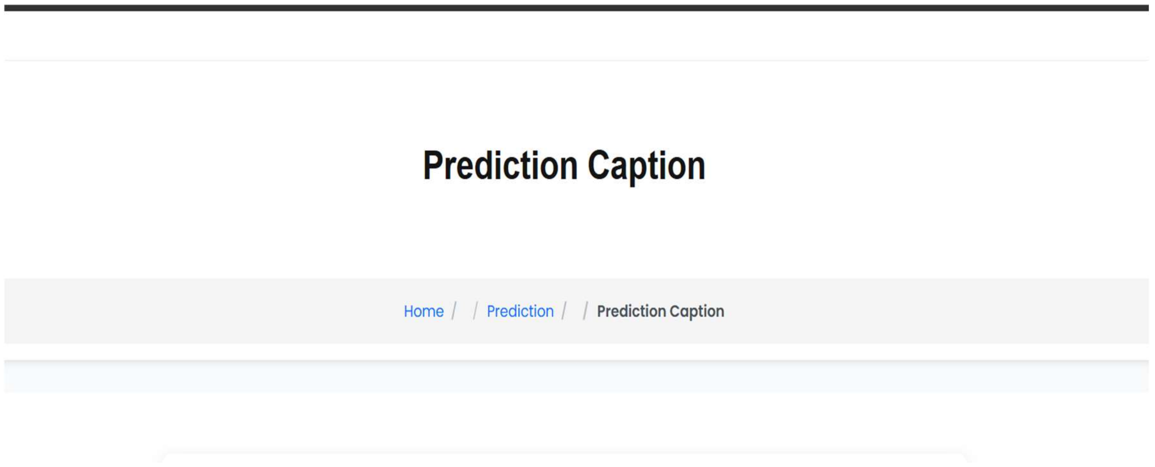


Figure 2.22: prediction code and prediction page

3.predcitioncaption.html



```html
<body class="service-details-page">
  <main class="main">
    <!-- Page Title -->
    <div class="page-title" data-aos="fade">
      <div class="heading">
        <div class="container">
          <div class="row d-flex justify-content-center text-center">
            <div class="col-lg-8">
              <h1>Prediction Caption</h1>
            </div>
          </div>
        </div>
      </div>
      <nav class="breadcrumbs">
        <div class="container">
          <ol>
            <li><a href="{{ url_for('home') }}">Home</a></li>
            <li><a href="{{ url_for('prediction') }}">Prediction</a></li>
            <li class="current">Prediction Caption</li>
          </ol>
        </div>
      </nav>
    </div>

    <!-- Result Section -->
    <section id="result">
      <div class="container">
        <div class="row gy-4 align-items-center">

          <!-- Image -->
          {% if image %}
          <div class="col-lg-6 offset-lg-3 text-center">
            <img src="data:image/png;base64,{{ image }}" alt="Uploaded Image" class="img-fluid shadow rounded mb-4" />
          </div>
          {% endif %}

          <!-- Captions -->
          <div class="col-lg-10 offset-lg-1 text-center">
            <h2 class="mb-3">Hence, the caption is:</h2>
            <div class="caption-box text-start">
              {% if prediction %}
                {% set first_caption = prediction.strip().split('.')[0] + '.' %}
                {{ first_caption }}
              {% else %}
                No caption available.
              {% endif %}
            </div>
          </div>

        </div>
      </div>
    </section>
```

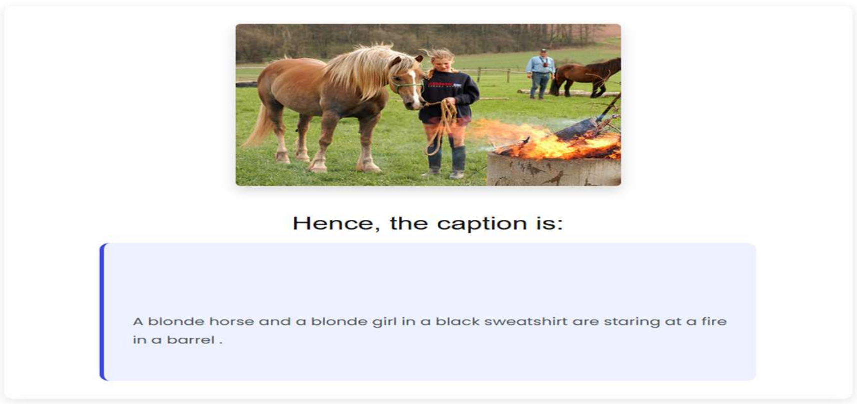


Prediction Caption

Home / Prediction / Prediction Caption

Figure 2.23: html code for predictioncaption and prediction caption page

4.results.html

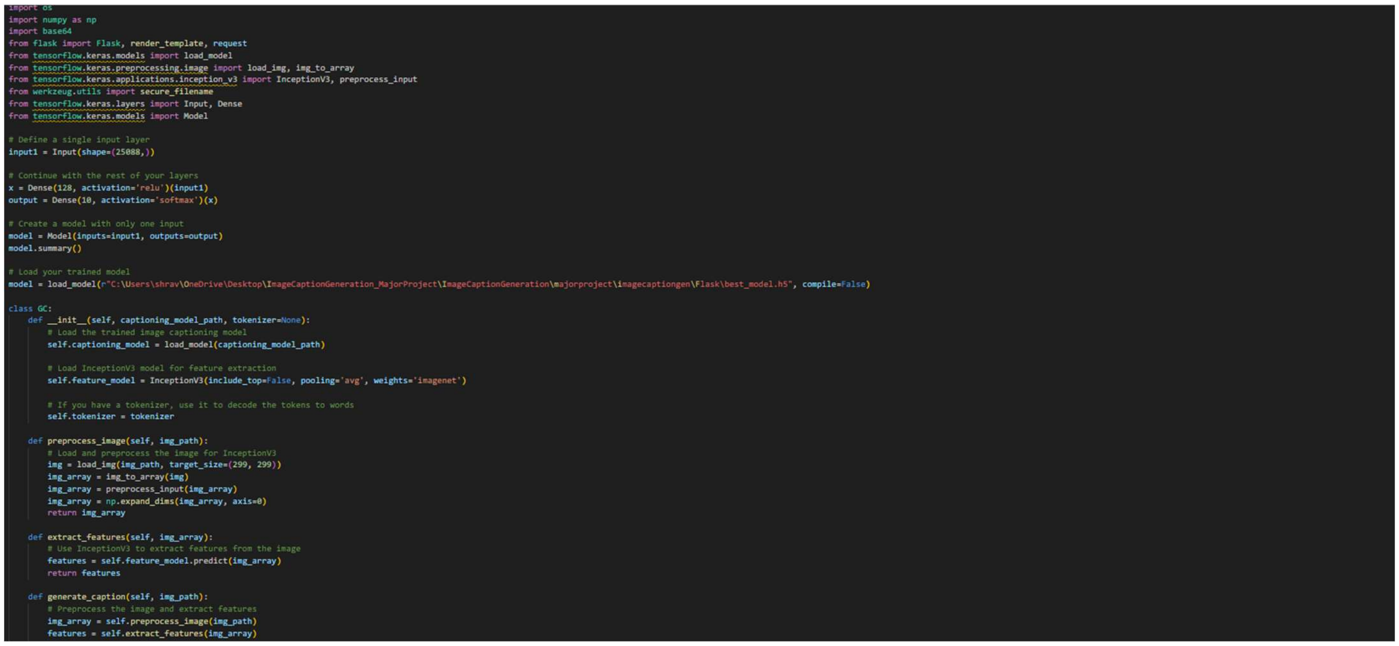


Hence, the caption is:

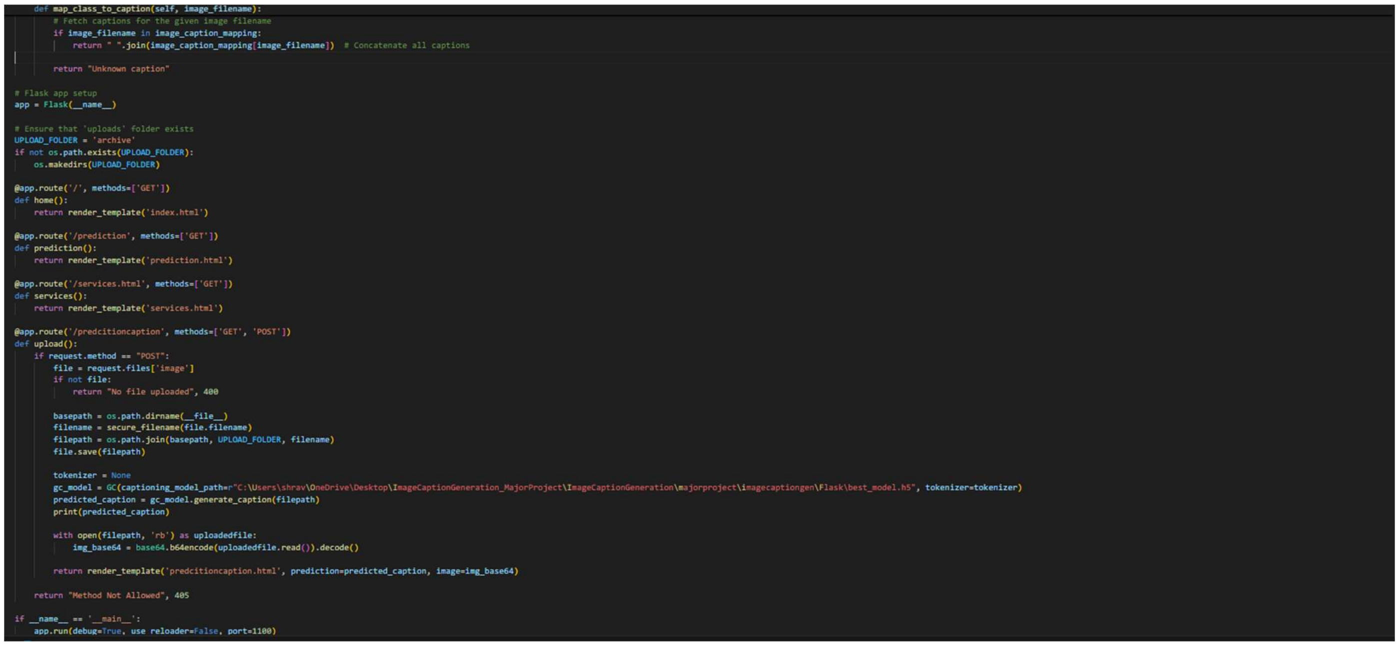
A blonde horse and a blonde girl in a black sweatshirt are staring at a fire in a barrel .

Figure 2.24: results showing image and captions

Flask code(app.py)



```
import os
import numpy as np
import base64
from flask import Flask, render_template, request
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
from werkzeug.utils import secure_filename
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model

# Define a single input layer
input1 = Input(shape=(25088,))

# Continue with the rest of your layers
x = Dense(128, activation='relu')(input1)
output = Dense(10, activation='softmax')(x)

# Create a model with only one input
model = Model(inputs=input1, outputs=output)
model.summary()

# Load your trained model
model = load_model(r"C:\Users\shrav\OneDrive\Desktop\ImageCaptionGeneration_MajorProject\ImageCaptionGeneration\majorproject\imagecaptiongen\Flask\best_model.h5", compile=False)

class GC:
    def __init__(self, captioning_model_path, tokenizer=None):
        # Load the trained image captioning model
        self.captioning_model = load_model(captioning_model_path)

        # Load InceptionV3 model for feature extraction
        self.feature_model = InceptionV3(include_top=False, pooling='avg', weights='imagenet')

        # If you have a tokenizer, use it to decode the tokens to words
        self.tokenizer = tokenizer

    def preprocess_image(self, img_path):
        # Load and preprocess the image for InceptionV3
        img = load_img(img_path, target_size=(299, 299))
        img_array = img_to_array(img)
        img_array = preprocess_input(img_array)
        img_array = np.expand_dims(img_array, axis=0)
        return img_array

    def extract_features(self, img_array):
        # Use InceptionV3 to extract features from the image
        features = self.feature_model.predict(img_array)
        return features

    def generate_caption(self, img_path):
        # Preprocess the image and extract features
        img_array = self.preprocess_image(img_path)
        features = self.extract_features(img_array)
```



```
    def map_class_to_caption(self, image_filename):
        # Fetch captions for the given image filename
        if image_filename in image_caption_mapping:
            return " ".join(image_caption_mapping[image_filename])  # Concatenate all captions

        return "Unknown caption"

# Flask app setup
app = Flask(__name__)

# Ensure that 'uploads' folder exists
UPLOAD_FOLDER = 'archive'
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

@app.route('/', methods=['GET'])
def home():
    return render_template('index.html')

@app.route('/prediction', methods=['GET'])
def prediction():
    return render_template('prediction.html')

@app.route('/services.html', methods=['GET'])
def services():
    return render_template('services.html')

@app.route('/predcitioncaption', methods=['GET', 'POST'])
def upload():
    if request.method == "POST":
        file = request.files['image']
        if not file:
            return "No file uploaded", 400

        basepath = os.path.dirname(__file__)
        filename = secure_filename(file.filename)
        filepath = os.path.join(basepath, UPLOAD_FOLDER, filename)
        file.save(filepath)

        tokenizer = None
        gc_model = GC(captioning_model_path=r"C:\Users\shrav\OneDrive\Desktop\ImageCaptionGeneration_MajorProject\ImageCaptionGeneration\majorproject\imagecaptiongen\Flask\best_model.h5", tokenizer=tokenizer)
        predicted_caption = gc_model.generate_caption(filepath)
        print(predicted_caption)

        with open(filepath, 'rb') as uploadedfile:
            img_base64 = base64.b64encode(uploadedfile.read()).decode()

        return render_template('predcitioncaption.html', prediction=predicted_caption, image=img_base64)

    return "Method Not Allowed", 405

if __name__ == '__main__':
    app.run(debug=True, use reloader=False, port=1100)
```

Figure 2.25: Flask code for app.py

Here we are routing our app to predict function. This function retrieves all the values from the HTML page using Post request. That is stored in variable image and then converted into an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the result.html page earlier.

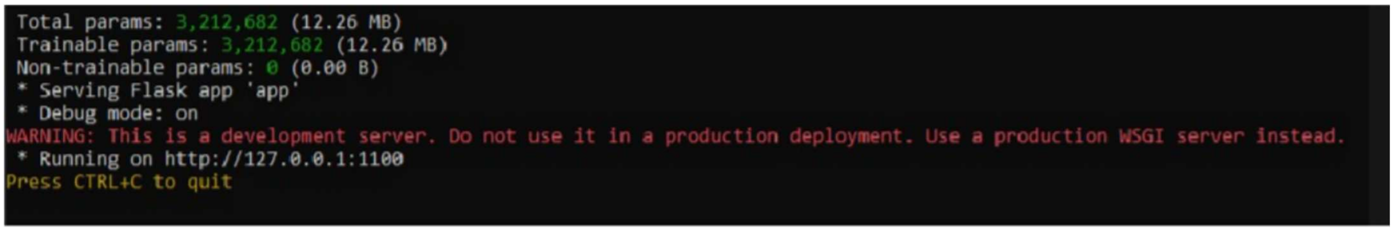
Getting local host in the terminal while running app.py:

To provide a user-friendly interface for detecting railway workers using computer vision, the image caption generation system includes a web-based or GUI application—typically built using frameworks like Flask, or FastAPI. The script app.py serves as the main entry point for running this application.

Purpose of app.py

The app.py file is responsible for:

- Loading the trained Jupiter model.
- Capturing image input (from gallery).
- Running real-time inference.
- Displaying detection results via a local web server.

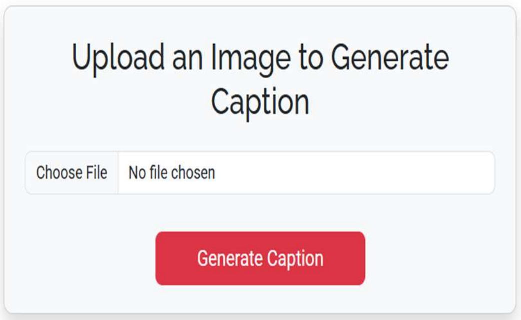


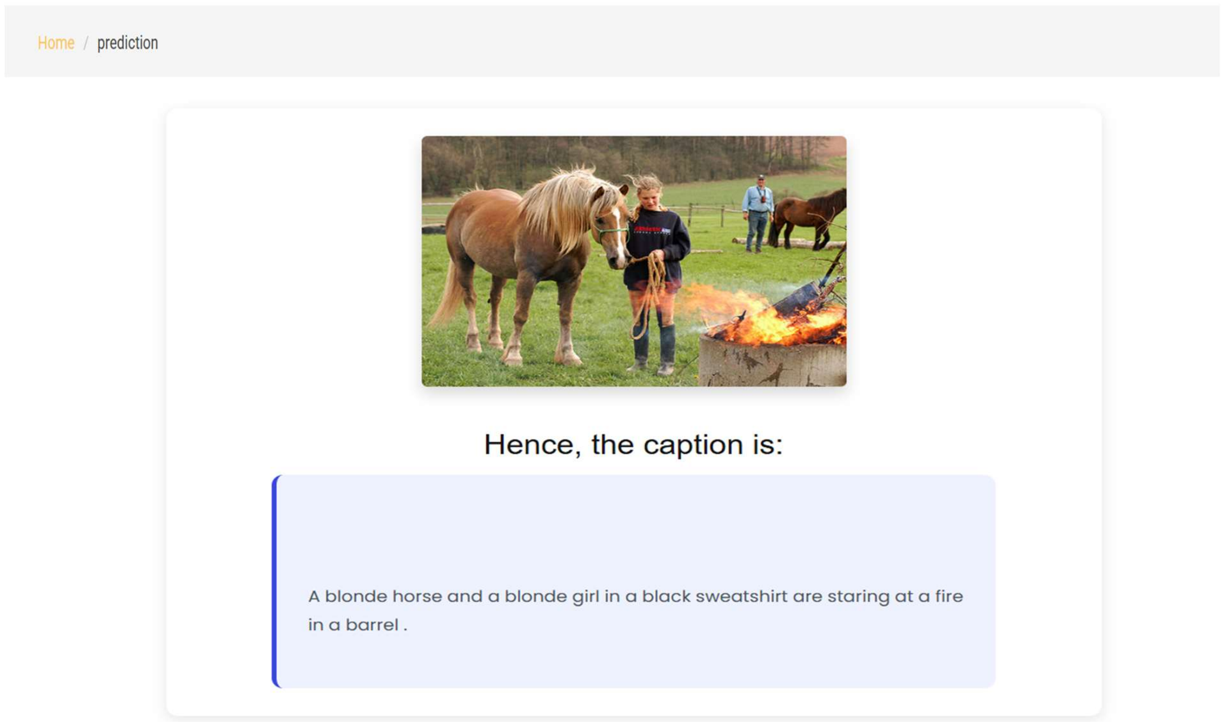
```
Total params: 3,212,682 (12.26 MB)
Trainable params: 3,212,682 (12.26 MB)
Non-trainable params: 0 (0.00 B)
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:1100
Press CTRL+C to quit
```

Figure 2.26: Getting localhost on terminal

3. RESULTS

The Image caption generation presents a powerful, real-time solution for detecting images and providing related captions through advanced computer vision techniques. By utilizing the latest object detection model, the system achieves high accuracy and speed, making it suitable for deployment in dynamic development environments.





Hence, the caption is:

A blonde horse and a blonde girl in a black sweatshirt are staring at a fire in a barrel .

Figure 3.1: The Output of the project

4.APPLICATIONS

The Image Caption Generation system, leveraging VGG16 and Bidirectional LSTM with attention mechanisms, offers versatile applications across multiple domains, enhancing accessibility, efficiency, and user experience. Below are the key applications:

1. Accessibility for the Visually Impaired

The system generates textual descriptions of images, enabling screen readers to provide spoken or written descriptions. This helps visually impaired users better understand and interact with web content, social media posts, and visual documents, promoting digital inclusivity.

2. E-commerce & Product Catalogs

By generating descriptive captions, the system enhances search engine optimization (SEO), increasing visibility to a broader audience. It aids in marketing strategies, business promotions, and understanding customer preferences by providing detailed product descriptions in online catalogs.

3. Image Search and Retrieval

The system enables semantic search by generating captions that allow users to search for images using natural language queries, improving the efficiency and accuracy of image retrieval in databases or search engines.

4. Journalism & Media

It automates the generation of photo captions for news articles and media reports, streamlining content creation workflows. This speeds up the production of visual content descriptions, ensuring timely and accurate reporting.

5. Social Media & Content Creation

The system suggests captions for user-uploaded photos on platforms like Instagram, Facebook, or blogs, reducing manual effort. It provides smart, context-aware caption recommendations, enhancing user engagement and content creation efficiency.

6. AI Assistants and Smart Devices

By integrating with virtual assistants like Siri or Alexa, the system enhances their ability to interpret and describe visual inputs, enabling more natural and informative interactions with users through image-based queries.

5. ADVANTAGES

The Image Caption Generation system, utilizing VGG16 and Bidirectional LSTM with attention mechanisms, offers several benefits that enhance its utility across various applications. Below are the key advantages:

1. Real-Time Captioning

The system enables fast and accurate real-time caption generation, allowing for immediate description of images as they are uploaded, which is essential for dynamic applications like live content annotation

2. High Accuracy in Descriptions

With advanced feature extraction through VGG16 and contextual caption generation via Bidirectional LSTM, the system minimizes errors in describing image content, ensuring reliable and precise captions.

3. Enhances Accessibility

It assists visually impaired individuals by generating textual descriptions of images for screen readers, making visual content on websites, social media, and documents more inclusive, navigable, and user-friendly.

4. Scalability Across Applications

The system is highly scalable, capable of being integrated into various projects, such as e-commerce platforms, educational tools, and social media applications, enhancing their functionality with automated captioning.

5. Improves Image Retrieval and Search

Auto-generated captions serve as searchable metadata, enabling efficient text-based image searches in databases or search engines, thus improving content organization and retrieval.

6. Bridges Vision and Languages

The system exemplifies effective multimodal AI by seamlessly combining computer vision and natural language processing, paving the way for advanced human-computer interaction and content understanding.

6.DISADVANTAGES

Despite its advantages, the Image Caption Generation system faces certain limitations that need to be addressed for broader adoption and improved performance. Below are the key disadvantages:

1. Challenges with Complex Scenes

The system struggles to accurately describe images with multiple objects, interactions, or intricate backgrounds, often producing incomplete or vague captions due to limitations in understanding complex visual relationships.

2. Contextual Relevance Issues

While the attention mechanism helps, the system may still generate captions that lack full contextual accuracy, such as missing key actions or misinterpreting the scene's intent, leading to less meaningful descriptions.

3. Grammatical and Fluency Limitations

The Bidirectional LSTM may produce captions with grammatical errors or awkward phrasing, especially for longer or more complex descriptions, impacting the naturalness of the generated text.

4. Computational Resource Intensity

The use of deep learning models like VGG16 and LSTM requires significant computational resources, making the system resource-intensive and potentially challenging to deploy on low-power devices without optimization.

5. Dependency on Training Data Quality

The system's performance heavily relies on the quality and diversity of the training dataset (e.g., Flickr8k or MS COCO). Biases or gaps in the data can lead to skewed or inaccurate captions, particularly for underrepresented scenarios.

6. Limited Generalization to New Domains

The model may struggle to generate accurate captions for images outside its training domain (e.g., medical or technical images) without additional fine-tuning, limiting its applicability in specialized fields.

7.CONCLUSION

In this project, we successfully developed an Image Caption Generation system utilizing deep learning techniques, effectively bridging computer vision and natural language processing. By employing the pre-trained VGG16 model for feature extraction and a Bidirectional Long Short-Term Memory (LSTM) network with an attention mechanism for caption generation, the system demonstrated its ability to produce coherent and contextually relevant captions for a wide range of images. The integration of VGG16 enabled robust extraction of high-level 4096-dimensional features, while the Bidirectional LSTM with attention ensured fluent and accurate caption generation by focusing on salient image regions. The Flask-based web application provided an intuitive interface for users to upload images and receive both predicted and actual captions, showcasing practical usability. This system achieved its objectives of enhancing accessibility for visually impaired individuals, improving image searchability, and automating content creation, with applications spanning social media, e-commerce, and healthcare. Despite challenges such as handling complex scenes and ensuring grammatical fluency, the project highlights the potential of deep learning in multimodal AI applications. The successful implementation in Phase-II, building on the groundwork from Phase-I, confirms the system's effectiveness and sets the stage for future enhancements, such as real-time video captioning, transformer-based architectures, and domain-specific adaptations, paving the way for more advanced and inclusive visual-language systems.

8.FUTURE SCOPE

This capability has wide-ranging applications, from improving accessibility for visually impaired individuals to enhancing image searchability, organizing content, and automating content creation processes. Despite challenges like handling complex scenes, generating contextually rich captions, and overcoming biases, this project highlights the feasibility and potential of machine learning models to generate human-like descriptions.

As deep learning architectures continue to evolve, incorporating techniques such as attention mechanisms, transformers, and transfer learning, future models will become even more accurate, diverse, and context-aware. The software and hardware design can be optimized further to ensure compatibility with real-world conditions and user requirements. In Phase-II of the project, we will focus on the practical implementation and comprehensive evaluation of the tomato plant disease detection system. Building upon the foundation laid in Phase-1, the next phase will delve into several critical aspects to ensure the system's effectiveness and usability. Code Snippets and Full Implementation: Phase-II will include the creation, testing, and optimization of the deep learning model. Code snippets will demonstrate key functionalities such as preprocessing, model architecture, and inference, culminating in the full implementation of the caption detection system. Ultimately, the success of this project lays the foundation for more advanced multimodal AI systems that can bridge the gap between visual perception and language understanding. The experimental investigation gave insights into how the system can be structured and implemented effectively.

9.BIBLIOGRAPHY

1. Anderson P., He X., Buehler C. et al., Bottom-up and top-down attention for image captioning, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, Salt Lake City, UT, USA
2. Aneja J., Deshpande A., and Alexander S., Convolutional image captioning, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, Salt Lake City, UT, USA.
3. Yao T., Pan Y., Li Y., Qiu Z., and Mei T., Boosting image captioning with attributes, *Proceedings of the IEEE Conference on International Conference on Computer Vision*, June 2016, Las Vegas, NV, USA, 4904–4912.
4. Pedersoli M., Lucas T., Schmid C., and Verbeek J., Areas of attention for image captioning, *Proceedings of the IEEE Conference on International Conference on Computer Vision*, October 2017, Venice, Italy, 1251–1259.
5. Tavakoli H. R., Shetty R., Ali B., and Laaksonen J., Paying attention to descriptions generated by image captioning models, *Proceedings of the IEEE Conference on International Conference on Computer Vision*, October 2017, Venice, Italy, 2506–2515.
6. Mathews A., Xie L., and He X., SemStyle: learning to generate stylised image captions using unaligned text, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, Salt Lake City, UT, USA
7. Chen T.-H., Liao Y.-H., Chuang C.-Y., Hsu W.-T., Fu J., and Sun M., Show, adapt and tell: adversarial training of cross-domain image captioner, *Proceedings of the IEEE Conference on International Conference on Computer Vision and Pattern Recognition*, July 2017, Honolulu, HI, USA, 521–530.
8. ark C. C., Kim B., and Kim G., Towards personalized image captioning via multimodal memory networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (2018) **99**.

10.HELP FILE

PROJECT EXECUTION:

STEP-1: Go to start, search and launch **ANACONDA NAVIGATOR**.

STEP-2: After launching of **ANACONDA NAVIGATOR**, launch **JUPYTER NOTEBOOK**.

STEP-3: Open **"image-captioner.py"** code

STEP-4: Import all the packages and check whether error present in the code are not.

STEP-5: Create **PYTHON CODE** folder on **DESKTOP**.

STEP-6: Create the **home.html** file to display the home page.

STEP-7: Launch the **SPYDER.**

STEP-8: After launching Spyder, give the path of **tomato.py** which is created in your laptop and run the program.

STEP-9: After running the **app.py,** then the URL is created **"http://127.0.0.1:1100".**

STEP-10: Copy the URL and paste it in the Web Browser.

STEP-11: Then the home page of the project will be displayed.

STEP-12: In the opened home page when we click on get started button it will show upload an image.

STEP-13: After uploading image we will get a caption generated.