9920103100_Image Captioning Bot

by Samarth Gupta

Submission date: 22-Apr-2024 09:01PM (UTC+0530)
Submission ID: 2358154851
File name: 9920103100_Image_Captioning_Bot.pdf (3.46M)
Word count: 11066
Character count: 65465

IMAGE CAPTIONING BOT

Enrollment Numbers – 9920103096, 9920103097, 9920103100

Name of Students – Chandrika Rajvanshi, Abhay Arya, Samarth Gupta

Name of supervisor – Dr. Himani Bansal

May 2024

Submitted in partial fulfillment of the Degree of

Bachelor of Technology

in

Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING AND INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

TABLE OF CONTENTS

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Jaypee Institute of Information Technology, Noida
Date:

Signature:
Name: Chandrika Rajvanshi
Enrollment No: 9920103096

Signature:
Name: Abhay Arya
Enrollment No: 9920103097

Signature:
Name: Samarth Gupta
Enrollment No: 9920103100

CERTIFICATE

This is to certify that the work titled "**Image Captioning Bot**" submitted by **Chandrika Rajvanshi (9920103096), Abhay Arya (9920103097), Samarth Gupta (9920103100)** in partial fulfilment for the award of degree of B. Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:

Name of Supervisor: Dr. Himani Bansal

Designation: Assistant Professor (Senior Grade)

Date:

ACKNOWLEDGEMENT

We would like to place on record our deep sense of gratitude to Dr. Himani Bansal, Assistant Professor (Senior Grade) in the department of Computer Science at Jaypee Institute of Information Technology, Noida for her generous guidance, help and useful suggestions.

We also wish to extend our thanks to our friends and other classmates for their insightful comments and constructive suggestions to improve the quality of this project work.

Signature:
Name: Chandrika Rajvanshi
Enrollment No: 9920103096
Date:

Signature:
Name: Abhay Arya
Enrollment No: 9920103097
Date:

Signature:
Name: Samarth Gupta
Enrollment No: 9920103100
Date:

SUMMARY

The project endeavors to develop an innovative image captioning bot, leveraging deep learning techniques within the realm of computer vision. By harnessing the Flickr 8k dataset, comprising 8,000 images with five captions each, the model is trained to generate descriptive and contextually relevant captions. Data preprocessing involves cleaning captions, constructing a vocabulary, and encoding images using pre-trained convolutional neural networks, including ResNet50, VGG-16, and InceptionV3. The model architecture incorporates a blend of recurrent neural networks (RNNs) and convolutional networks to predict captions, prioritizing relevance, and accuracy. Implementation extends to a user-friendly website, where users can upload images and receive captions, alongside relevant images sourced from Google. The system's backend utilizes PostgreSQL for database management, facilitating user login, storage of uploaded images, and caption history. Despite demonstrating promising results, future enhancements are envisioned, encompassing larger datasets, hyperparameter fine-tuning, and exploration of advanced techniques like attention mechanisms. This project serves as an initial exploration into the domain of image captioning, highlighting its diverse applications and setting the stage for further advancements.

_____

Signature of Student

Name: Chandrika Rajvanshi

Date:

_____

Signature of Supervisor

Name: Dr. Himani Bansal

Date:

_____

Signature of Student

Name: Abhay Arya

Date:

_____

Signature of Student

Name: Samarth Gupta

Date:

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS & ACRONYMS

SYMBOL	FULL FORM
ML	Machine Learning
DL	Deep Learning
NN	Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory

Chapter 1: INTRODUCTION

1.1 GENERAL INTRODUCTION

In recent years, the fusion of deep learning techniques with computer vision has spurred remarkable advancements in various fields. Among these advancements, the development of an image captioning bot stands out as a cutting-edge application, harnessing the power of artificial intelligence to generate descriptive and contextually relevant captions for images. This project embarks on exploring the potential of image captioning technology, driven by its promising real-world applications across diverse domains.

The motivation behind this endeavor stems from the profound impact that image captioning can have on a multitude of sectors. From aiding autonomous vehicles in understanding their surroundings to assisting visually impaired individuals in interpreting visual information, the implications of this technology are vast. The potential to enhance surveillance through CCTV cameras and improve image search capabilities adds layers of practicality to the exploration of image captioning. Furthermore, the integration of image captioning into social media platforms could revolutionize content accessibility.

To realize the goals of this project, we leverage the Flickr 8k dataset, a rich repository of images accompanied by descriptive captions. Through meticulous data preprocessing, including caption cleaning and vocabulary creation, we lay the groundwork for training our model. Employing pre-trained CNN, namely ResNet-50, VGG 16, and Inception-v3, we encode images to extract meaningful features, thus facilitating caption generation. Following the feature extraction phase, we employ recurrent neural network (RNN) architectures such as LSTM and GRU to decode these features and generate coherent captions. Fine-tuning the model parameters and conducting rigorous evaluations ensure the production of accurate and contextually relevant descriptions for a wide range of images.

This project serves as a foundational exploration into the realm of image captioning, showcasing its potential impact on various domains and paving the way for future advancements. Through this report, we aim to provide a comprehensive overview of our journey in developing an image captioning bot. From data preprocessing to model architecture and evaluation metrics, each aspect of our methodology is meticulously documented. By sharing our insights and findings, we hope to contribute to the collective knowledge base in the field of computer vision and inspire further innovation in this technology.

1.2 PROBLEM STATEMENT

Developing accurate and contextually relevant image captioning systems presents significant challenges despite their potential benefits. Existing approaches struggle with generating coherent and semantically meaningful captions, limiting their practical utility. Additionally, scalability, dataset limitations, and evaluation discrepancies hinder the widespread adoption of image captioning technology in real-world scenarios. The dynamic nature of visual content and the diversity of human language further complicate the task of generating accurate captions. Addressing these challenges requires innovative approaches that leverage advances in deep learning, natural language processing, and multimodal understanding. Through collaborative efforts and interdisciplinary research, we can overcome these hurdles and unlock the full potential of image captioning for various applications.

To address these challenges, a multidisciplinary approach is needed, encompassing advancements in deep learning architectures, dataset curation, and evaluation metrics. Our goal is to develop an image captioning system that generates accurate, contextually relevant captions while exhibiting robustness, scalability, and adaptability across diverse applications.

SIGNIFICANCE OF THE PROBLEM

The significance of the problem statement lies in the potential of image captioning within the realm of computer vision. By enabling machines to generate descriptive and contextually relevant captions for images, this project addresses challenges in understanding and interpreting visual data. This capability holds substantial implications across diverse real-world applications, including but not limited to enhancing accessibility for the visually impaired, augmenting surveillance systems, improving image search functionalities, and facilitating autonomous navigation for vehicles.

At its core, this project seeks to harness the power of deep learning techniques to bridge the semantic gap between visual content and textual descriptions. By training models on large-scale datasets, which encompasses a vast array of images with multiple associated captions, the project endeavors to equip AI systems with the ability to comprehend and articulate the content of images in a manner akin to human perception. This advancement not only enhances the interpretability of visual data but also paves way for innovations in other fields.

1.3 EMPIRICAL STUDY

Prior to the implementation of our image captioning project, a comprehensive empirical study was conducted to assess the current landscape of image captioning techniques, existing tools, and their effectiveness in generating descriptive captions for images. This study aimed to provide valuable insights into the state-of-the-art methodologies, identify gaps in the existing solutions, and lay the groundwork for our project's development.

The empirical study began with an extensive review of academic literature, research papers, and industry reports focused on image captioning and related fields such as computer vision and natural language processing. Key findings revealed a growing interest in leveraging deep learning architectures, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for image captioning tasks. Several notable approaches, including those based on encoder-decoder frameworks and attention mechanisms, were identified as leading methodologies for generating contextually relevant captions.

Furthermore, an analysis of existing image captioning tools and platforms was conducted to understand their functionalities, limitations, and user experiences. Popular platforms such as Microsoft's COCO dataset and Google's Image Captions API were evaluated in terms of their ease of use, accuracy in generating captions, and scalability. However, limitations such as the reliance on pre-defined datasets and generalization issues were observed, highlighting the need for more robust and adaptable solutions. Additionally, feedback from users and developers of these tools provided valuable insights into the practical challenges and opportunities in the field of image captioning.

Overall, the empirical study underscored the importance of advancing image captioning technologies to address real-world challenges and opportunities. By synthesizing insights from academic research, existing tools, and user feedback, we gained a deeper understanding of the current landscape and identified areas for innovation. These findings informed the design and development of our project, guiding our choice of methodologies, dataset selection, and evaluation criteria to ensure the creation of a robust and impactful image captioning solution. The iterative nature of our approach allowed for continuous refinement and improvement. The iterative nature of our approach facilitated continuous refinement and improvement, enabling us to adapt to emerging trends and challenges in this field.

1.4 BRIEF DESCRIPTION OF THE SOLUTION APPROACH

The image captioning project harnesses the capabilities of deep learning, specifically focusing on neural networks (NN), to autonomously generate descriptive captions for images. At the core of our solution lies the fusion of Convolutional Neural Networks (CNNs) for image feature extraction and Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM), for sequence generation. This approach aligns with the cutting-edge advancements in deep neural networks, allowing our model to comprehend and articulate the content of images effectively.

In our project, we leverage pre-trained CNN models such as ResNet-50, VGG-16, and Inception-v3 for image feature extraction. These models have been pre-trained on large-scale image datasets and possess the capability to extract high-level features from images efficiently. By utilizing pre-trained models, we benefit from their learned representations of visual features, saving computational resources and time during training. Incorporating these models enhances the model's ability to capture intricate details and nuances within images.

The CNN component of our model processes input images, extracting salient features that encapsulate the visual content. These features serve as a rich representation of the image, capturing its essential characteristics. Subsequently, the extracted features are fed into the RNN and LSTM layers, which are responsible for generating coherent and contextually relevant captions. This sequential processing enables the model to comprehend the visual context and generate captions that accurately describe the content of the images. By combining the strengths of CNNs for feature extraction and RNNs/LSTMs for sequence generation, our approach ensures a holistic understanding of the images.

The use of large datasets, such as the Flickr 8k dataset divided into training, development, and test sets, ensures that the model learns from diverse image samples, enhancing its generalization capabilities. By employing a combination of CNNs, RNNs, and LSTM networks along with pre-trained models, our solution approach aims to deliver robust and accurate image captioning capabilities, showcasing the potential of deep learning in addressing real-world challenges across various domains. By fine-tuning these models on domain-specific data and incorporating attention mechanisms, we can improve the model's ability to capture intricate details and generate more contextually relevant captions. Additionally, ongoing research in transfer learning and multimodal fusion techniques holds promise for further enhancing the performance and versatility of image captioning systems.

1.6 COMPARISON OF EXISTING APPROACHES TO THE PROBLEM FRAMED

These are some of the prominent approaches to image captioning that have been explored in the literature, each approach has its strengths and weaknesses.

Table 1.1: Comparison of different existing approaches

APPROACH	DESCRIPTION
Template-Based Methods	Template-based methods rely on predefined templates or rules to generate captions for images. These methods often require manually crafted templates or linguistic rules to match image features with predefined textual descriptions. While they are straightforward to implement, they lack flexibility and struggle to capture the diversity of image content. Template-based approaches are limited in their ability to adapt to varying image contexts and may produce generic or repetitive captions.
Attention Mechanism	Attention mechanisms aim to improve the performance of image captioning models by selectively focusing on relevant image regions while generating captions. These methods dynamically adjust the attention of the model based on the saliency of different image regions, allowing the model to attend to the most informative parts of the image when generating each word of the caption. Attention mechanisms have been shown to enhance the contextual relevance and coherence of generated captions, leading to improvements in caption quality and accuracy.
Reinforcement Learning	Reinforcement learning (RL) approaches involve training image captioning models using a reward-based framework, where the model learns to generate captions through trial and error. RL methods typically involve an agent that interacts with the environment (i.e., the image) and receives rewards based on the quality of the generated captions. By optimizing caption generation to maximize cumulative rewards, RL-based approaches can adapt to diverse image contexts and produce captions that are both descriptive and semantically relevant. However, RL methods often require extensive training and may suffer from issues such as reward sparsity.

Transformer-Based Models	Transformer-based models have gained popularity in image captioning tasks due to their ability to capture long-range dependencies and semantic relationships within the input data. These models leverage self-attention mechanisms to aggregate information from different parts of the input sequence, enabling them to generate captions based on a holistic understanding of the image. Transformer architectures, such as the Transformer-XL and BERT, have demonstrated promising results in image captioning by effectively modeling contextual information and capturing fine-grained image details.

Chapter-2 LITERATURE SURVEY

2.1 SUMMARY OF PAPERS STUDIED

Paper Title: "Image Caption Generation using Deep Learning Technique"

Authors: Chetan Amritkar and Vaishali Jabade

Published Date: 2018

The paper presents a novel approach to image caption generation utilizing a combination of CNNs and RNNs with Long Short-Term Memory (LSTM) units. The CNN functions as an image encoder, extracting meaningful features from input images through pre-trained networks such as VGG. These extracted features are then fed into the LSTM-based RNN decoder, which sequentially generates words to construct coherent captions.

Throughout the training process, the model optimizes parameters to maximize the likelihood of producing accurate captions given input images. This optimization is achieved through the utilization of a predefined loss function based on negative log-likelihood. Evaluation of the model's performance employs metrics such as BLEU scores, indicating the effectiveness of the generated captions. High BLEU scores signify accurate and semantically aligned captions.

Table 2.1: Performance Evaluation of Image Caption Generation Model [1]

DATASET	BLEU SCORE
Flickr8k	0.53356
Flickr30k	0.61433
MSCOCO	0.67257

This approach represents a significant advancement in both computer vision and natural language processing domains, bridging the gap between visual perception and linguistic comprehension with unprecedented effectiveness. The integration of advanced techniques such as attention mechanisms and multimodal learning approaches enhances the model's ability to capture intricate visual details.

Paper Title: "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention"

Authors: Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio

Published Date: Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015.

Convolutional neural networks (CNNs) are widely utilized for extracting high-level visual features from images, followed by their decoding into sequential language tokens by recurrent neural networks (RNNs) or transformers, forming descriptive captions. Attention mechanisms, a key focus of research, enhance caption quality by enabling models to selectively focus on relevant image regions, akin to human visual attention. Variants like soft and hard attention refine this process, with soft attention computing a weighted sum of image features based on relevance scores, and hard attention directly selecting image regions to attend to.

Table 2.2: BLEU Scores for Model Evaluation [2]

DATASET	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Flickr8k	67.0	45.7	31.4	21.3
Flickr30k	66.9	43.9	29.6	19.9
COCO	71.8	50.4	35.7	25.0

Recent studies underscore significant advancements in image captioning, with models achieving high accuracy and improved descriptive quality. Integration of multimodal learning approaches has further enhanced performance by leveraging cross-modal interactions. Consequently, image captioning systems can now produce human-like descriptions of complex visual scenes, facilitating applications in image understanding, content indexing, and assistive technologies. the incorporation of attention mechanisms has refined the focus ability of image captioning models. The incorporation of attention mechanisms has refined the focus ability of image captioning models. attention mechanisms enable image captioning models to dynamically focus on relevant regions of the image while generating descriptions, improving the coherence and relevance of the generated captions.

Paper Title: " Deep Learning based Automatic Image Caption Generation "

Authors: Varsha Kesavan, Vaidehi Muley, and Megha Kolhekar

Published Date: October 18-20, 2019, at the Global Conference for Advancement in Technology (GCAT) in Bangalore, India.

The paper presents a comprehensive exploration of various deep learning architectures for automatic image caption generation. The study delves into the effectiveness of different encoder-decoder architectures, including VGG-16, InceptionV3, and Inception-ResNetV2, when coupled with GRU-based decoders. Notably, the research demonstrates commendable performance metrics, particularly with the VGG-16 model, which, even without attention mechanisms, produced captions of significant semantic relevance to input images.

However, the authors highlight significant improvements achieved by integrating attention mechanisms with InceptionV3 and Inception-ResNetV2 models. This enhancement enables the decoder to selectively focus on relevant image features, thereby improving caption quality. Evaluation using BLEU scores consistently favored the attention-based models.

Table 2.3: BLEU Scores of Image Captioning Models [3]

MODEL	BLEU SCORE
VGG-16 (without attention)	-
InceptionV3	0.7912
Inception-ResNetV2	0.765

The experimental results provide empirical evidence supporting the efficacy of attention mechanisms in image captioning architectures. By offering more accurate and contextually relevant descriptions of visual content, these findings contribute significantly to the advancement of image processing and natural language generation technologies, the incorporation of attention mechanisms not only improves the quality of generated captions but also enables better interpretability of model decisions, enhancing trust and transparency in AI systems, also facilitate better interpretability of model decisions.

Paper Title: "Visual Image Caption Generator Using Deep Learning" [4]

Authors: Grishma Sharma, Priyanka Kalena, Nishi Malde, Aromal Nair, and Saurabh Parkar

Published Date: 2nd International Conference on Advances in Science & Technology (ICAST-2019)

The paper delves into the realm of generating accurate image captions, a pivotal task with wide-ranging applications in artificial intelligence, from robotic vision to assisting visually impaired individuals. The study underscores the significance of employing deep learning and machine learning methodologies, leveraging datasets like Flickr 8k to train models effectively.

The researchers compare the efficacy of LSTM and GRU architectures for sentence generation, employing BLEU scores as performance metrics. They highlight the evolution of image captioning techniques from template-based solutions to RNN-based approaches, which excel in natural language processing tasks. Notably, LSTM and GRU architectures address challenges such as the vanishing gradient problem, ensuring robust learning and prediction of captions. The comparison sheds light on the trade-offs between LSTM and GRU architectures, revealing nuanced differences in their performance and training characteristics. While LSTM architectures excel in capturing long-term dependencies and producing coherent captions, GRU architectures offer advantages in terms of computational efficiency.

Their methodology integrates three key models: Feature Extraction, Encoder, and Decoder. The Feature Extraction model harnesses VGG-16 to extract comprehensive image features, which are then processed by an Encoder model equipped with LSTM for caption generation. The Decoder model seamlessly integrates these features, predicting words based on contextual image information. Evaluation through BLEU scores sheds light on the performance disparities between LSTM and GRU models. LSTM achieves a perfect BLEU-1 score of 1.0, indicating exact matches with reference captions, while both architectures score 0.5 for BLEU-2, suggesting partial overlap in bigram sequences.

Further comparison experiments between VGG+LSTM and VGG+GRU models reveal nuanced differences in training time and performance. Despite GRU offering faster training, LSTM consistently outperforms in capturing detailed caption semantics. The study concludes by advocating for LSTM's superiority in accuracy. These findings underscore the importance of carefully selecting the appropriate architecture for image captioning tasks, considering both performance and training efficiency.

Paper Title: "Automatic Image Caption Generation using Deep Learning"

Authors: Akash Verma, Arun Kumar Yadav, Mohit Kumar, and Divakar Yadav

Published Date: 1 June 2023

The study explores the intricate task of automatically generating descriptive captions for images, integrating principles from natural language processing and computer vision domains. Utilizing the VGG-16 Hybrid Places 1365 model as an encoder and LSTM as a decoder, the proposed method harnesses pre-trained CNN architectures to extract visual features and generate coherent captions through recurrent neural networks. The utilization of pre-trained CNN architectures such as VGG-16 Hybrid Places 1365 enhances the efficiency and effectiveness of image feature extraction, capturing intricate details.

Table 2.4: Evaluation Metrics and Scores for Image Captioning Model [5]

EVALUATION METRIC	SCORE
BLEU-1	0.6666
BLEU-2	0.4704
BLEU-3	0.3893
BLEU-4	0.2878
ROUGE-L	0.3076
METEOR	0.5060
GLEU	0.2469

The experimental findings underscore the superiority of the VGG-16 Hybrid Places 1365 and LSTM model over existing approaches in terms of caption quality and accuracy. Notably, the model achieves a BLEU score of 0.6666 on the Flickr8k dataset, indicating its strong performance in producing captions aligned with reference captions. Furthermore, evaluations on random live images emphasize the model's robustness and versatility, validating its effectiveness in real-world applications.

2.2 INTEGRATED SUMMARY OF THE LITERATURE STUDIED

The research papers underscore the significant strides made in the domain of image caption generation using deep learning techniques. Across these studies, researchers demonstrate a keen focus on improving the accuracy and contextual relevance of automated image descriptions, addressing challenges inherent in tasks ranging from aiding the visually impaired to enhancing robotic vision systems. Leveraging diverse datasets such as Flickr 8k and employing deep learning architectures like VGG-16, InceptionV3, and Inception-ResNetV2, these investigations highlight the pivotal role of convolutional neural networks (CNNs) in effectively extracting image features, thereby laying a robust foundation for subsequent caption generation processes. The findings from these research papers contribute to a deeper understanding of the complexities involved in generating descriptive captions for images, shedding light on the nuances of model training, dataset selection, and evaluation metrics.

Attention mechanisms emerge as a central theme in enhancing the descriptive quality of automated image captions. Studies reveal that integrating attention mechanisms with CNN-LSTM/GRU architectures significantly improves caption quality by enabling the model to selectively focus on relevant image features. The comparative evaluations using metrics like BLEU scores consistently demonstrate the superiority of attention-based models over their counterparts, with attention-enhanced variants achieving higher scores across multiple evaluation criteria. These findings underscore the importance of contextual understanding in generating accurate and meaningful image captions, paving the way for advancements in both image processing and natural language generation technologies.

The comparison between different deep learning architectures and recurrent neural network (RNN) variants offers valuable insights into the strengths and weaknesses of each approach. While LSTM architectures generally excel in capturing fine-grained details and achieving higher accuracy in caption generation, GRU architectures exhibit advantages in terms of training efficiency and speed. This nuanced understanding of model performance and trade-offs informs researchers' decision-making process, guiding the selection of appropriate architectures based on specific application requirements and constraints. these studies collectively contribute to a deeper understanding of the intricate interplay between DL techniques, attention mechanisms, and image caption generation. Additionally, such comparative studies pave the way for further advancements in deep learning architectures tailored to address the evolving demands of image captioning tasks, fostering innovation and progress in the field.

CHAPTER 3 – REQUIREMENT ANALYSIS AND SOLUTION APPROACH

3.1 OVERALL DESCRIPTION

The Image Caption Generator is a web-based application crafted to generate descriptive captions for uploaded images employing machine learning techniques. Users can engage with the system via a user-friendly web interface, allowing them to upload images of their preference.

3.1.1 Product Perspective

1. The Image Caption Generator website operates as an independent web application.
2. Through a web interface, users can upload images and interact directly with the backend system.
3. Flask is employed in the backend to manage user requests and deliver generated captions.
4. It incorporates both the Inception-v3 and LSTM models for generating image captions.
5. Additionally, the system integrates with external APIs like the Google Images API to retrieve relevant images.

3.1.2 Product Functions

1. Users have the option to upload images directly to the website.
2. The system utilizes machine learning models to automatically generate captions for the uploaded images.
3. It showcases the generated captions alongside relevant images sourced from Google.
4. For registered users, the system retains uploaded images and their corresponding captions within the user's history table in the database.

3.1.3 User Characteristics

1. The users accessing the Image Caption Generator website may exhibit a wide range of technical proficiency levels and may possess varying degrees of familiarity with ML principles.
2. Among the user base are individuals seeking to generate descriptive captions for their images, as well as those intrigued by the functionalities and potential applications of machine learning algorithms.
3. Some users may require additional assistance or guidance in navigating the website and understanding the image captioning process, highlighting the importance of clear and intuitive user interfaces and comprehensive user support resources.

3.1.4 Constraints

1. There exists a limitation on the computational resources necessary to effectively train and operate the machine learning model.

3.1.5 Assumptions and Dependencies

1. A reliable internet connection is essential for accessing external APIs and services, which the system relies on for various functionalities.
2. The system operates under the assumption that users will supply suitable images for caption generation and that the resulting captions will be pertinent and precise.

3.1.6 Apportioning of Requirements

1. Future iterations may refine caption accuracy and enhance performance.

3.2 REQUIREMENT ANALYSIS

3.2.1 Functional requirements

1. **User Authentication:**
    - Users should have the capability to register for an account, facilitating personalized experiences and access to exclusive features.
    - Upon successful registration, users should be able to securely log in to their accounts, ensuring data privacy and account security.
    - Access control mechanisms should be implemented to restrict unauthorized users from accessing sensitive functionalities, such as saving image-caption pairs to their history. This measure guarantees the safeguarding of user data, ensuring that only authorized individuals have access to it.
2. **Image Upload:**
    - Users must be capable of uploading images in standard formats like JPEG and PNG using the web interface.
    - The system needs to verify uploaded images to guarantee compliance with size and format specifications.
    - Uploaded images should be securely stored to ensure confidentiality and integrity throughout subsequent processing stages.

3. **Caption Generation:**
    - The system must employ state-of-the-art machine learning models such as Inception-v3 and LSTM to effectively analyze and interpret the visual content of uploaded images
    - The captions generated must not only precisely depict the content of the images uploaded by the user but also capture subtle details and nuances, providing informative and engaging descriptions
    - Caption generation should be completed within a reasonable timeframe, leveraging efficient algorithms and optimized processing techniques to ensure timely delivery of captions
4. **Related Images Display:**
    - The system ought to retrieve related images from Google through the Google Images API.
    - The related images displayed should be pertinent to the content of the uploaded image, augmenting the user's comprehension.
    - The presentation of related images should be visually appealing and well-organized to enhance user experience.
5. **User History Management:**
    - The system must retain a record of images uploaded by individual users, along with their corresponding generated captions.
    - Users should have access to their upload history and the captions associated with each image.
    - Users must be provided with the capability to selectively delete particular entries from their upload history.

3.2.2 Non-Functional requirements

1. **Performance:**
    - The system should be capable of handling multiple user requests simultaneously without significant performance degradation.
    - Caption generation should occur within a reasonable time frame, even during periods of high user traffic.
    - Response times for user interactions (e.g., uploading images, viewing captions) should be minimal to provide a responsive user experience.

2. **Security:**
    - User authentication and session management should be implemented securely to prevent unauthorized access to user accounts and data. Uploaded images and user data should be stored securely to protect user privacy.
    - The system should be resistant to common security threats, such as SQL injection and cross-site scripting (XSS).
3. **Scalability:**
    - The system architecture should be designed to accommodate future growth and scalability.
    - It should be capable of handling an increasing number of users and images without compromising performance or reliability.
4. **Reliability:**
    - The system should be reliable and available for use 24/7, with minimal downtime for maintenance or updates.
    - Data integrity should be maintained to ensure that uploaded images, captions, and user history remain accurate and consistent.
5. **Usability:**
    - The user interface should be intuitive and easy to use for users of varying technical backgrounds.
    - Clear and informative error messages should assist users in effectively troubleshooting issues.
    - Ensuring compliance with accessibility standards is essential for enabling comfortable application use for all users.

3.2.3 Logical Database requirements

1. **User Information Table:**
    - Attributes: id (Primary Key), e-mail id, name, password
    - Description: This table stores user authentication information. Each user is assigned a unique id. Email id is used for login purposes, and passwords are securely stored for authentication. Implementing proper access control mechanisms to ensure that only authenticated users can access and modify data.

2. **User History Table:**
    - Attributes: id (Primary Key), user_id (Foreign Key), image_path, caption
    - Description: This table maintains a history of images uploaded by users along with their generated captions. It is linked to the User Information table via the user_id foreign key, allowing for easy retrieval of user-specific data.
3. **Database Relationships:**
    - Relationship between User Information and User History: One-to-Many
    - Each user can have multiple entries in the User History table, representing their uploaded images and the corresponding generated captions.
    - The one-to-many relationship is established through the user_id attribute, which serves as the foreign key in the User History table, linking each entry to the respective user's information stored in the User Information table.
    - This relational structure allows for efficient retrieval of user-specific upload history and facilitates personalized user experiences.

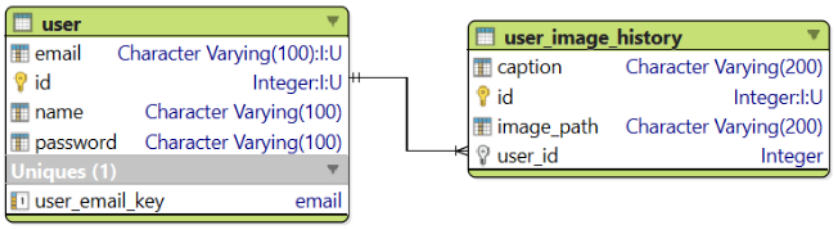


Fig 3.1: Database schema

4. **Data Integrity Constraints:**
    - Primary Key Constraints: Each record in a table has a unique identifier (e.g., id).
    - Foreign Key Constraints: referential integrity between related tables (e.g., user_id in User History references id in User Information).
    - NOT NULL Constraints: Presence of essential attributes (e.g., email, name, password).
    - Unique Constraints: Unique values for certain attributes (e.g., email).
    - Default Constraints: Providing default values for columns when no value is explicitly specified during insertion (e.g., setting a default value for the "status" column to "active").

3.3 SOLUTION APPROACH

The solution approach for the Image Caption Generator project involves a combination of deep learning algorithms and web development techniques to create a system capable of generating descriptive captions for uploaded images. The core components of the solution include feature extraction using pre-trained convolutional neural network (CNN) models and caption generation using a Long Short-Term Memory (LSTM) network. Additionally, a Flask backend is employed to handle user requests and serve the generated captions through a user-friendly web interface.

1. **Image Feature Extraction:**

   The first module of the solution involves extracting features from uploaded images using pre-trained CNN models. Specifically, three widely used CNN architectures - ResNet-50, Inception-v3, and VGG-16 - are utilized to extract high-level features from the images. Each CNN model can capture different aspects of the image's content, such as edges, textures, and object shapes. These features serve as input to the caption generation module.

   a. **ResNet-50:**

   ResNet-50 architecture consists of multiple building blocks called residual blocks, each containing a set of convolutional layers followed by shortcut connections (identity mappings). These shortcut connections allow the network to bypass certain layers, facilitating the direct flow of information from earlier layers to deeper layers.

   ResNet-50's innovative design with residual blocks and shortcut connections revolutionized deep learning architectures, addressing the vanishing gradient problem, and enabling the training of significantly deeper networks with improved performance and efficiency. This breakthrough in architecture design has paved the way for more sophisticated CNN.

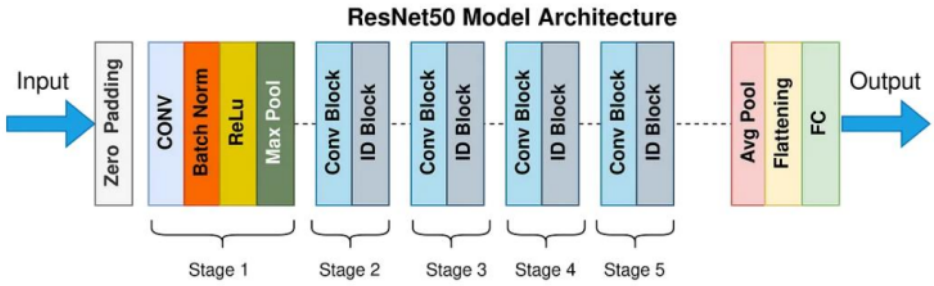


Fig 3.2 ResNet-50 model architecture [6]

b. Inception-v3:

Inception-v3 represents a convolutional neural network architecture crafted by Google's DeepMind team, renowned for its prowess in image classification. Belonging to the esteemed Inception lineage, it stands out for its efficiency and efficacy. Enhancements over its forerunners include the integration of factorized convolutions, batch normalization, and rigorous regularization techniques.

These include the utilization of factorized convolutions, batch normalization, and the application of aggressive regularization techniques, all contributing to its superior performance in handling complex visual data. Moreover, Inception-v3 demonstrates a remarkable ability to capture intricate patterns and features within images.

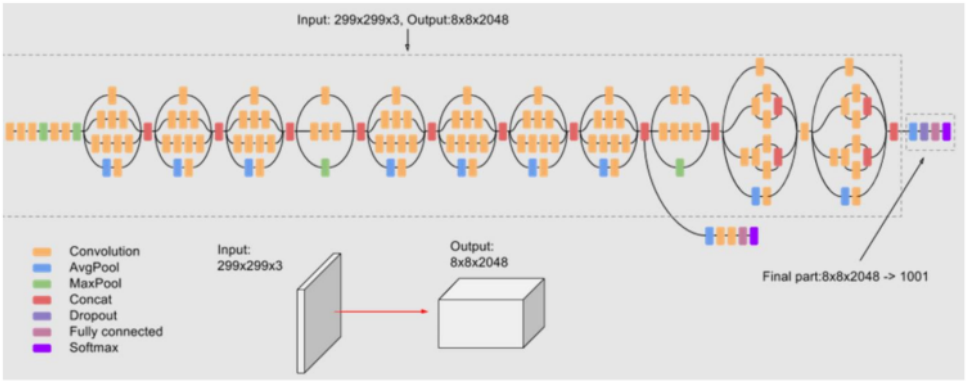


Fig 3.3 Feature vector extraction from Inception-v3 [7]

c. VGG-16:

The VGG-16 architecture, devised by the Visual Geometry Group at the University of Oxford, has significantly influenced the landscape of deep learning architectures. Its inception marked a pivotal moment in the field, introducing a standardized and straightforward approach to convolutional neural networks (CNNs). By employing a stack of multiple layers with small receptive fields, VGG-16 demonstrated the importance of depth in neural network architectures, challenging the prevailing notion that deeper networks are harder to train. This architectural simplicity, coupled with its remarkable performance on image classification.

The design philosophy behind VGG-16 prioritizes interpretability and reproducibility, making it an attractive choice for both researchers and practitioners. The consistent use of 3x3 convolutional filters throughout the network enhances interpretability, as each layer's transformation can be readily understood. Moreover, this uniformity facilitates model replication and experimentation, enabling researchers to explore various modifications and extensions with ease. Consequently, VGG-16 serves as a foundational model for advancing the understanding of deep neural networks and continues to inspire innovations.

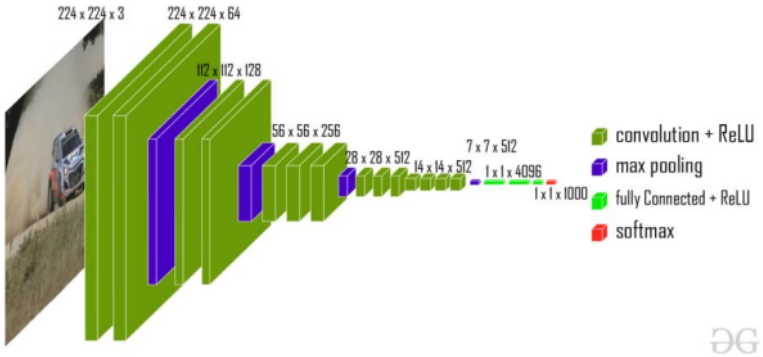


Fig 3.4 VGG-16 architecture [8]

2. **Caption Generation:**

Once the image features are extracted, they are fed into an LSTM network for generating descriptive captions. The LSTM network is trained on a large corpus of image-caption pairs to learn the relationships between image features and corresponding captions. During inference, the LSTM generates a sequence of words based on the input image features, gradually constructing a coherent caption that describes the content of the image.

3. **Flask Backend:**

The Flask backend serves as the interface between the user and the caption generation system. It handles user requests, such as uploading images and retrieving generated captions, through HTTP endpoints. Upon receiving an image upload request, the backend triggers the image feature extraction process using the pre-trained CNN models. Subsequently, the extracted features are passed to the LSTM network for caption generation.

24

4. User Interface:

The user interface provides a user-friendly web interface for interacting with the Image Caption Generator system. It allows users to upload images from their devices and view the corresponding generated captions. Additionally, the interface may display related images fetched from external sources, such as the Google Images API, to provide additional context and visual references for the uploaded images. User authentication and history tracking functionalities may also be integrated into the interface to enhance user experience and personalization.

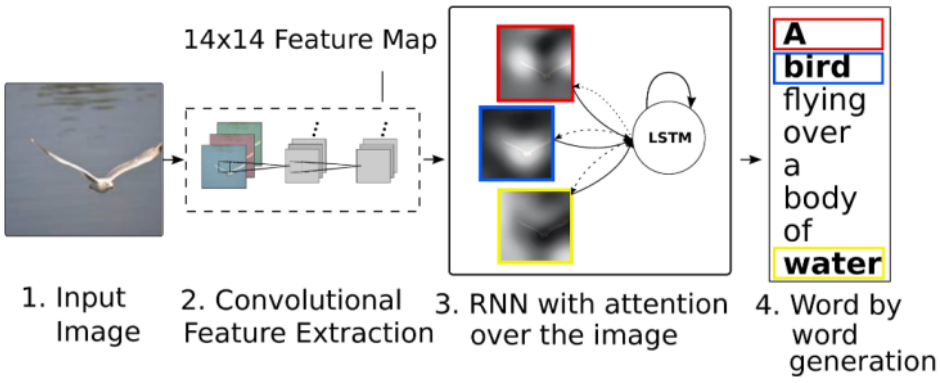


Fig 3.5: Diagram depicting image captioning bot working [9]

Chapter-4 MODELING AND IMPLEMENTATION DETAILS

4.1 DESIGN DIAGRAMS

4.1.1 Use Case diagrams

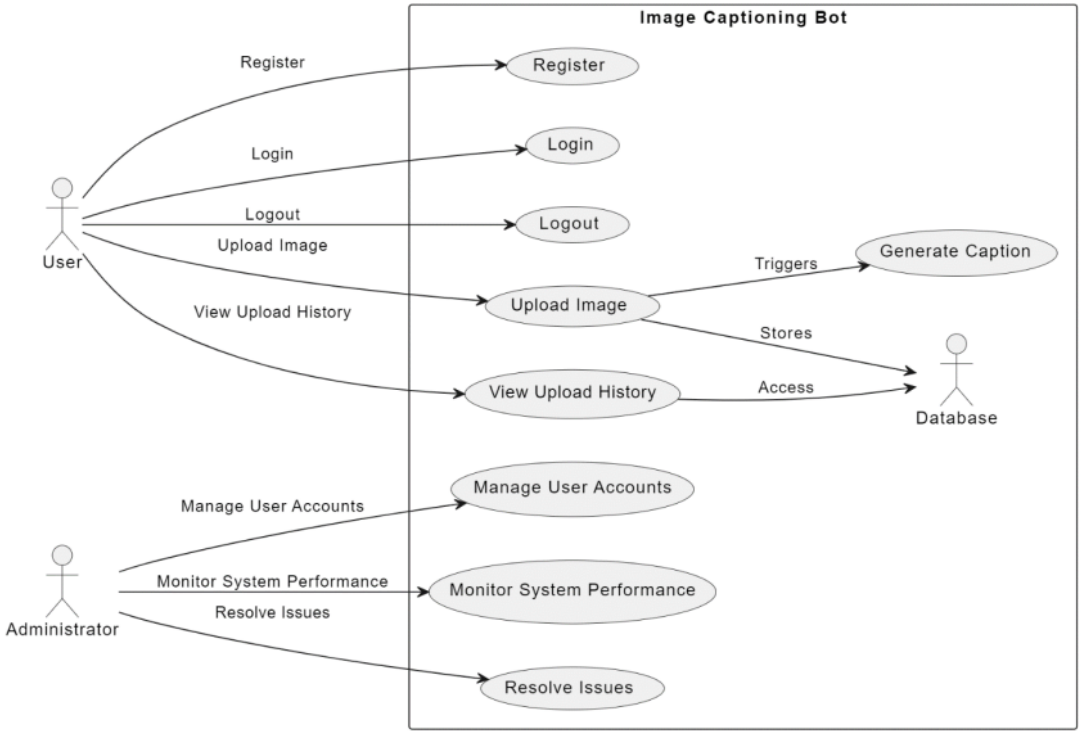


Fig 4.1: Use Case diagram for Image Captioning Bot

4.1.2 Control Flow Diagram

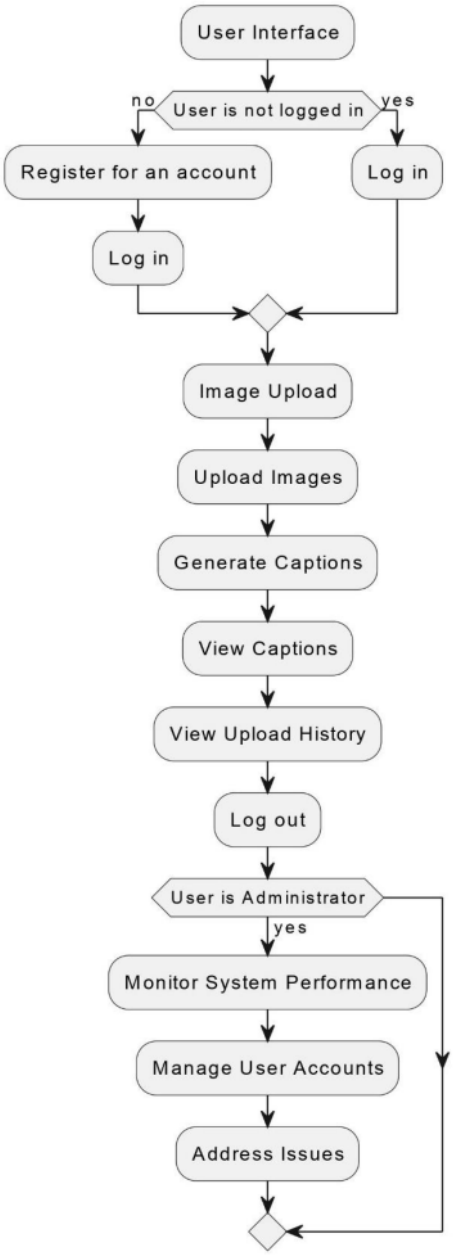


Fig 4.2: Control Flow Diagram for Image Captioning Bot

4.1.3 Sequence Diagram

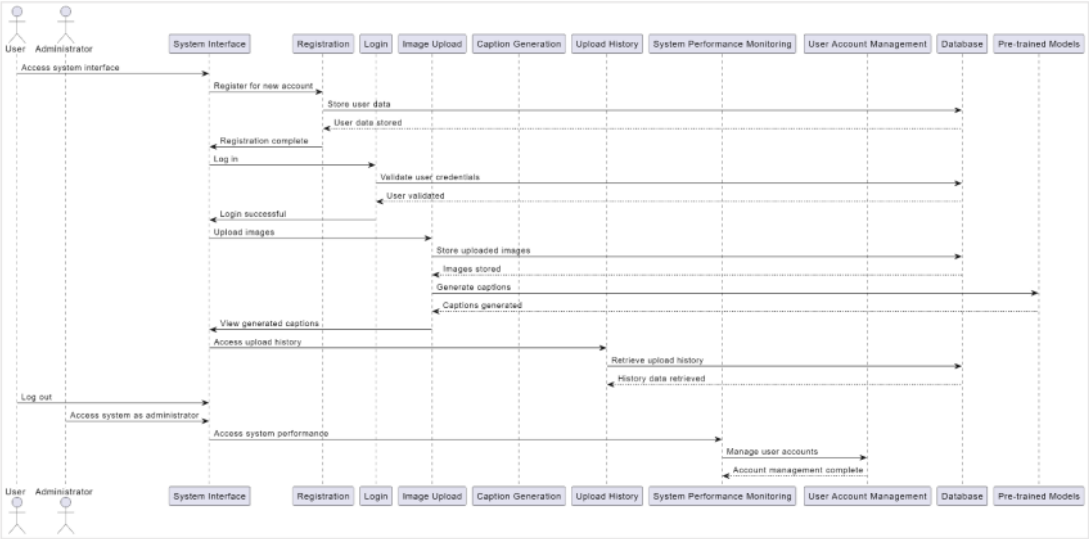


Fig 4.3: Sequence Diagram for Image Captioning Bot

4.2 IMPLEMENTATION DETAILS AND ISSUES

In our implementation, we utilized the Flickr 8k dataset, comprising 8,000 images, each accompanied by five captions. This dataset served as the foundation for training and evaluating our image captioning model and has provided ample training data to capture a wide range of visual concepts and linguistic expressions. Now, let us delve into the specifics of our implementation.

4.2.1 Data pre-processing

During the data cleaning phase, various pre-processing steps were applied to the text data to ensure its suitability for model training. The goal was to create a clean and standardized representation of the textual information contained in the image captions. The following steps were performed:

1. Creating descriptions dictionary:

   One of the files "Flickr8k.token.txt" in dataset contained the name of each image along with its 5 captions. To read the 5 captions for a single image, we created the dictionary to map each image with list of captions it has.

```
# Dictionary
Image_Description = {}

for index,row in captions.iterrows():
    image_name = row['Image'].split('.')[0]
    caption = row['Caption']
    if image_name not in Image_Description:
        Image_Description[image_name] = []
    Image_Description[image_name].append(caption)
```

```
Image_Description['1000268201_693b08cb0e']
```

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

Fig 4.4: Dictionary named 'Image_Description'

2. Basic Text Cleaning:

1. Lowercasing: To ensure consistency and prevent the model from treating words with different cases as distinct entities, all words were converted to lowercase. This step mitigates issues such as 'hello' and 'Hello' being considered separate words.

2. Special Token Removal: Special characters and tokens, such as '%', '$', and '#', were removed from the text. This simplifies the vocabulary and avoids unnecessary complexity in the model.

3. Elimination of Numeric Words: Words containing numbers, such as 'hey199', were eliminated from the text. This step helps in creating a more linguistically relevant.

```
import re
def clean_caption(captions):
    # Lowercasing
     for i in range(len(captions)):
        caption = captions[i]
        caption = caption.lower()
        caption = re.sub("[^a-z]+"," ",caption)
        caption = caption.strip()
        captions[i] = caption

for image_name, caption in Image_Description.items():
    clean_caption(caption)
len(Image_Description)
```

```
8091
```

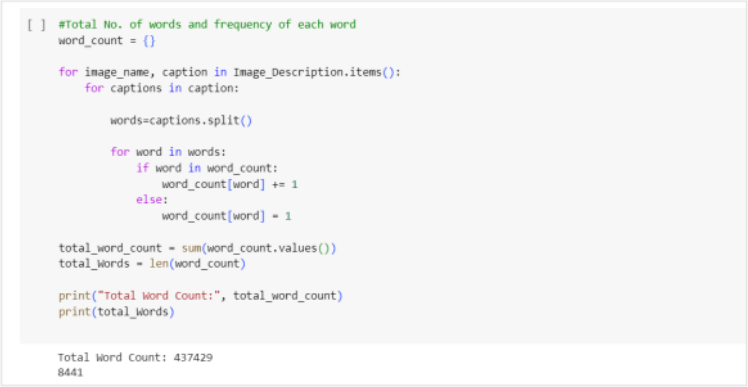
Fig 4.5: Code for data cleaning

3. Vocabulary Creation:

   A vocabulary containing all unique words across the entire dataset was created. The dataset consisted of 40,000 image captions from the 8,000 images, each having 5 captions. The initial vocabulary size was found to be 8,424 unique words.

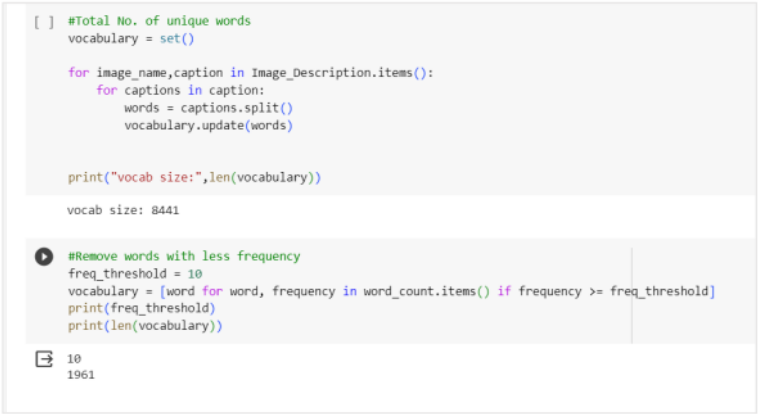
4. Pruning the Vocabulary:

   To enhance the model's predictive capabilities and reduce the risk of overfitting to rare words, a decision was made to filter the vocabulary. Only words occurring at least 10 times in the entire corpus were retained.



```
[ ] #Total No. of words and frequency of each word
    word_count = {}

    for image_name, caption in Image_Description.items():
        for captions in caption:

            words=captions.split()

            for word in words:
                if word in word_count:
                    word_count[word] += 1
                else:
                    word_count[word] = 1

    total_word_count = sum(word_count.values())
    total_Words = len(word_count)

    print("Total Word Count:", total_word_count)
    print(total_Words)

    Total Word Count: 437429
    8441
```

Fig 4.6 Code for counting frequency of each word



```
[ ] #Total No. of unique words
    vocabulary = set()

    for image_name,caption in Image_Description.items():
        for captions in caption:
            words = captions.split()
            vocabulary.update(words)


    print("vocab size:",len(vocabulary))

    vocab size: 8441

    #Remove words with less frequency
    freq_threshold = 10
    vocabulary = [word for word, frequency in word_count.items() if frequency >= freq_threshold]
    print(freq_threshold)
    print(len(vocabulary))

    10
    1961
```

Fig 4.7: Code for threshold frequency

4.2.2 Loading training image names

The file "Flickr_8k.trainImages.txt" served as a reference to identify the images belonging to the training set. The names of these images were loaded into a Python list named "train". This step ensured a clear separation of the 6,000 images earmarked for training.

```
[ ] import random

    image_names = list(Image_Description.keys())

    len(image_names)

    8091

    from sklearn.model_selection import train_test_split
    random.seed(42)

    random.shuffle(image_names)

    train_images, test_val_images = train_test_split(image_names, test_size=0.2, random_state=42)
    val_images, test_images = train_test_split(test_val_images, test_size=0.5, random_state=42)

    train_set = {image_name: Image_Description[image_name] for image_name in train_images}

    print("Train Set:", len(train_set))
    print("Validation Set:", len(val_images))
    print("Test Set:", len(test_images))

    for image_name, captions in train_set.items():
        train_set[image_name] = [f"startseq {caption} endseq" for caption in captions]

    Train Set: 6472
    Validation Set: 809
    Test Set: 810
```

Fig 4.8: Separating training images in the list

The descriptions of the training images were loaded from the "descriptions.txt" file, which was previously saved on the hard disk. During this loading process, two essential tokens were added to each caption:

- 'startseq': This token was appended at the beginning of every caption, signifying the start of the sequence. It served as an indicator for the model to commence generating a caption.
- 'endseq': This token was added at the end of every caption, marking the conclusion of the generated sequence. It signaled to the model that the caption had been completed.

The inclusion of the 'startseq' and 'endseq' tokens provided clear delineation of caption boundaries, aiding in the proper processing and interpretation of captions by the caption generation model. This systematic preprocessing step laid the foundation for training the model to generate relevant captions for input images.

4.2.3 Image Feature Extraction

During this phase, our focus was on preparing the image data for the model. Our chosen approach centered on transfer learning, utilizing the ResNet-50, VGG-16, and InceptionV3 models.

1. **ResNet-50:**

RESNET-50 MODEL

```
[ ] model_resnet = ResNet50(weights = "imagenet",input_shape=(224,224,3))
    model_resnet.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3)]  0          []

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64)  9472       ['conv1_pad[0][0]']

 conv1_bn (BatchNormalizati     (None, 112, 112, 64)  256        ['conv1_conv[0][0]']
 on)

 conv1_relu (Activation)        (None, 112, 112, 64)  0          ['conv1_bn[0][0]']

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64)  0          ['conv1_relu[0][0]']

 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0          ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2     (None, 56, 56, 64)   4160       ['pool1_pool[0][0]']
 D)

 conv2_block1_1_bn (BatchNo     (None, 56, 56, 64)   256        ['conv2_block1_1_conv[0][0]']
 rmalization)

 conv2_block1_1_relu (Activ     (None, 56, 56, 64)   0          ['conv2_block1_1_bn[0][0]']
 ation)
```

Fig 4.9: ResNet-50 model summary

```
[ ] def encode_image_resnet(img):
        img = preprocess_img_resnet(img)
        feature_vector = model_new_resnet.predict(img)
        feature_vector = feature_vector.reshape((-1,))
        return feature_vector
```

```
    encoding_train_resnet = {}

    for ix,img_id in enumerate(train_set):
        img_path = IMG_PATH+img_id+".jpg"
        encoding_train_resnet[img_id] = encode_image_resnet(img_path)

        if ix%100==0:
            print("Encoding in Progress Time Step %d"%ix)
```

Fig 4.10: Image encoding using ResNet-50

```
[ ] import pickle
    with open("/content/drive/MyDrive/archive/Flickr8K/encoding_train_resnet_features.pkl","wb") as f:
        pickle.dump(encoding_train_resnet,f)
```

```
[ ] with open("/content/drive/MyDrive/archive/Flickr8K/encoding_train_resnet_features.pkl", 'rb') as file:
        train_features_resnet = pickle.load(file)
```

```
[ ] print(train_features_resnet['1000268201_693b08cb0e'].shape)

    (2048,)
```

```
[ ] encoding_test_resnet = {}

    for ix,img_id in enumerate(test_images):
        img_path = IMG_PATH+img_id+".jpg"
        encoding_test_resnet[img_id] = encode_image_resnet(img_path)

        if ix%100==0:
            print("Encoding in Progress Time Step %d"%ix)
```

```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
Encoding in Progress Time Step 0
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
```

Fig 4.11: Passing image to ResNet-50

2. Inception-v3:

INCEPTION V3 MODEL

```
model_inception = InceptionV3(weights='imagenet')
model_inception.summary()
```

```
Model: "inception_v3"
```

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 299, 299, 3)]	0	[]
conv2d (Conv2D)	(None, 149, 149, 32)	864	['input_2[0][0]']
batch_normalization (BatchNormalization)	(None, 149, 149, 32)	96	['conv2d[0][0]']
activation (Activation)	(None, 149, 149, 32)	0	['batch_normalization[0][0]']
conv2d_1 (Conv2D)	(None, 147, 147, 32)	9216	['activation[0][0]']
batch_normalization_1 (BatchNormalization)	(None, 147, 147, 32)	96	['conv2d_1[0][0]']
activation_1 (Activation)	(None, 147, 147, 32)	0	['batch_normalization_1[0][0]']
conv2d_2 (Conv2D)	(None, 147, 147, 64)	18432	['activation_1[0][0]']
batch_normalization_2 (BatchNormalization)	(None, 147, 147, 64)	192	['conv2d_2[0][0]']
activation_2 (Activation)	(None, 147, 147, 64)	0	['batch_normalization_2[0][0]']

✓ 0s completed at 5:36 PM

Fig 4.12: Inception-v3 model summary

```
[ ] model_new_inception = Model(model_inception.input, model_inception.layers[-2].output)

[ ] def preprocess_img_inception(img):
        img = image.load_img(img,target_size=(299,299))
        img = image.img_to_array(img)
        img = np.expand_dims(img,axis=0)

    #     Normalisation
        img = preprocess_input_inception(img)
        return img
```

Fig 4.13: Image pre-processing using Inception-v3

```
IMG_PATH = "/content/drive/MyDrive/archive/Flickr8K/Flicker8k_Images/"
import cv2
import matplotlib.pyplot as plt
img = preprocess_img_inception(IMG_PATH+"1000268201_693b08cb0e.jpg")
print(img[0].shape)
plt.imshow(img[0])
plt.show()
```

```
(299, 299, 3)
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

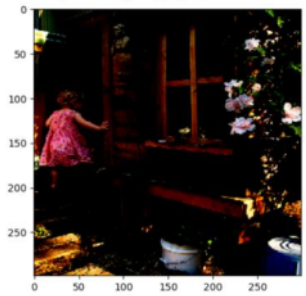


Fig 4.14: Passing image to Inception-v3

3. VGG-16:

VGG 16 MODEL

```
[ ] model_vgg16 = VGG16(weights='imagenet')
    model_vgg16.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 224, 224, 3)]     0
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
```

✓ 0s completed at 5:36 PM

Fig 4.15: VGG-16 model summary

```
[ ] model_new_vgg16 = Model(model_vgg16.input, model_vgg16.layers[-2].output)
```

```
[ ] def preprocess_img_vgg16(img):
        img = image.load_img(img,target_size=(224,224))
        img = image.img_to_array(img)
        img = np.expand_dims(img,axis=0)

    #     Normalisation
        img = preprocess_input_vgg16(img)
        return img
```

Fig 4.16: Image pre-processing using VGG-16

```
IMG_PATH = "/content/drive/MyDrive/archive/Flickr8K/Flicker8k_Images/"
import cv2
import matplotlib.pyplot as plt
img = preprocess_img_vgg16(IMG_PATH+"1000268201_693b08cb0e.jpg")
plt.imshow(img[0])
plt.show()
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats
```

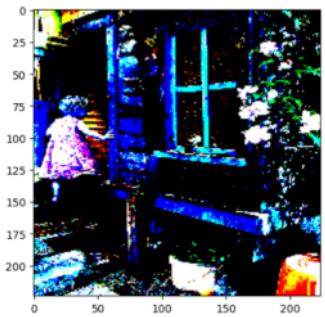


Fig 4.17: Passing an image to VGG-16

4.2.4 Caption pre-processing

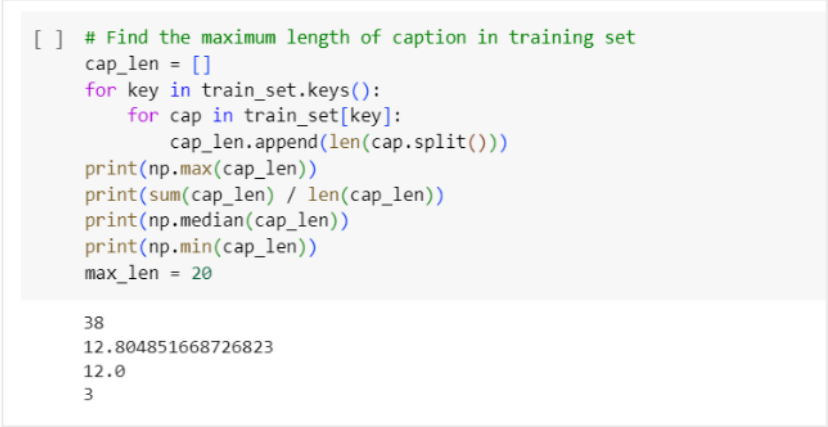
During the data preprocessing phase focused on captions, the goal was to prepare the captions for training, considering that they would serve as the target variables (Y) during the model training.

- The vocabulary was indexed and this indexing allowed for the representation of each unique word by an integer. These dictionaries served the purpose of mapping words to their corresponding indices and vice versa.

```
[ ] word_to_idx = {}
    idx_to_word = {}

    for i,word in enumerate(vocabulary):
        word_to_idx[word] = i+1
        idx_to_word[i+1] = word

[ ] print(len(idx_to_word))

    1961

    idx_to_word[1962] = 'startseq'
    word_to_idx['startseq'] = 1962

    idx_to_word[1963] = 'endseq'
    word_to_idx['endseq'] = 1963

    vocab_size = len(word_to_idx) + 1
    print("vocab size",vocab_size)

    vocab size 1964
```

Fig 4.18: Indexing vocabulary

- Another crucial parameter in the preprocessing step was calculating the maximum length of a caption. This information was crucial for setting appropriate parameters in the subsequent model design, ensuring that the model could effectively handle captions of varying lengths during training.



```
[ ] # Find the maximum length of caption in training set
    cap_len = []
    for key in train_set.keys():
        for cap in train_set[key]:
            cap_len.append(len(cap.split()))
    print(np.max(cap_len))
    print(sum(cap_len) / len(cap_len))
    print(np.median(cap_len))
    print(np.min(cap_len))
    max_len = 20

    38
    12.804851668726823
    12.0
    3
```

Fig 4.19: Finding maximum length of captions

4.2.5 Data Preparation using Generator Function

Given the large size of the dataset and memory constraints, the implementation necessitated the use of a generator function to handle data loading in batches. Generators provided an efficient solution for processing large datasets by loading and processing data on-the-fly during model training. By using generator functions, the need to store the entire dataset in memory was alleviated, leading to improved memory utilization and enhanced training efficiency.

```
def data_generator(train_set,features,word_to_idx,max_len,batch_size):
    x1,x2,y = [],[],[]

    n=0
    while True:
        for key,caption_list in train_set.items():
            n += 1

            photo = features[key]
            for caption in caption_list:
              # Converting each caption into a list of numbers
              seq = []
              for word in caption.split():
                if word in word_to_idx:
                  seq.append(word_to_idx[word])

                for i in range(1,len(seq)):
                    xi = seq[0:i]
                    yi = seq[i]

                    #0 denote padding word
                    xi = pad_sequences([xi],maxlen = max_len,value=0,padding='post')[0]
                    yi = to_categorical([yi],num_classes=vocab_size)[0]

                    x1.append(photo)
                    x2.append(xi)
                    y.append(yi)

                if n == batch_size:
                    yield[[np.array(x1),np.array(x2)],np.array(y)]
                    x1,x2,y = [],[],[]
                    n = 0
```

✓ 0s completed at 5:36 PM

Fig 4.20: Code for Data Generator function

4.2.6 Word embeddings

In the context of the project, word embeddings played a crucial role in mapping each word (index) to a vector. The following section provides a detailed account of this aspect:

1. Loading GLOVE Model:

   The GLOVE model was employed to generate word embeddings, offering a 200-dimensional representation for each word.

```
[ ] f = open(r"/content/drive/MyDrive/archive (1)/glove.6B.50d.txt", encoding = 'utf8')

[ ] embedding_index = {}

    for line in f:
        values = line.split()
        word = values[0]
        word_embedding = np.array(values[1:],dtype = 'float')
        embedding_index[word] = word_embedding

[ ] f.close()
    embedding_index['apple']

    array([ 0.52042 , -0.8314  ,  0.49961 ,  1.2893  ,  0.1151  ,  0.057521,
           -1.3753  , -0.97313 ,  0.18346 ,  0.47672 , -0.15112 ,  0.35532 ,
            0.25912 , -0.77857 ,  0.52181 ,  0.47695 , -1.4251  ,  0.858   ,
            0.59821 , -1.0903  ,  0.33574 , -0.60891 ,  0.41742 ,  0.21569 ,
           -0.07417 , -0.5822  , -0.4502  ,  0.17253 ,  0.16448 , -0.38413 ,
            2.3283  , -0.66682 , -0.58181 ,  0.74389 ,  0.095015, -0.47865 ,
           -0.84591 ,  0.38704 ,  0.23693 , -1.5523  ,  0.64802 , -0.16521 ,
           -1.4719  , -0.16224 ,  0.79857 ,  0.97391 ,  0.40027 , -0.21912 ,
           -0.30938 ,  0.26581 ])
```

Fig 4.21: Word embedding

2. Creating Embedding Matrix

   For unique words in the project's vocabulary, an embedding matrix was crafted.

```
[ ] def get_embedding_matrix():
        emb_dim = 50
        matrix = np.zeros((vocab_size,emb_dim))
        for word,idx in word_to_idx.items():
            embedding_vector = embedding_index.get(word)

            if embedding_vector is not None:
                matrix[idx] = embedding_vector
        return matrix

[ ] embedding_matrix = get_embedding_matrix()
    embedding_matrix.shape

    (1964, 50)
```

Fig 4.22: Generate embedding matrix

4.2.7 Model architecture

The architecture of the image captioning model was designed using the Functional API provided by the Keras library. Since the input consisted of two components— an image vector and a partial caption— the Sequential API was not suitable, and the Functional API allowed the creation of a Merge Model. The high-level sub-modules of the architecture were as follows:

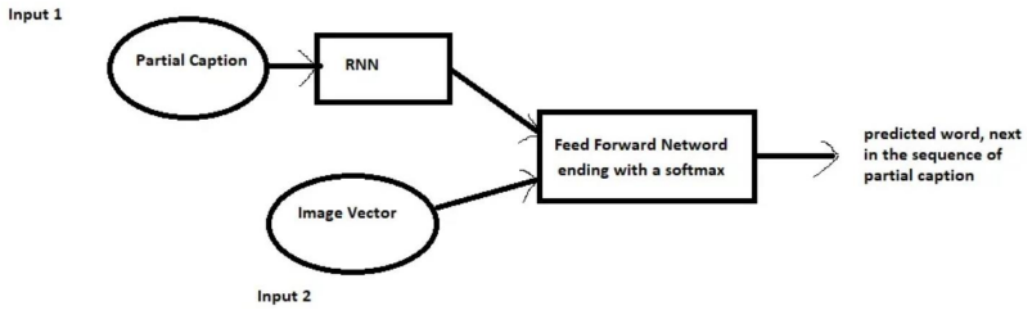


Fig 4.23 High level architecture [10]

```
[ ]  input_img_features = Input(shape=(2048,))
     inp_img1 = Dropout(0.3)(input_img_features)
     inp_img2 = Dense(256,activation='relu')(inp_img1)

     #Caption as Input
     input_captions = Input(shape=(max_len,))
     inp_cap1 = Embedding(input_dim = vocab_size, output_dim=50, mask_zero=True)(input_captions)
     inp_cap2 = Dropout(0.3)(inp_cap1)
     inp_cap3 = LSTM(256)(inp_cap2)

     decoder1 = add([inp_img2,inp_cap3])
     decoder2 = Dense(256,activation='relu')(decoder1)
     outputs = Dense(vocab_size,activation='softmax')(decoder2)

     # Combined model
     model = Model(inputs=[input_img_features,input_captions],outputs=outputs)

     model.summary()
```

Fig 4.24: Defining LSTM model

```
model.summary()
```

```
Model: "model_3"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_5 (InputLayer)        [(None, 20)]              0         []

 input_4 (InputLayer)        [(None, 2048)]            0         []

 embedding (Embedding)       (None, 20, 50)            98200     ['input_5[0][0]']

 dropout (Dropout)           (None, 2048)              0         ['input_4[0][0]']

 dropout_1 (Dropout)         (None, 20, 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, 1964)              504748    ['dense_1[0][0]']

==================================================================================================
Total params: 1507652 (5.75 MB)
Trainable params: 1507652 (5.75 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

Fig 4.25: LSTM model summary

```
[ ]  # Pre-initiate Embedding Layer
     model.layers[2].set_weights([embedding_matrix])
     model.layers[2].trainable = False
```

```
[ ]  model.compile(loss='categorical_crossentropy',optimizer="adam")
```

Fig 4.26: Initializing the embedding layer

4.2.8 Training of models

1. **ResNet-50:**

```
[ ]  epochs = 30
     batch_size = 32
     steps = len(train_set)//batch_size
```

```
for i in range(epochs):
    generator = data_generator(train_set,train_features_resnet,word_to_idx,max_len,batch_size)
    model.fit(generator,epochs=1,steps_per_epoch = steps, verbose=1)
    model.save('/content/drive/MyDrive/archive/Flickr8K/model_{}.keras'.format(i))
```

```
202/202 [==============================] - 31s 90ms/step - loss: 4.4866
202/202 [==============================] - 18s 91ms/step - loss: 3.5676
202/202 [==============================] - 18s 90ms/step - loss: 3.2811
202/202 [==============================] - 19s 95ms/step - loss: 3.1074
202/202 [==============================] - 19s 93ms/step - loss: 2.9802
202/202 [==============================] - 19s 92ms/step - loss: 2.8758
202/202 [==============================] - 19s 92ms/step - loss: 2.7924
202/202 [==============================] - 18s 89ms/step - loss: 2.7220
202/202 [==============================] - 18s 89ms/step - loss: 2.6612
202/202 [==============================] - 19s 92ms/step - loss: 2.6105
202/202 [==============================] - 19s 93ms/step - loss: 2.5658
202/202 [==============================] - 18s 89ms/step - loss: 2.5272
202/202 [==============================] - 18s 91ms/step - loss: 2.4931
202/202 [==============================] - 18s 90ms/step - loss: 2.4586
202/202 [==============================] - 18s 91ms/step - loss: 2.4279
202/202 [==============================] - 18s 87ms/step - loss: 2.4001
```

Fig 4.27: Model training (ResNet-50)

2. Inception-v3:

```
for i in range(epochs):
    generator = data_generator(train_set,train_features_inception,word_to_idx,max_len,batch_size)
    model.fit(generator,epochs=1,steps_per_epoch = steps, verbose=1)
    model.save('/content/drive/MyDrive/archive/Flickr8K/inception_model_{}.keras'.format(i))
```

```
202/202 [==============================] - 20s 101ms/step - loss: 2.8844
202/202 [==============================] - 19s 95ms/step - loss: 2.6237
202/202 [==============================] - 21s 106ms/step - loss: 2.5317
202/202 [==============================] - 19s 94ms/step - loss: 2.4724
202/202 [==============================] - 19s 95ms/step - loss: 2.4283
202/202 [==============================] - 19s 92ms/step - loss: 2.3911
202/202 [==============================] - 19s 95ms/step - loss: 2.3580
202/202 [==============================] - 19s 93ms/step - loss: 2.3314
202/202 [==============================] - 19s 92ms/step - loss: 2.3074
202/202 [==============================] - 18s 90ms/step - loss: 2.2849
202/202 [==============================] - 18s 92ms/step - loss: 2.2636
202/202 [==============================] - 19s 97ms/step - loss: 2.2449
202/202 [==============================] - 18s 89ms/step - loss: 2.2273
202/202 [==============================] - 19s 95ms/step - loss: 2.2138
202/202 [==============================] - 18s 89ms/step - loss: 2.1988
202/202 [==============================] - 18s 89ms/step - loss: 2.1846
202/202 [==============================] - 18s 91ms/step - loss: 2.1721
202/202 [==============================] - 19s 93ms/step - loss: 2.1602
202/202 [==============================] - 18s 91ms/step - loss: 2.1495
202/202 [==============================] - 19s 96ms/step - loss: 2.1360
202/202 [==============================] - 18s 91ms/step - loss: 2.1261
202/202 [==============================] - 18s 90ms/step - loss: 2.1170
```

Fig 4.28: Model training (Inception-v3)

3. VGG-16:

```
# Pre-initiate Embedding Layer
model.layers[2].set_weights([embedding_matrix])
model.layers[2].trainable = False
model.compile(loss='categorical_crossentropy',optimizer="adam")
```

```
for i in range(epochs):
    generator = data_generator(train_set,train_features_vgg16,word_to_idx,max_len,batch_siz
    model.fit(generator,epochs=1,steps_per_epoch = steps, verbose=1)
    model.save('/content/drive/MyDrive/archive/Flickr8K/vgg16_model_{}.keras'.format(i))
```

```
202/202 [==============================] - 33s 109ms/step - loss: 4.5732
202/202 [==============================] - 22s 106ms/step - loss: 3.6058
202/202 [==============================] - 22s 108ms/step - loss: 3.2933
202/202 [==============================] - 21s 105ms/step - loss: 3.1026
202/202 [==============================] - 20s 100ms/step - loss: 2.9615
202/202 [==============================] - 21s 103ms/step - loss: 2.8536
```

✓ 0s completed at 5:36 PM

Fig 4.29: Model training (VGG-16)

4.2.9 Predictions

1. **Resnet-50:**

```
all_img_names = list(test_features_resnet.keys())
import numpy as np
import matplotlib.pyplot as plt

# Check the length of your list of image names
num_images = len(test_features_resnet)

# Loop to display random images
for _ in range(15):
    # Generate a random index within the valid range
    idx = np.random.randint(0, num_images)

    # Retrieve the image name using the generated index
    img_name = all_img_names[idx]

    # Assuming the rest of your code remains the same
    photo_2048 = test_features_resnet[img_name].reshape((1, 2048))
    img_path = r"/content/drive/MyDrive/archive/Flickr8K/Flicker8k_Images/" + img_name + ".jpg"
    i = plt.imread(img_path)
    model = load_model('/content/drive/MyDrive/archive/Flickr8K/model_29.keras')

    caption = predict_caption(photo_2048,model)

    print(caption)

    plt.imshow(i)
    plt.axis("off")
    plt.show()
```

Fig 4.30: Predicting captions (ResNet-50)

2. **Inception-v3:**

```
# #pick some random images and see results
all_img_names = list(test_features_inception.keys())
import numpy as np
import matplotlib.pyplot as plt

# Check the length of your list of image names
num_images = len(test_features_inception)

# Loop to display random images
for _ in range(15):
    # Generate a random index within the valid range
    idx = np.random.randint(0, num_images)

    # Retrieve the image name using the generated index
    img_name = all_img_names[idx]

    # Assuming the rest of your code remains the same
    photo_2048 = test_features_inception[img_name].reshape((1, 2048))
    img_path = r"/content/drive/MyDrive/archive/Flickr8K/Flicker8k_Images/" + img_name + ".jpg"
    i = plt.imread(img_path)
    model = load_model('/content/drive/MyDrive/archive/Flickr8K/inception_model_29.keras')
    caption = predict_caption(photo_2048,model)

    print(caption)

    plt.imshow(i)
    plt.axis("off")
    plt.show()
```

Fig 4.31: Predicting captions (Inception-v3)

3. **VGG 16:**

```
all_img_names = list(test_features_vgg16.keys())

import numpy as np
import matplotlib.pyplot as plt

# Assuming encoding_test_resnet and predict_caption functions are defined

# Check the length of your list of image names
num_images = len(test_features_vgg16)

# Loop to display random images
for _ in range(15):
    # Generate a random index within the valid range
    idx = np.random.randint(0, num_images)

    # Retrieve the image name using the generated index
    img_name = all_img_names[idx]

    # Assuming the rest of your code remains the same
    photo_2048 = test_features_vgg16[img_name].reshape((1, 4096))
    img_path = r"/content/drive/MyDrive/archive/Flickr8K/Flicker8k_Images/" + img_name + ".jpg"
    i = plt.imread(img_path)
    model = load_model('/content/drive/MyDrive/archive/Flickr8K/vgg16_model_29.keras')
    caption = predict_caption_vgg16(photo_2048,model)

    print(caption)

    plt.imshow(i)
    plt.axis("off")
    plt.show()
```

Fig 4.32: Predicting captions (VGG-16)

4.2.10 Website Creation

The creation of our website encompasses several essential components, each tailored to enhance user experience and functionality. Integration of our image captioning model with the website is seamlessly achieved through Flask, a lightweight and flexible web framework for Python, we have PostgreSQL as our database.

1. **Signup/Login Page:**

The signup/login page provides users with a secure means to access our website. Through robust authentication mechanisms, user credentials are securely stored and validated against encrypted data in the database. This ensures the protection of sensitive information and fosters trust among users. Furthermore, secure authentication enables personalized experiences tailored to individual users, thereby enriching user engagement and satisfaction.

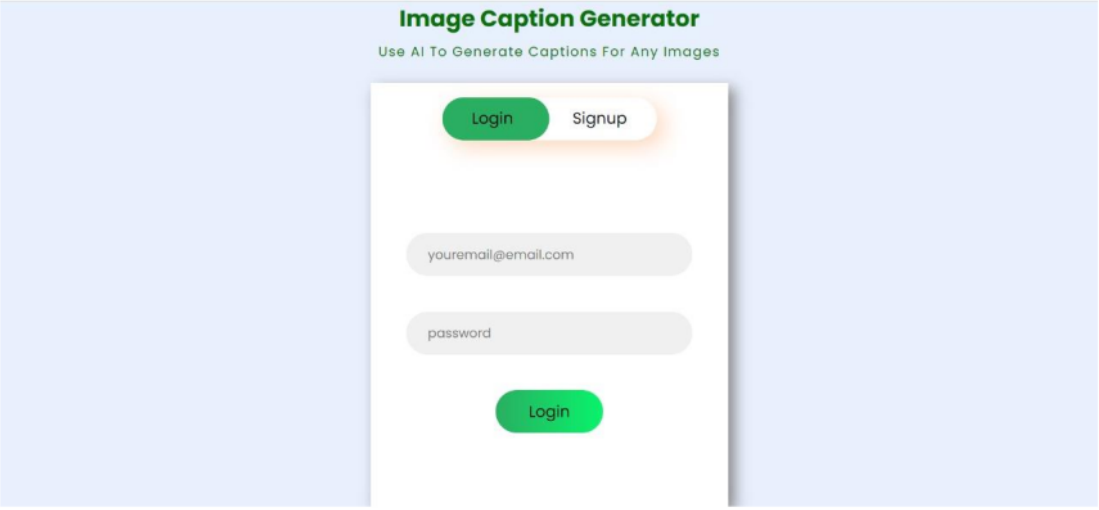


Fig 4.33: Login/Sign up page

2. **Home Page:**

Our home page serves as the central hub for users to interact with our image captioning bot. Its intuitive interface guides users through the process of uploading images seamlessly. Integration with the image captioning bot enhances user engagement, offering a unique and interactive feature directly accessible from the home page. Upon image upload, the backend processes the images and generates captions, which are then displayed to users, completing the interactive experience.

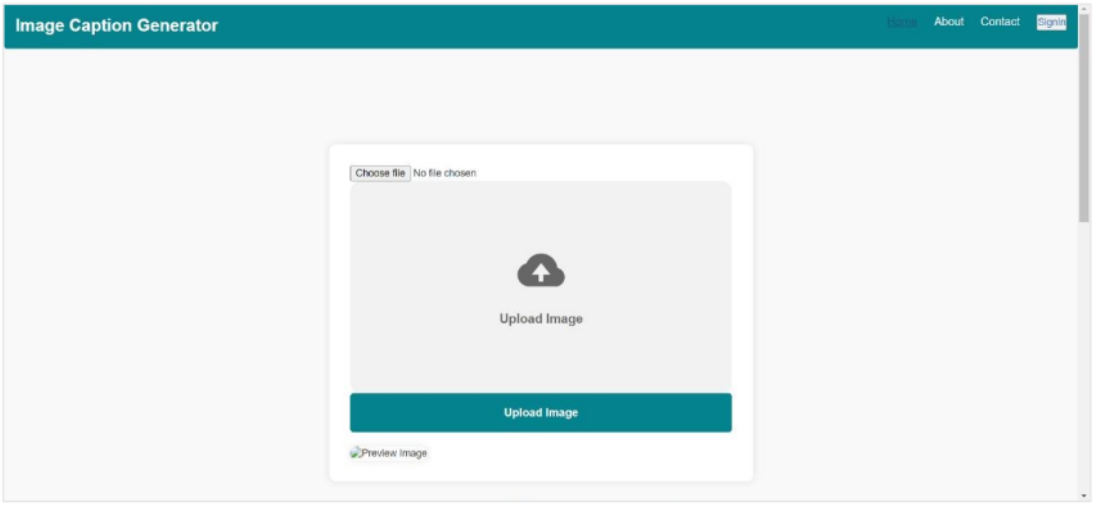


Fig 4.34 Home page (before uploading image)

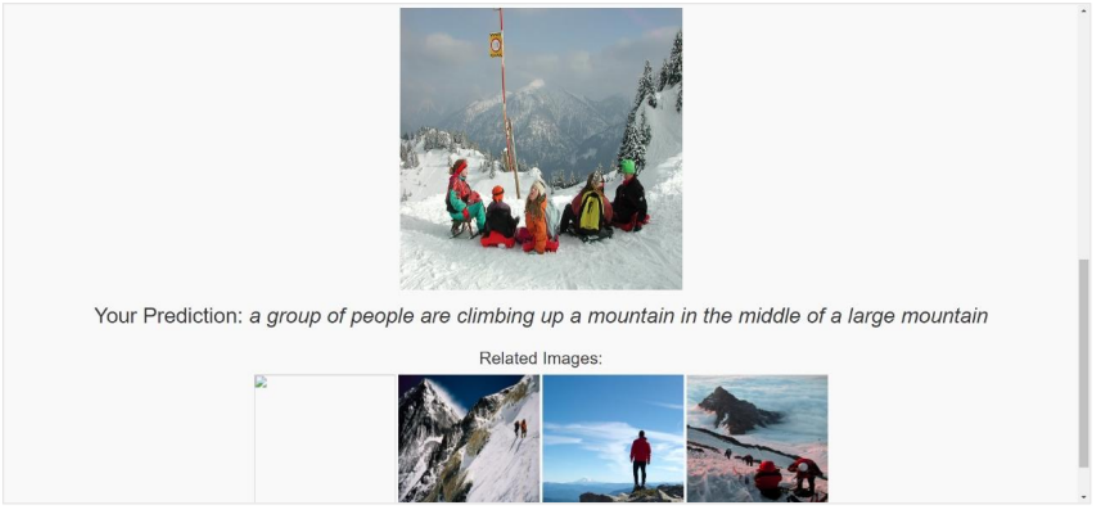


Fig 4.35: Home page (after uploading image)

3. **Contact Us Page:**

The contact us page facilitates communication between users and our team, serving as a vital channel for feedback, issue reporting, and inquiries. Users can provide feedback on their experiences, report any issues encountered, or seek assistance regarding website features. This direct line of communication enhances user engagement and support, demonstrating our commitment to addressing user concerns and improving the overall experience.

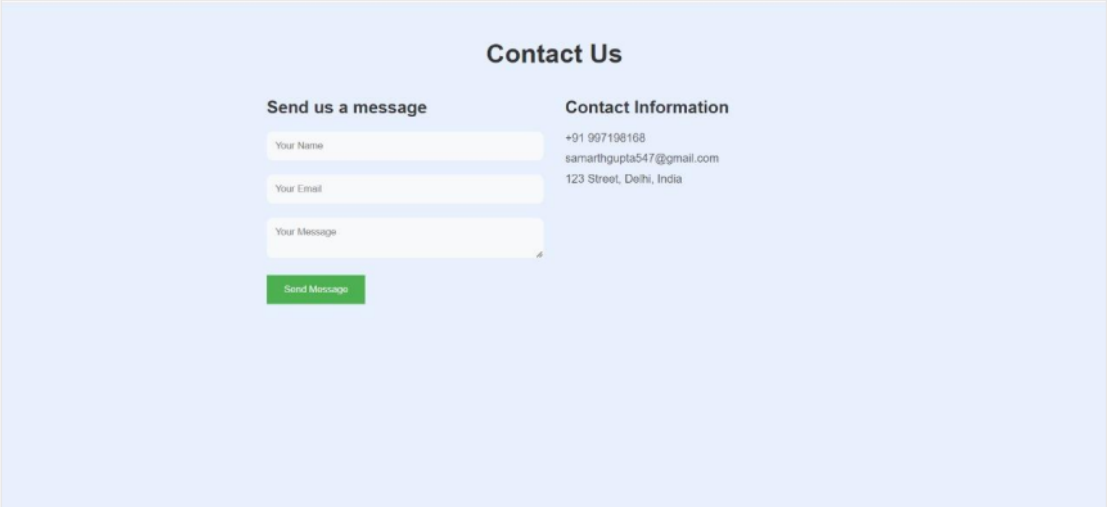


Fig 4.36: Contact us page

4. **User History:**

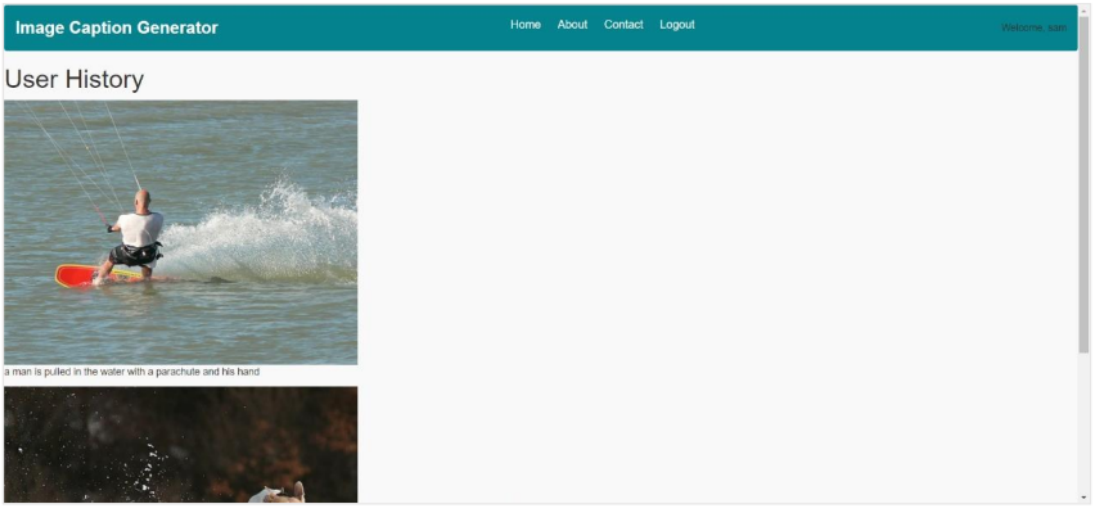


Fig 4.37: User History page

Chapter-5 TESTING

5.1 TESTING PLAN

Table 5.1: Testing Types

TESTING TYPE	WILL TEST BE PERFORMED?	COMMENTS	SOFTWARE COMPONENT	TESTING TYPE
Requirements Testing	Yes	This type of testing is essential to ensure that the image caption generator meets the specified requirements, such as generating accurate captions for a wide variety of images.	Image Captioning Algorithm	Requirements Testing
Unit Testing	Yes	Unit testing involves testing individual units of code to ensure that they behave as expected. Since the project includes three algorithms each algorithm's components will be tested to ensure they return the correct output given certain inputs.	ResNet-50, Inception-v3, VGG-16	Unit Testing
Integration Testing	Yes	The integration testing involves testing how the different components of the system work together. For this project, integration between the image processing module and the caption generation module will be tested to ensure that images are correctly processed and captions are generated.	Image Processing Module, Caption Generation Module	Integration Testing
Performance Testing	Yes	Performance testing is essential to ensure that the image caption generator can handle a large volume of requests within an acceptable response time. Since the project involves generating captions for uploaded images, the performance of the caption generation process will be tested.	Image Captioning Algorithm	Performance Testing

Stress Testing	No	Since the project is basic and does not include mechanisms to handle high loads or stress conditions, stress testing is not necessary.	N/A	Stress Testing
Compliance Testing	No	Compliance testing ensures that the system complies with relevant standards or regulations. Since the project is basic and does not have the specific requirements, compliance testing is not applicable.	N/A	Compliance Testing
Security Testing	No	Security testing ensures that the system is protected from unauthorized access or malicious attacks. Since the project is basic and does not include security features, security testing is not applicable.	N/A	Security Testing
Load Testing	No	Load testing ensures that the system can handle expected loads without crashing or slowing down. Since the project is basic and does not include mechanisms to handle high loads, load testing is not applicable.	N/A	Load Testing
Volume Testing	No	Volume testing ensures that the system can handle a large volume of requests without crashing or slowing down. Since the project is basic and does not include mechanisms to handle high loads, volume testing is not applicable.	N/A	Volume Testing
Error Handling Testing	No	Error handling testing ensures that the system can handle unexpected errors gracefully and provide appropriate error messages. Since the project is basic and does not include error handling mechanisms, error handling testing is not applicable.	N/A	Error Handling Testing

Table 5.2: Test Team Details

ROLE	NAME	SPECIFIC RESPONSIBILITIES/COMMENTS
Test Lead	Abhay	Abhay oversaw the overall testing process, including test planning, coordination with the development team, and ensuring that testing activities aligned with project objectives. He also reviewed test results and communicated testing progress to project stakeholders.
Test Analyst	Chandrika	Chandrika was responsible for requirements analysis, test case design, and execution. She collaborated closely with the development team to ensure that test cases covered all functional requirements and participated in defect triage meetings to prioritize and track issues.
Test Engineer	Samarth	Samarth focused on test execution, including running test cases, logging defects, and verifying fixes. He also assisted in test environment setup and configuration, ensuring that test environments mirrored production environments as closely as possible.

Table 5.3: Test Schedule

ACTIVITY	START DATE	COMPLETION DATE	HOURS	COMMENTS
Obtain Input Data	04/10/24	04/10/24	2	Input data, including sample images and captions, was obtained for testing purposes.
Test Environment Setup	04/11/24	04/12/24	4	The test environment was set up, including configuring the image captioning algorithms and database connections.

Test Case Design	04/13/24	04/14/24	6	Test cases were designed based on project requirements and functional specifications.
Test Execution	04/15/24	04/17/24	12	Test cases were executed to validate the functionality of the image captioning bot and ensure accurate caption generation.
Defect Tracking	04/18/24	04/19/24	4	Any defects identified during testing were tracked, prioritized, and resolved by the test team.

Table 5.4 Test Environment

SOFTWARE ITEMS	
Operating System	Ubuntu 20.04 LTS
Image Captioning Models	ResNet-50, Inception-v3, VGG-16
Database	PostgreSQL 12
Test Framework	PyTest
Development Environment	Python 3.8
	Dependencies installed locally, pre-trained models downloaded and integrated.
HARDWARE ITEMS	
Processor	Intel Core i7
Storage	SSD

5.2 COMPONENT DECOMPOSITION AND TYPE OF TESTING REQUIRED

Table 5.5: Component Decomposition and Identification of Tests required

S. NO.	LIST OF VARIOUS COMPONENTS (MODULES)	TYPE OF TESTING REQUIRED
1.	Image Feature Extraction	Requirement
2.	Caption Generation	Requirement
3.	Flask Backend	Unit
4.	User Interface	System

5.3 LIST ALL TEST CASES

Table 5.6: Test cases for component Image Feature Extraction

TEST CASE ID	INPUT	EXPECTED OUTPUT	STATUS
1.	Image with notable objects	Descriptive caption generated	Pass
2.	Image with multiple objects	All objects identified	Pass
3.	Low-resolution image	Object in the image identified	Pass
4.	High-resolution image	Objects accurately identified	Pass
5.	High-resolution image	Objects accurately identified	Pass

Table 5.7: Test cases for component Caption Generation

TEST CASE ID	INPUT	EXPECTED OUTPUT	STATUS
1.	Image features extracted from a simple scene	“A dog is running in a field”	Pass
2.	Image extracted from a complex scene	“A group of people is sitting on a snowy mountain"	Pass

Table 5.8: Test cases for component Flask Backend

TEST CASE ID	INPUT	EXPECTED OUTPUT	STATUS
1.	POST request with valid image field	HTTP status code 200 with generated caption	Pass
2.	GET request to retrieve user upload history	HTTP status code 200 with user upload history	Pass
3.	GET request to retrieve related images for an image	HTTP status code 200 with related images URLs	Pass
4.	POST request with missing image file	HTTP status code 400 with error message in JSON format	Pass

Table 5.9: Test cases for component User Interface

TEST CASE ID	INPUT	EXPECTED OUTPUT	STATUS
1.	User register on the website with correct details	User information gets stored in database	Pass
2.	User register on website with incorrect details	User asked to enter correct details again	Pass
3.	User enter correct login id and password	User gets logged in	Pass
4.	User enters incorrect login id or password	User asked to enter correct details again	Pass
5.	Upload valid image file	Image is displayed on UI with generated caption	Pass
6.	Navigate between different UI pages	UI pages load and display content as expected	Pass

5.4 ERROR AND EXCEPTION HANDLING

After analyzing the test case results of the image captioning bot, the following debugging techniques were applied to correct the errors:

Table 5.10: Debugging techniques used

TEST CASE ID	TEST CASE FOR COMPONENT	DEBUGGING TECHNIQUE USED
3	Flask Backend	Backtracking
6	User Interface	Post-mortem debugging

- Backtracking: This technique was utilized to trace the sequence of events and identify the root cause of the failure in the Flask Backend component. By systematically analyzing the execution flow and pinpointing the error, the necessary adjustments were made to ensure the expected output was achieved.
- Post-mortem Debugging: In cases where errors occurred during user interface interactions, post-mortem debugging was employed. This involved examining logs and runtime data captured during the failure to understand the underlying issue. Through careful analysis and reconstruction of the scenario, the necessary fixes were implemented to rectify the user interface discrepancies.

After applying the debugging techniques, test case results are as follow:

Table 5.11: Test cases after debugging

TEST CASE ID	INPUT	EXPECTED OUTPUT	STATUS
3.	GET request for related images of an image	HTTP status code 200 with related images URLs	Pass
6.	Navigate between different UI pages	UI pages load and display content as expected	Pass

After applying the debugging techniques, all failed test cases were successfully corrected, and they now pass without any issues, demonstrating the effectiveness of employing systematic debugging approaches in identifying and resolving issues.

5.5 LIMITATIONS OF THE SOLUTION

1. Data Dependency: One significant limitation of image captioning bots is their reliance on large and diverse datasets for training. These models require substantial amounts of annotated image-caption pairs to learn effectively. Limited availability or quality of data can hinder the performance of the model, leading to inaccurate or irrelevant captions.

2. Biased Outputs: Image captioning models may exhibit biases present in the training data, leading to biased or stereotypical captions. If the training data is not sufficiently diverse or contains biased annotations, the model may generate captions that reflect those biases, which can be problematic, especially in applications where neutrality and inclusivity are essential.

3. Ambiguity Handling: Image captioning bots may struggle to handle ambiguity present in images, resulting in vague or nonsensical captions. Images with complex scenes or multiple objects can pose challenges for the model to accurately describe all relevant elements, leading to ambiguity in the generated captions.

4. Lack of Context Understanding: While image captioning models can generate captions based on visual features, they may lack broader context understanding. They may fail to incorporate contextual information beyond the visual content of the image, such as temporal or situational context, which can impact the relevance and coherence of the generated captions.

5. Performance Variability: The performance of image captioning models can vary based on factors such as the choice of architecture, training data, and hyperparameters. It may require extensive experimentation and fine-tuning to achieve optimal performance, and even then, the model's performance may vary across different datasets or types of images.

6. Resource Intensive: Training and deploying image captioning models can be computationally intensive and require significant resources in terms of computing power, storage, and time. This can be a limiting factor for individuals or organizations with limited resources or infrastructure in their domain.

Chapter-6 FINDINGS, CONCLUSION, AND FUTURE WORK

6.1 FINDINGS

Throughout the implementation of our image captioning project, several noteworthy findings emerged, shedding light on the effectiveness and performance of the developed system.

1. Model Performance Evaluation:

   The primary focus of our project was to assess the performance of three pre-trained models, namely ResNet-50, VGG-16, and InceptionV3, in generating descriptive captions for images. Through rigorous evaluation, we observed variations in the accuracy and coherence of captions generated by each model. Despite leveraging state-of-the-art architectures, we identified nuances in model performance influenced by factors such as model complexity and feature extraction capabilities of the model.

2. Effectiveness of Model Architectures:

   Our findings revealed that while all three models exhibited promising capabilities in generating captions, each had distinct strengths and weaknesses. ResNet-50 demonstrated superior performance in capturing fine-grained visual details, resulting in more precise and contextually relevant captions. VGG-16 excelled in capturing global image features, producing captions with a broader contextual understanding. InceptionV3 showcased a balance between detailed feature extraction and computational efficiency, making it a viable option for real-time captioning applications.

3. Impact of Dataset Size:

   An intriguing aspect of our project was the exploration of dataset size and its influence on model performance. Despite utilizing the Flickr 8k dataset, which contains a substantial number of images and captions, we observed limitations in the diversity and richness of visual content. The impact of dataset size on model generalization and adaptability highlighted the importance of incorporating larger and more diverse datasets to enhance model robustness and performance. The examination of dataset size prompted considerations regarding the representativeness of the training data and its alignment with real-world scenarios. This exploration underscored the need for continuous augmentation to capture a broader spectrum of visual contexts.

4. Predictions using each model:

Resnet-50:

Fig 5.1: Prediction result (ResNet-50)

Inception-v3:

Fig 5.2: Prediction result (Inception-v3)

VGG-16:

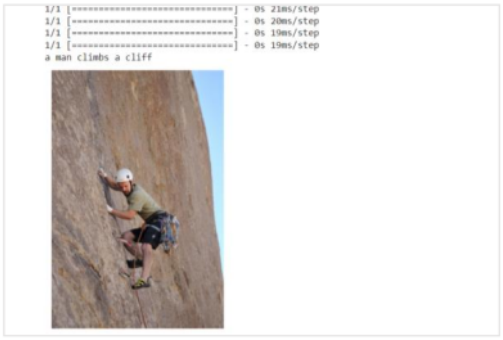


Fig 5.3: Prediction result (VGG-16)

5. Evaluation

To understand how good the model is, we tried to generate captions on images from the test dataset and found the BLEU Score for each model.

```
RESNET-50
BLEU-1: 0.561449
BLEU-2: 0.371495
BLEU-3: 0.241142
BLEU-4: 0.151472
INCEPTION V3
BLEU-1: 0.558786
BLEU-2: 0.374694
BLEU-3: 0.246545
BLEU-4: 0.155791
VGG16
BLEU-1: 0.528038
BLEU-2: 0.345001
BLEU-3: 0.217850
BLEU-4: 0.132301
```

Fig 5.4: BLEU scores

We looked at some examples where the captions are not very relevant and sometimes even irrelevant.

Fig 5.5 Output-1

In Fig 4.44, the output indicates that probably the color of the shirt got mixed with the color in the background.

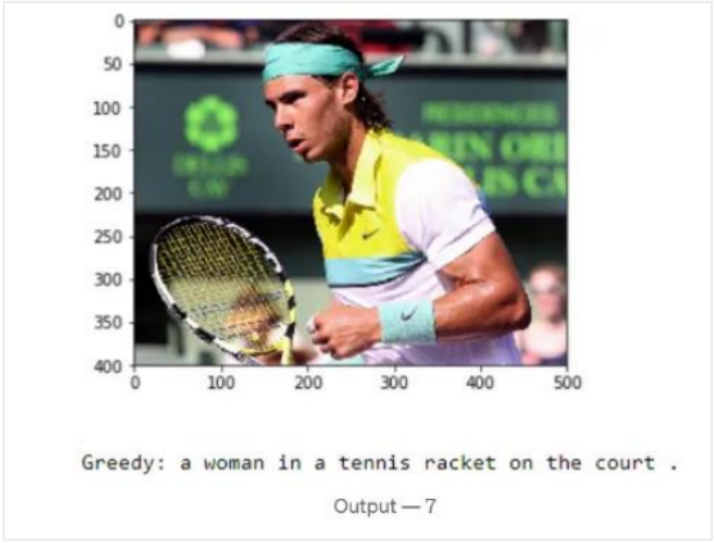


Fig 5.6: Output-2

In Fig 4.45, because of long hair model classified Rafael Nadal as a woman which is again misinterpretation.

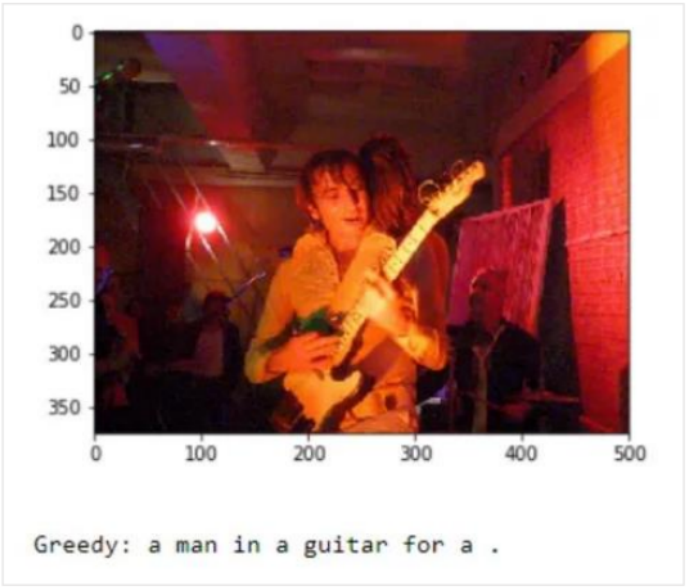


Fig 5.7: Output-3

Clearly, in fig 4.46 the model has tried its best to understand the scenario but still the caption is not a good one.

6.2 CONCLUSION

In conclusion, our project has explored the domain of image captioning, leveraging DL techniques to develop a robust and effective solution. Through meticulous data preprocessing, model training, and evaluation, we have demonstrated the potential of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) in generating descriptive captions for images. Our endeavor to integrate pre-trained models such as ResNet-50, VGG-16, and InceptionV3 has provided valuable insights into their respective strengths and limitations, guiding future research and development in this field. As we expand our dataset collection and explore novel model architectures, we broaden the scope of our solution and enhance its adaptability to diverse applications and environments.

The evaluation of our models using metrics like BLEU Score has provided quantitative measures of their performance, facilitating a comprehensive assessment of their effectiveness in generating contextually relevant captions. By analyzing the results and findings from our project, we have identified areas for improvement, including the exploration of larger and more diverse datasets, fine-tuning of model hyperparameters, and integration of advanced techniques like attention mechanisms. These insights serve as valuable contributions to the ongoing advancement of image captioning technology. y carefully analyzing the generated captions and understanding the underlying processes of the neural network models, we gain insights into their decision-making mechanisms and potential biases. By conducting thorough error analysis and studying failure cases, we can identify specific areas where the models may be underperforming or exhibiting biases, guiding future research efforts towards addressing these challenges and improving the overall reliability and inclusivity of image captioning systems.

Overall, our project underscores the significance of image captioning in enhancing the interpretability and accessibility of visual data. The successful implementation of our solution showcases the potential of deep learning in addressing real-world challenges and opens avenues for future research and innovation. As we continue to refine and iterate upon our methodology, we envision broader applications of image captioning across diverse domains, including but not limited to autonomous navigation, accessibility assistance, and content understanding. Through collaboration and continued exploration, we aim to push the boundaries of image captioning technology and contribute to its widespread adoption and impact. This understanding is crucial for ensuring the reliability and fairness of AI systems, particularly in sensitive domains where accurate interpretation is paramount.

6.3 FUTURE WORK

1. Enhancing User Interface (UI):

Improving the user interface of the website to enhance user experience and accessibility. This could involve implementing modern design principles, optimizing page load times, and incorporating intuitive navigation features to streamline user interactions.

2. Adding Additional Features to the Website:

Expanding the functionality of the website by integrating additional features such as image editing tools, social sharing options, or community forums. This would enrich the user experience and provide users with more tools to interact with the image captioning bot.

3. Improving Caption Quality:

Investigating techniques to enhance the quality and coherence of generated captions. This may involve exploring advanced natural language processing (NLP) techniques, fine-tuning model parameters, or incorporating external knowledge sources to improve caption understanding.

4. Enhancing Model Performance:

Continuously improving the performance of the image captioning models by fine-tuning architecture designs, experimenting with different optimization algorithms, or exploring novel training strategies. This would result in more accurate and contextually relevant captions for a wide range of images.

5. Exploring More Pre-Trained Models:

Investigating the effectiveness of additional pre-trained models for image captioning, beyond the ones already used in the project. This could include newer architectures, ensemble models, or models specifically trained on image captioning datasets to leverage their unique strengths and capabilities.

6. Implementing Multimodal Learning:

Exploring the integration of multimodal learning techniques to enhance caption generation by leveraging not only image features but also additional modalities such as text or audio. This approach could lead to more comprehensive and contextually rich descriptions of visual content.

REFERENCES

[1] C. Amritkar and V. Jabade, "Image Caption Generation Using Deep Learning Techniques," in 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-4.

[2] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, Yoshua Bengio Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:2048-2057, 2015.

[3] V. Kesavan, V. Muley and M. Kolhekar, "Deep Learning based Automatic Image Caption Generation," 2019 Global Conference for Advancement in Technology (GCAT), Bangalore, India, pp. 1-6, 2019.

[4] G. Sharma, P. Kalena, N. Malde, A. Nair, and S. Parkar, "Visual image caption generator using deep learning," in 2nd International Conference on Advances in Science & Technology (ICAST), 2019.

[5] A. Verma, A. Yadav, M. Kumar, et al., "Automatic image caption generation using deep learning," Multimed Tools Appl, vol. 83, pp. 5309–5325, 2024.

[6] S. Mukherjee, "The Annotated ResNet-50," Towards Data Science, https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758 , (accessed: April 13, 2024).

[7] "Inception-v3", Papers with Code, https://paperswithcode.com/method/inception-v3 , (Accessed: April 13, 2024).

[8] "VGG-16 CNN Model", GeeksforGeeks, [Online]. https://www.geeksforgeeks.org/vgg-16-cnn-model/ . (Accessed: April 13, 2024).

[9] Anish Shah, "Model", W&B AI, https://wandb.ai/fauxvo-faux/image_captioning/reports/Model--VmlldzoxNjM1Nzc1 . (Accessed: April 13, 2024).

[10] "Image Captioning with Keras", Towards Data Science, https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8 . (Accessed: April 13, 2024).

Chandrika Rajvanshi

Adaptable Tech Enthusiast

rajvanshichandrika@gmail.com | 9520665596 | Meerut, India | chandrika-rajvanshi | Github

EDUCATION

Bachelor of Technology in Computer Science, *JIIT, Noida* 2020 – 2024 | Noida
CGPA: 8.1

Class 12th, *Translam Academy International, Meerut* 2019 – 2020 | Meerut
Percentage: 94%

SKILLS

- **Languages:** C, C++, Python, R, HQL, HTML, CSS
- **Databases:** MySQL, MongoDB, Redis
- **Developer Tools:** Power BI, Tableau, VS Code, GitHub, Jupyter Notebook, Hadoop, Linux
- **Soft Skills:** Problem Solving, Project Management, Interpersonal communication, Team work and leadership
- **Coursework:** Data Structures and Algorithms, Object Oriented Programming, Operating Systems, CN, DMBS

PROFESSIONAL EXPERIENCE

Summer Intern, *S&P Global Market Intelligence* 2023/05 – 2023/08 | Noida, India
- Developed calculation rules and conducted error checks for data transformation department.
- Utilized Foresee AI to extract and analyze unstructured data from PDF and HTML formats.

PROJECTS

MedZee, *For healthy lives* 2021/03
- Developed a website which aimed at collecting and donating household medicines, facilitating the disposal of unused medications.
- Optimized user experience on website, implementing intuitive features, encouraging active participation in the donation initiative.

Tech Stack: **HTML, CSS, JS, Bootstrap**

The Marshal App, *Three in one app* 2022/08 – 2022/12
- Developed a mobile application using Flutter SDK, implemented features to improve user productivity and organization
- The three sub-applications in this application include daily reminders, money management, and password management

Tech stack: **Flutter Framework, Dart Programming Language, Android Studio**

Healthcare Analytics, *For predicting LOS* 2023/01 – 2023/02
- Developed models to accurately forecast the Length of Stay for patients, enabling hospitals to optimize resource allocation
- Contributed to the healthcare system optimization by leveraging data analysis techniques to provide accurate predictions

Tech stack: **Python, Pandas, NumPy, Matplotlib, Keras, TensorFlow, Jupyter Notebook**

Mr Trash Wheel, *Waste classifier* 2023/01 – 2023/05
- Created an intuitive waste classification web application that employs CNN models to classify waste items into 6 categories.
- Streamlined waste management processes and promoted waste sorting practices and efficient waste management strategies.

Tech stack: **HTML, CSS, Python, CNN, TensorFlow, Flask**

ACHIEVEMENTS

Hack the Solution
Participated in Hack The Solution, 36 hours virtual hackathon held from March 27 to 28, 2021.

Write-O-Tech (TechWeek 2023)
Participated in Write-O-Tech organized by NSUT, showcasing proficiency in technical writing

POSITIONS OF RESPONSIBILITY

IEEE Student Branch JIIT, *Volunteer* 2021/09 – 2022/03 | Noida
- Contributed to the organization and execution of various events
- Managed communication with guest speakers and attendees

Eloquence, *Head* 2021/09 – present | Noida
- Organized various literary events and served as an anchor
- Mentored and guided junior members

INTERESTS

Volunteering | Community Involvement | Listening to music | Writing | Travel

Abhay Arya
Enroll.: 9920103097
BTech CSE
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY , NOIDA

+91-8506066546
abhay.arya.official@gmail.com

Education

•JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY *2020-2024*
BTech CSE CGPA: 7.1

•ST.THOMAS SCHOOL , Indirapuram, Ghaziabad *2020*
Central Board of Secondary Education, Uttar Pradesh Percentage: 88.6

•ST.THOMAS SCHOOL , Indirapuram, Ghaziabad *2018*
Central Board of Secondary Education, Uttar Pradesh Percentage: 93.4

Personal Projects

•Voice Based E-Mail

Aims to develop an application that allows blind and visually impaired individuals to send and receive emails using their voice.

- Tools & technologies used: Django , Python , HTML , CSS , JavaScript
- The application will have a user-friendly interface that can be easily navigated using voice commands. Users will be able to dictate their email content and have the application convert their spoken words into text. The application will then read the contents of the email back to the user, allowing them to review and make any necessary edits before sending the message.

•Image Captioning Bot

Developing an image captioning system to automatically generate relevant and descriptive captions for a given image.

- Tools & technologies used: Machine learning algorithms, CNN Models (VGG16, ResNet, and Inception), Python
- Users upload images via a user-friendly web interface, triggering feature extraction with the selected pre-trained CNN model. The bot generates descriptive captions, creating an engaging and accurate image captioning experience. This project involves implementing and analyzing three prominent image analysis models - VGG16, ResNet, and Inception - for the task of image feature extraction .

•Ride Sharing App Using Blockchain

The aim is to create a decentralized, transparent, and efficient ride-sharing platform using blockchain technology.

- Tools & technologies used: Blockchain Platform, Smart Contracts, Cryptocurrency, Ganache
- The proposed solution for a blockchain ride-sharing application based on smart contracts is to create a decentralized platform that connects riders and drivers directly without the need for intermediaries. The platform will be built on blockchain technology, which provides security, transparency, and immutability to the system. The use of blockchain technology will ensure that the transaction records are tamper-proof and transparent.

Technical Skills and Interests

Languages: C , C++ , Python , HTML , CSS
Skills: Data Structures and Algorithms , OOPS , DevOPs
Soft Skills: Time Management, Adaptable , Problem Solving , Communication
Areas of Interest: Analytics, prediction , AI , Python

Work Experience

•Internship Opportunity at Centre Of Excellence (COE), JIIT

- Mental Health Analysis Machine Learning Project:
  This project will use machine learning to analyze data from electronic health records, social media, and wearable devices. The system will extract important information from the data using natural language processing, computer vision, and signal processing techniques. The models will be trained to predict and classify mental health conditions accurately. By tuning the system's parameters and validating its performance, we aim to enable early detection and personalized interventions, helping mental health professionals make better decisions for patient care.

Achievements AND Certifications

•Data Analytics Course Completed 6 month Data Analytics course from Ducat

•AI with Python Course Completed 6 month AI with Python course from Ducat

•C, C++ and SQL Course Completed 6 month C,C++and SQL course from Ducat

Samarth Gupta

+91-9971981168 | samarthgupta547@gmail.com | https://www.linkedin.com/in/samarth-gupta-ab36231b6

Education

Bachelor of Technology in Computer Science, JIIT, Noida — Noida, Uttar Pradesh
CGPA: 7.5 — *Sept. 2020 – May 2024*

Class 12th, Little Flowers Public Senior Secondary School — Shahdara, Delhi
Percentage: 90 — *2018 – 2019*

Experience

Data Science Intern — June 2023 – July,2023
EDUNET Foundation, IBM SkillsBuild Internship Program

- Analyzed a large dataset using Python to identify key trends and insights.
- Employed exploratory data analysis and visualization tools to extract meaningful insights.
- Demonstrated proficiency in data wrangling and cleaning techniques, ensuring the integrity and quality of the dataset for analysis purposes.

Data Analytics Intern
KPMG Data Analytics Consulting Virtual Internship

- Conducted a comprehensive assessment of data quality and completeness using Excel, ensuring the reliability and integrity.
- Utilized Excel to analyze customer demographics and attributes, identifying high-value customer segments.
- Designed visually appealing and informative presentations using Excel.

Projects

Performance Analysis of Various Machine Learning Algorithms in Heart Disease Prediction | *Python*

- Conducted a thorough investigation into the performance of multiple machine learning algorithms, including decision trees, logistic regression, support vector machines (SVM), and random forests, for heart disease prediction.
- Employed comprehensive evaluation criteria, such as model accuracy, precision, recall, and area under the receiver operating characteristic curve (AUC-ROC), to assess the effectiveness of each machine learning algorithm in accurately predicting heart disease.

Analysis of Sample Superstore Dataset | *Python, Pandas, Matplotlib*

- Led end-to-end analysis of a Superstore dataset, encompassing data preprocessing, exploratory data analysis (EDA), and data visualization techniques.
- Evaluated the performance of different product categories, regions, and customer segments within the Superstore dataset to assess their contribution to overall revenue and profitability.
- Utilized visualization tools to identify trends, correlations, and outliers within the dataset, providing valuable insights into sales performance, customer behavior, and product categories.

Technical Skills

Languages: Python, C/C++, SQL (MySQL)
Coursework: Data Structures and Algorithms, Object Oriented Programming, Database Management Systems
Developer Tools: VS Code, Visual Studio, Jupyter Notebook, Google Colab
Libraries: pandas, NumPy, Matplotlib

Achievement/Certifications

- HackerRank SQL(Intermediate) Skill Assessment
- The SQL Programming Essentials Immersive Training by Udemy
- IBM SkillsBuild Data Fundamentals

9920103100_Image Captioning Bot

ORIGINALITY REPORT

12%	10%	3%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	2%
2	www.slideshare.net Internet Source	1%
3	github.com Internet Source	1%
4	towardsdatascience.com Internet Source	1%
5	vdocuments.mx Internet Source	1%
6	Submitted to Indian Institute of Technology Guwahati Student Paper	<1%
7	www.researchsquare.com Internet Source	<1%
8	www.erpublications.com Internet Source	<1%
9	Submitted to Fresno Pacific University Student Paper	<1%

10	Submitted to University of Wales Institute, Cardiff Student Paper	<1 %
11	www.ajist.org Internet Source	<1 %
12	fastercapital.com Internet Source	<1 %
13	Tarun Jaiswal, Manju Pandey, Priyanka Tripathi. "Advancing image captioning with V16HP1365 encoder and dual self-attention network", Multimedia Tools and Applications, 2024 Publication	<1 %
14	assets.researchsquare.com Internet Source	<1 %
15	Submitted to University of Hertfordshire Student Paper	<1 %
16	papers.ssrn.com Internet Source	<1 %
17	Submitted to Engineers Australia Student Paper	<1 %
18	www.mdpi.com Internet Source	<1 %
19	www.usebraintrust.com Internet Source	<1 %

20	Submitted to The University of the West of Scotland Student Paper	<1 %
21	docplayer.net Internet Source	<1 %
22	goo.by Internet Source	<1 %
23	Submitted to NOVA School of Business and Economics Student Paper	<1 %
24	www.hindawi.com Internet Source	<1 %
25	Submitted to Queen Mary and Westfield College Student Paper	<1 %
26	de.slideshare.net Internet Source	<1 %
27	Konstantinos Demertzis, Konstantinos Rantos, Lykourgos Magafas, Lazaros Iliadis. "A Cross-Modal Dynamic Attention Neural Architecture to Detect Anomalies in Data Streams from Smart Communication Environments", Applied Sciences, 2023 Publication	<1 %
28	Submitted to UOW Malaysia KDU University College Sdn. Bhd	<1 %

Student Paper

29	Submitted to University of East London Student Paper	<1%
30	eurchembull.com Internet Source	<1%
31	www.ijraset.com Internet Source	<1%
32	Submitted to Liverpool John Moores University Student Paper	<1%
33	Submitted to University of Lincoln Student Paper	<1%
34	iq.opengenus.org Internet Source	<1%
35	Anirudh B Mitta, Ajay H Hegde, Beluvigi Shreegagana, Mohammad Rooh Ullah, Asha Rani K P, Gowrishankar S. "Comparative Analysis on Machine Learning Models for Image Captions Generation", 2023 4th International Conference on Smart Electronics and Communication (ICOSEC), 2023 Publication	<1%
36	Submitted to Staffordshire University Student Paper	<1%

Exclude quotes	On	Exclude matches	< 14 words
Exclude bibliography	On		