Tribhuvan University

Institute of Science and Technology

Final Year Project Report On

"IMAGE CAPTIONING SYSTEM FOR SANSKRIT SCRIPT"

(Subject code: CSC 412)

In partial fulfillment of the requirement for the Bachelor's Degree in Computer Science and Information Technology (B.Sc. CSIT)

Under the supervision of:

Ramesh Singh Saud

Department of Computer Science and Information Technology

Bagbazar, Kathmandu

Submitted by:

Iliya Fathma (25787/077)

Smriti K.C (25808/077)

Submitted to:

Padmakanya Multiple Campus

Department of Computer Science and Information Technology

Bagbazar, Kathmandu

April 2025

Padmakanya Multiple Campus

Bagbazaar, Kathmandu

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by **ILIYA FATHMA** and **SMRITI K.C** entitled "**Image Captioning System using ViT and LSTM for Sanskrit Script**" in partial fulfillment of the requirement for the degree of Bachelor of Computer Science and Information Technology and proceed for the final evaluation.

…….

Mr. Ramesh Singh Saud

Supervisor

Department of Computer Science and Information Technology

Padmakanya Multiple Campus

Bagbazar, Kathmandu

LETTER OF APPROVAL

This is to clarify that the project proposed by Miss Iliya Fathma and Miss Smriti K.C entitled Image Captioning System Using Vit and LSTM, in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In my opinion, it is satisfactory in scope and quality as a project for the required degree.

Evaluation Committee

……	……
Mr. Ramesh Singh Saud	**Mrs. Sunita Shrestha**
Project Supervisor	Coordinator
Padmakanya Multiple Campus	Bsc.CSIT

……	……
Internal Examiner	**External Examiner**

ACKNOWLEDGEMENT

While gathering this project requirement, we went through numerous difficulties and rules and are spurred to make it effective, so want to offer our warm thanks to every one of them who helped us in each potential way. To begin with, we want to snatch this chance to offer our genuine thanks to the Department of Science and Technology, Tribhuvan University for making an astounding course as a piece of educational plan in B.Sc. CSIT.
It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highly respected **Mr. Ramesh Singh Saud** and **Mr. Mohan Bhandari** for investing their valuable time in guiding for completing this work. We will always be in debt for their kindness and help. We are also thankful to our faculty coordinator, **Mrs. Sunita Shrestha** for showing us the right direction and coordinating with us to start the journey of ours.

We would also like to show our gratitude to all those who have either straightforwardly or indirectly guided us in writing this report. The support and criticism were constantly valued. Finally, we would likewise want to thank our companion for encouraging us and giving positive feedback for our effort.

Members:

Iliya Fathma

Smriti K.C

ABSTRACT

This project discusses the development of an Image Captioning System to describe images descriptively through deep learning concepts. The paper describes a concept for an integrated model that adopts a Vision Transformer (ViT) for visual feature extraction that uses self-attention to encapsulate complex relations between spatial components and an LSTM network for capturing sequential dependencies on the generated words. The approach learns from mapping visual content and textual descriptions and is trained in large-scale paired datasets. The generated captions are accurate, contextually relevant, and provide a natural language description of the given image. Applications include image accessibility, content analysis, and auto-generated content, which further closes the gap between computer vision and natural language processing.

Keywords: *Image Captioning, Vision Transformer (ViT), LSTM, Deep Learning.*

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BLEU	BiLingual Evaluation Understudy
CNN	Convolution Neural Network
DL	Deep Learning
LSTM	Long Short Term Memory
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
RNN	Recurrent Neural Network
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
ViT	Vision Transformer

CHAPTER 1: INTRODUCTION

1.1 Introduction

Image captioning is a crucial task in artificial intelligence that involves generating descriptive textual interpretations of visual data. This process combines the strengths of computer vision and (NLP) to analyze and communicate visual content effectively [1]. Image captioning has a wide range of practical applications, including assisting visually impaired individuals by providing real-time descriptions of their surroundings, enhancing content discovery on social media platforms by enabling better image indexing, and improving automated content moderation systems by facilitating contextual understanding of visual data. Furthermore, it plays an essential role in human-computer interaction, medical imaging, autonomous navigation, and multimedia retrieval systems, making it an active research area in artificial intelligence.

The primary goal of an image captioning system is to generate a contextually rich and semantically accurate description of an image. This requires not only object recognition but also an understanding of object relationships, spatial configurations, and contextual nuances. Traditional image captioning models relied on (CNNs) for feature extraction and (RNNs) such as (LSTM) networks for text generation. However, recent advancements in deep learning have introduced (ViTs) as powerful alternatives to CNNs, significantly improving the ability of image captioning systems to capture complex dependencies in images.

(ViTs) revolutionize feature extraction by utilizing a self-attention mechanism, which processes an image as a sequence of non-overlapping patches instead of analyzing pixels through convolutional layers. This approach allows ViTs to capture long-range dependencies and detailed spatial relationships across the entire image, leading to a more comprehensive and structured representation of visual information. Unlike CNNs, which primarily focus on local features, ViTs enable a global perspective on the image, improving the system's ability to generate captions that accurately reflect the interactions and contexts within the visual scene.

Once the ViT extracts high-dimensional feature representations from the image, the LSTM component of the system takes over as the decoder, generating a descriptive caption in a sequential manner, word by word. LSTMs are specifically designed to handle sequential data, making them well-suited for language modeling and text generation. They retain contextual information from previously generated words while predicting the next word in the sequence, ensuring that the captions produced are grammatically coherent and semantically meaningful.

In the proposed system, the ViT serves as the encoder, converting raw image data into structured feature representations, while the LSTM functions as the decoder, sequentially transforming these features into textual descriptions. By leveraging the complementary strengths of ViTs for vision processing and LSTMs for language modeling, the system aims to produce highly accurate, contextually relevant, and diverse captions. This hybrid approach enhances the robustness of automatic visual description systems, paving the way for more sophisticated and adaptable image-captioning models that can generalize across diverse image domains.

1.2 Problem Statement

The task of accurately generating descriptive captions for images remains a challenging problem in artificial intelligence, primarily due to the complexities involved in interpreting visual data and converting it into coherent, human-like text. Traditional image captioning systems, which typically rely on CNNs for feature extraction and RNNs, such as LSTMs, for text generation, have made significant progress in this field. However, they suffer from several key limitations. Sanskrit, though one of the oldest and highly linguistically enriched languages, still has very little representation in today's AI-driven applications, majorly in an image captioning system. Much development has taken place in providing image captions with English; however, there is a serious deficiency in models which are capable of generating descriptive captions in Sanskrit.

An AI-based image captioning is proposed to bridge the gap, which may be trained over the Flickr8k dataset using Sanskrit captions. This provides a state-of-the-art outcome by using the pre-trained encoder of the Vision Transformer with an LSTM-based decoder in describing images using Sanskrit captions effectively.

Image captioning in Sanskrit faces difficulties due to issues such as dataset diversity, grammatical complexity, and translation inconsistencies. In developing and optimizing this model, we contribute to Sanskrit language processing, making the language more accessible

1.3 Objectives

1. To generate a Sanskrit dataset for image captioning using the Flickr dataset.
2. To develop a Sanskrit image captioning model utilizing Vision Transformer (ViT) and LSTM networks to generate coherent and sequential captions.
3. To improve the accuracy and richness of generated captions by capturing fine-grained details, object relationships, and contextual information in images.
4. To develop a user-friendly interface that allows users to upload images and receive generated captions in real time, making the system accessible to non-technical users.

1.4 Scope

This project centers on automated image captioning in Sanskrit, utilizing the Flickr8k dataset. The system employs a pretrained Vision Transformer (ViT) encoder along with an LSTM-based decoder to create Sanskrit descriptions for images. Given the dataset's variety of everyday scenes, the system has applications across several domains, including:

1. Sanskrit Language Processing and AI Research: It contributes to Natural Language Processing (NLP) by enhancing Sanskrit text generation. It serves as a benchmark for training AI models in low-resource languages like Sanskrit.

2. Image Captioning for Sanskrit Learning: This system aids educational platforms by producing Sanskrit descriptions for images, making the learning process more engaging. It can be integrated into Sanskrit language learning applications to assist with vocabulary development and sentence construction exercises.

3. Multilingual AI and Translation Support: The system is beneficial for machine translation by offering Sanskrit captions for general images, which can facilitate parallel text generation for Sanskrit-English translation models. It can enhance multilingual search engines, enabling users to find images using Sanskrit phrases.

4. Digital Humanities and Cultural Preservation: It plays a role in digitizing Sanskrit texts for image-based archives. This can be utilized in libraries and digital repositories to provide Sanskrit descriptions for a variety of images.

5. Assistive Technology for Visually Impaired Sanskrit Learners: The system can be incorporated into text-to-speech technologies to assist visually impaired users in hearing Sanskrit descriptions of images. It offers accessibility features for Sanskrit readers who depend on audio-based learning.

6. Social Media and Content Creation: This technology can automatically generate Sanskrit captions for images shared on social media platforms.

1.5 Limitations

While this project is a step forward in Sanskrit-based AI, it does come with a few challenges:

1. Limited Dataset Diversity: The model is trained on the Flickr8k dataset, which mainly contains everyday scenes (people, animals, objects, and outdoor activities). As a result it might struggle with complex or specialized images (e.g., medical scans, satellite images, or abstract art). The captions may sometimes be too generic, missing finer details.

2. Sanskrit Language Complexity: Sanskrit has a rich grammatical structure with complex word formations (Sandhi) and inflections. Because of this the model might not always generate grammatically perfect sentence, some Sanskrit case endings (Vibhaktis) or verb forms may be incorrect or inconsistent.

3. Lack of Contextual Understanding: Since the model only looks at images and doesn’t understand deeper context it might misinterpret ambiguous images (e.g., mistaking a "boy playing with a dog" for "a dog chasing a boy"). The captions may miss emotions or abstract meanings that a human would easily infer.

4. Data Bias and Translation Errors: The original Flickr8k dataset was in English, and the captions were translated into Sanskrit. Any translation errors or loss of meaning might affect the quality of generated captions. The model learns from existing data, so it might favor certain sentence structures over others, making its responses feel repetitive.

5. Computational Requirements: Running the model efficiently requires a GPU for faster processing. On lower-end devices, caption generation might be slow. Deploying the model for large-scale use would require optimization techniques to improve speed and accuracy.

1.6 Development Methodology

The Sanskrit Image Captioning System follows a structured Waterfall Model, progressing from problem identification to deployment. The process begins with requirement analysis, identifying the need for generating Sanskrit captions. System design structures the model architecture, where ViT extracts image features, and LSTM generates captions.

Data preprocessing ensures efficient training, and the user interface enables image uploads with real-time Sanskrit captions. During implementation, the model is trained, saved, and integrated with Gradio for interactive caption generation. The system is then deployed on Kaggle and local environments for user interaction. Continuous improvements refine model predictions based on feedback. Future enhancements include replacing LSTM with Transformers, expanding the dataset, and integrating text-to-speech for audio captions. This methodology ensures efficiency, scalability, and user-friendly image captioning in Sanskrit.

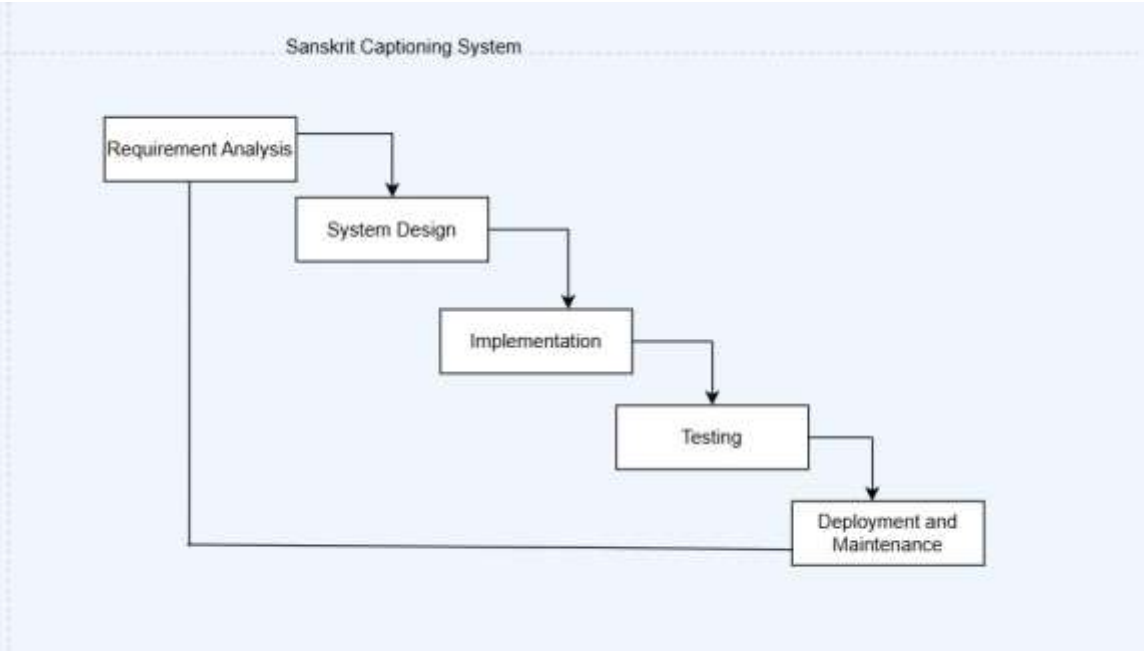


Figure 1: Development Methodology of Image Captioning System

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Image captioning is an important field of research that combines computer vision with natural language processing (NLP) to generate coherent text descriptions of visual information. The evolution of deep learning approaches, such as Vision Transformers (ViT) for feature extraction and Long Short-Term Memory (LSTM) networks for sequence generation, has improved the effectiveness of captioning systems significantly. Compared to the plethora of large datasets and advanced models available for English and other leading languages, the field of Sanskrit image captioning is less explored owing to its morphological complexity and lack of annotated datasets. To fill this gap, this review explores the recent trends in image captioning using the Flickr 8K dataset, outlining the role of ViT as an encoder and LSTM as a decoder, while also considering the challenges involved in applying these architectures to Sanskrit.

2.1.1 Image Captioning Techniques

1. Traditional CNN-RNN Based Models

Early approaches to image captioning used Convolutional Neural Networks (CNNs) to extract features from images and Recurrent Neural Networks (RNNs), such as LSTMs, to generate sequence outputs. Although effective, these approaches suffered from limitations in handling long-range dependencies and global contextual information. [2] proposed a CNN-LSTM model that utilized both the Flickr 8K dataset and a self-constructed dataset, achieving improvements in semantic relevance. [3] reported that CNN–LSTM models work well with structured datasets, but struggle with complex real-life images.

2. Transformer-Based Image Captioning

Vision Transformers (ViT) have revolutionized feature extraction methods by using self-attention mechanisms, unlike the use of local receptive fields commonly used in CNNs. ViTs have higher efficacy in extracting global context from images, leading to improved captioning performance. [4] proposed a ViT-LSTM model pre-trained on ImageNet 21k and fine-tuned on Flickr 8K. The results showed that ViTs outperformed CNNs in feature

extraction for captioning. [5] compared ViT-LSTM and CNN-Transformer models, showing that ViT-based models achieve better BLEU scores when tested using the Flickr 8K dataset.

3. Hybrid ViT-LSTM Architectures

A common approach in modern image captioning is the combination of ViT for feature extraction and LSTM for text generation, thus balancing efficiency and contextual understanding. [6]showed that ViT-LSTM models can minimize computational expenses while maintaining high accuracy, making them suitable for low-resource languages like Sanskrit. [7] compared VGG16-LSTM with ViT-LSTM models, showing that ViT-LSTM had better generalization across datasets.

2.1.2 Flickr 8K Dataset for Image Captioning

1. Overview

The Flickr 8K dataset contains 8,000 images, each with five descriptive captions, making it a well-known benchmark for training image captioning models.

2. Performance on Flickr 8K

Several studies have evaluated different architectures using the Flickr 8K dataset. [8] found that ViT-GPT-2 performed better than ViT-LSTM, though LSTM was effective under constrained computational resources. [9] showed that ViT-LSTM models improved BLEU scores by 12% compared to CNN-LSTM.

2.1.3 Challenges in Sanskrit Image Captioning

1. Morphological Complexity

Sanskrit is a highly inflectional language with rich grammatical structures, making NLP tasks challenging. Unlike English, where word order is rigid, Sanskrit allows flexible word positioning, posing challenges for LSTM-based decoding.

2. Limited Annotated Datasets

Unlike English, large-scale Sanskrit image captioning datasets are scarce. Current models rely on transfer learning from English models and manually curated Sanskrit corpora for fine-tuning.

3. Lack of Pre-trained Sanskrit Models

There are limited transformer-based models for Sanskrit, making it difficult to adapt existing ViT-LSTM architectures for Sanskrit captioning.

2.2 Conclusion

This review highlights the advancements in image captioning techniques, emphasizing ViT-LSTM architectures and their applications in Flickr 8K dataset-based captioning. While these models have shown significant improvements over traditional CNN-RNN approaches, challenges remain in low-resource languages like Sanskrit, primarily due to morphological complexity and dataset limitations. Future research should focus on developing large-scale annotated Sanskrit datasets and improving pre-trained transformer models for Sanskrit-based captioning tasks.

CHAPTER 3: SYSTEM ANALYSIS

3.1 Methodology

The methodology of the Sanskrit Captioning System is based on a structured approach: dataset selection and preprocessing, model training, and finally, model evaluation. Utilize the Flickr8k dataset for image-caption pairs, and pre-process to extract features with meaningful information. A Vision Transformer ViT encoder for visual capture and an LSTM decoder for generating Sanskrit captions are used. A model assured this way will generate Sanskrit image captions effectively. The methodology for the Sanskrit Captioning System consists of the following key stages, as depicted in the diagram:

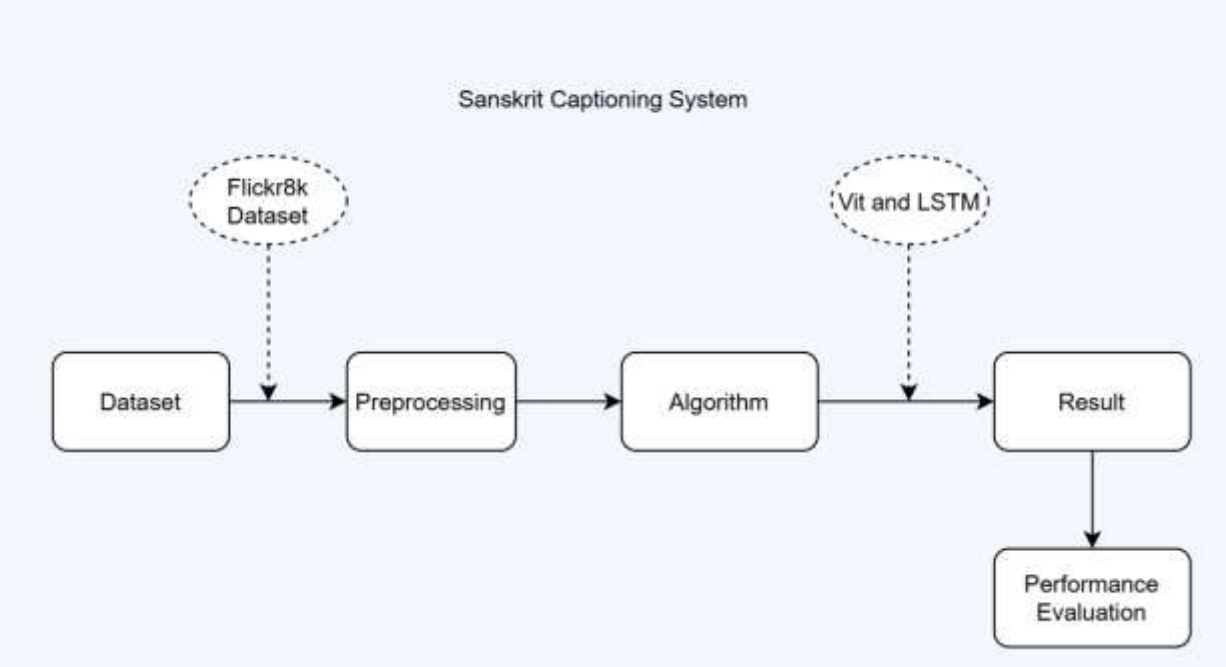


Figure 2: Methodology of Image Captioning System

3.1.1 Dataset Selection

The dataset selection process involves choosing a suitable dataset that provides sufficient image-caption pairs for training the model. In this system, the Flickr8k dataset is used, which consists of 8,000 images, each with multiple descriptive captions. [10] This dataset serves as the foundation for training the image captioning model, enabling it to learn meaningful image-to-text associations. The dataset is carefully examined to ensure that the captions align well with the images and provide adequate context for generating accurate descriptions.

3.1.2 Preprocessing

The next step is preprocessing, where both images and textual captions undergo necessary transformations before being fed into the model. Image preprocessing includes resizing, normalization, and feature extraction to make them compatible with the Vision Transformer (ViT). Meanwhile, text preprocessing involves tokenization, vocabulary building, and encoding captions into numerical sequences that the model can process. Special tokens such as <start> and <end> are added to the captions to indicate the beginning and end of sentences, ensuring proper structuring during training.

3.1.3 Algorithm

Following preprocessing, algorithm is applied , where a Vision Transformer (ViT) encoder extracts high-level visual features from input images. These extracted features are then passed to a Long Short-Term Memory (LSTM) decoder, which sequentially generates captions. The model is trained using a sequence-to-sequence learning approach, where it learns to associate visual features with relevant textual descriptions in Sanskrit. The training process involves optimizing the model's parameters using a loss function, ensuring that the generated captions closely match the actual captions.

3.1.4 Result Generation

Once the model is trained, the result generation phase begins. Given an input image, the model processes it through the ViT encoder to extract visual features, which are then decoded by the LSTM to generate a meaningful Sanskrit caption. The generated captions are compared with ground truth descriptions to verify their correctness. This step ensures that the system can successfully generate image captions that are both contextually relevant and grammatically accurate.

3.1.5 Performance Evaluation

Performance evaluation is conducted to assess the accuracy and effectiveness of the generated captions. Standard evaluation metrics like BLEU and Rough Score are utilized to quantify the model's performance numerically. These metrics are used to compare training epochs, analyze the learning trend, and assess the model's convergence. Evaluation helps identify areas where the model needs improvement and aids in refining it for enhanced caption generation. Through this step-by-step process, the Sanskrit Captioning System ensures high-quality and descriptive image explanations.

3.2 System Analysis

Before designing the system, there has to be a profound examination of the users' specification, feasibility study, and risk analysis before the system design. Understanding the user requirements makes it possible to be sure the system will be for its purpose and meet the expectations of the users accordingly. Feasibility studies allow checking on the workability of the project, considering the technical limitations and if the proposed solution is viable. Moreover, it allows analyzing the associated risks well in advance of potential challenges in development. These steps will help in the effective allocation of resources according to priority, so that more important aspects are given due consideration. This structured approach helps in informed decisions about where exactly the efforts should be concentrated to optimize time and resources while overcoming potential obstacles. Ultimately, a well-planned and prioritized development process enhances the overall effectiveness and success of the system.

3.3 Requirement Analysis

The process of requirement identification focuses on understanding the expectations and needs for a new image captioning system. This includes gathering both functional and non-functional requirements, ensuring they are well-defined, measurable, and achievable. It is crucial for the success of the project as it lays the foundation for system design and development. Requirements identification involves collaboration between different stakeholders and includes both hardware and software needs, as well as considerations for human interaction with the system.

3.3.1 Functional Requirements

Functional requirements define the specific tasks or actions the system must perform. For this image captioning project, the core functional requirements are:

- Upload Image: The user uploads an image to the system, which serves as the input for the image captioning process.
- Preprocess Image: The system processes the uploaded image to enhance quality, resize, normalize, or apply other necessary transformations to prepare it for feature extraction.
- Extract Features: The system extracts important visual features from the image, which will be used for generating descriptive captions.
- Generate Features: The extracted features are processed and transformed into meaningful representations that the captioning model can understand.
- Display Caption: The generated caption is displayed to the user, describing the content of the image in natural language.

Use Case Diagram:

The image captioning system use case diagram illustrates the interaction of the user and the system. It indicates how the user inputs an image into the system, which processes the image to obtain its visual features. The system uses the obtained features to generate a descriptive caption from the analyzed data. The system then displays the generated caption

to the user. Once the caption is read out, the user can give feedback on how relevant and accurate the caption is. The diagram correctly illustrates the activity flow between the user and the image captioning system and indicates the main processes that are conducted in image caption generation.

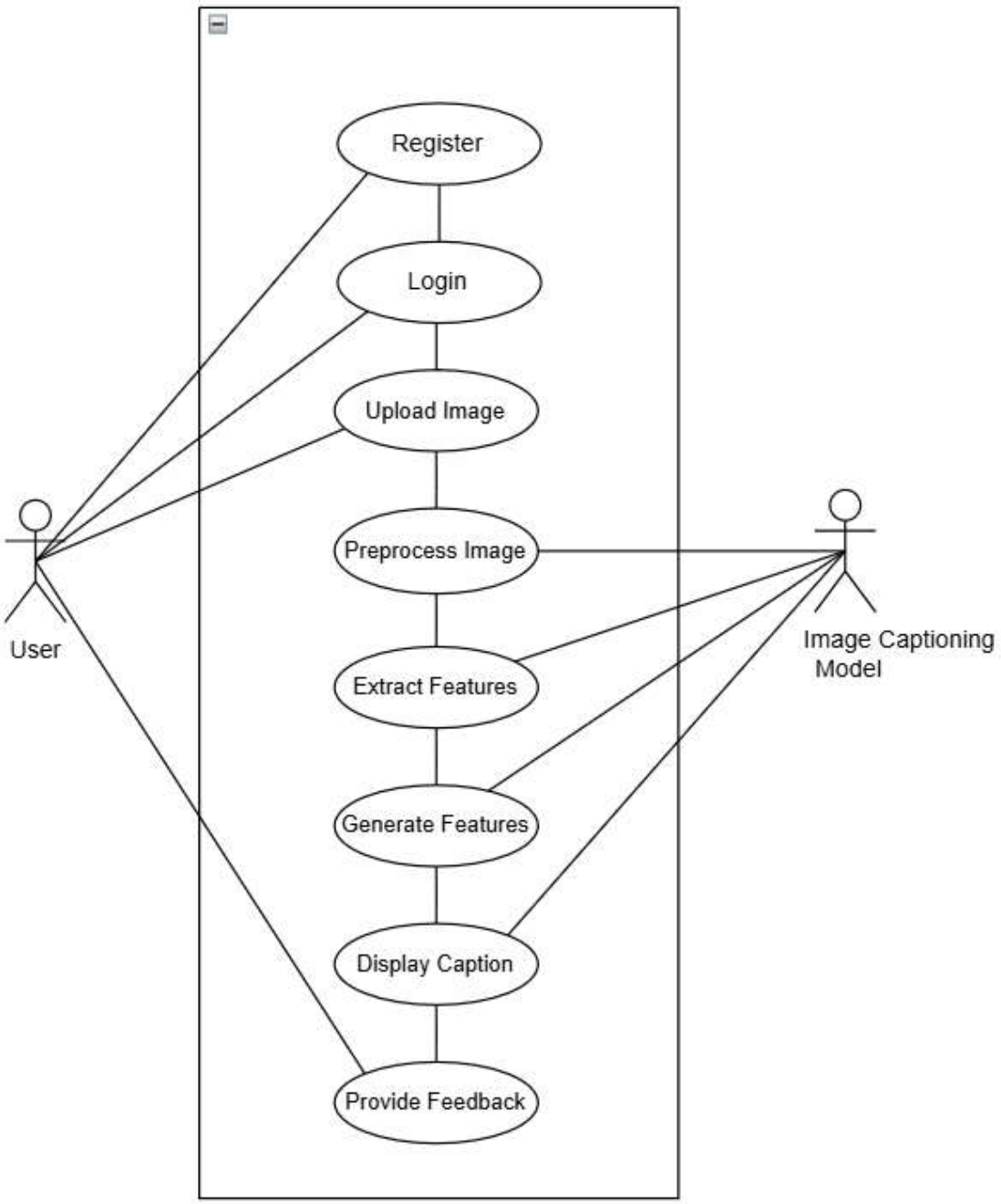


Figure 3: Use Case Diagram of Image Captioning System

3.3.2 Non-Functional Requirements

Non-functional requirements address the overall quality and performance characteristics of the system, ensuring it meets certain standards. For this project, the non-functional requirements include:

- Efficiency: The training process for generating captions must be fast and resource-efficient, handling large datasets without excessive computational delays.
- Scalability: The system should be scalable to accommodate increased data volumes and growing user demands without degrading performance.
- Reliability: The system should be reliable with minimal downtime, ensuring that caption generation is always available.
- Usability: The system should be user-friendly, enabling users of varying technical expertise to interact with the system effectively.

3.4 Feasibility Study

A feasibility study is an essential part of the planning phase for the image captioning system, determining whether the project is viable from technical, operational, and economic perspectives. Below is the feasibility analysis for the proposed system using Vision Transformers (ViT) and LSTM for image captioning:

3.4.1 Technical Feasibility

The technical feasibility of the image captioning system relies on the availability and maturity of deep learning models and frameworks. Modern libraries like PyTorch and TensorFlow, which support both ViT and LSTM, are widely used in the research community and industry for similar applications. The system can leverage pre-trained models and datasets, reducing the need for extensive computational resources and enabling rapid development and testing.

- Tools and Libraries: The project uses PyTorch, a highly flexible deep learning framework, along with supporting libraries such as transformers, OpenCV, and

NumPy. These are well-documented, widely used, and provide all the necessary functionality for image preprocessing, model building, and training.

- Model Availability: ViT and LSTM are well-established deep learning models for image processing and text generation, respectively. The integration of these models for image captioning is well-supported in academic literature and open-source implementations, which makes it technically feasible to build and optimize the system.
- Hardware Requirements: The system requires sufficient computational resources, such as GPUs for training deep learning models, but given that GPUs are increasingly available in cloud computing platforms and most research institutions, this is not a major barrier.

3.4.2 Schedule Feasibility

The system which we are going to develop will be completed within scheduled time and will not exceed the scheduled time. The expected time duration for completion of the project is drawn with the help of GANTT chart.

The GANTT chart showing the expected schedule is given as below:

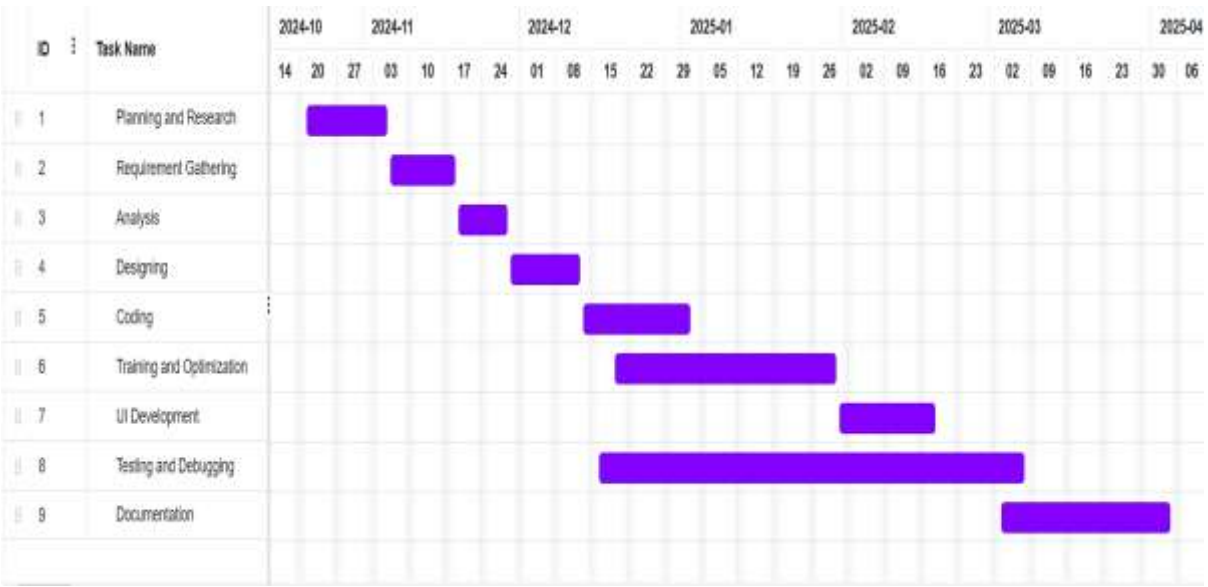


Figure 4: Gantt Chart

3.5 Analysis

A class diagram represents a system's classes, interfaces, and relationships. It shows the static structure of the system and models the classes, attributes, methods, and their relationships.

3.5.1 Object modelling using Class Diagram:

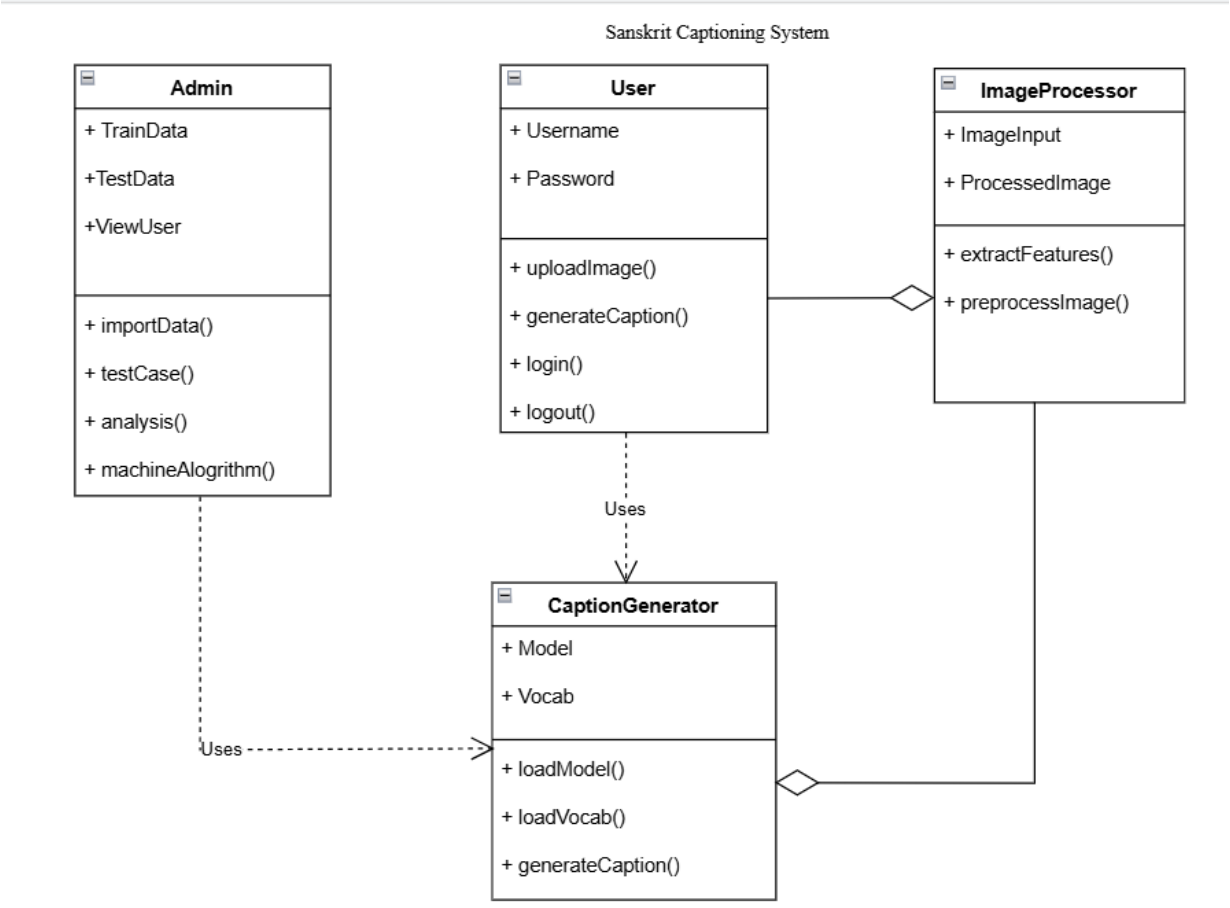


Figure 5: Class Diagram of Image Captioning System

The Sanskrit Captioning System class diagram illustrates the structure and behavior of the system. There are four principal classes in the system: Admin, User, ImageProcessor, and CaptionGenerator. The Admin class handles operations related to data, such as training, testing, and analysis, using methods such as importData(), testCase(), and analysis(). The

User class handles operations such as image upload, caption generation, login, and logout. The ImageProcessor class processes uploaded images by extracting and preprocessing features through methods such as extractFeatures() and preprocessImage(). The CaptionGenerator class generates text captions through methods such as loadModel(), loadVocab(), and generateCaption().

3.5.2 Dynamic modeling using State and Sequence Diagrams

State Diagram

The Sanskrit Captioning System flowchart demonstrates the order of operations from the idle mode to the creation and display of image captions. The operation begins in the idle mode, where the user starts the process by carrying out user authentication. If the credentials for login are correct, the user logs in and then uploads an image. If invalid credentials are provided, access is denied and the user is prompted to input the credentials again. On successful login, the user uploads the image, and the generation of the caption is initiated. The system shows the generated caption to the user once the caption has been generated. The system executes a test case for verifying the accuracy of the generated caption. The user logs out after performing the activities, and the system goes back to the idle state. The flowchart effectively shows the step-by-step generation of the captions in the Sanskrit Captioning System.

The state diagram below shows the process for Sanskrit image captioning system.

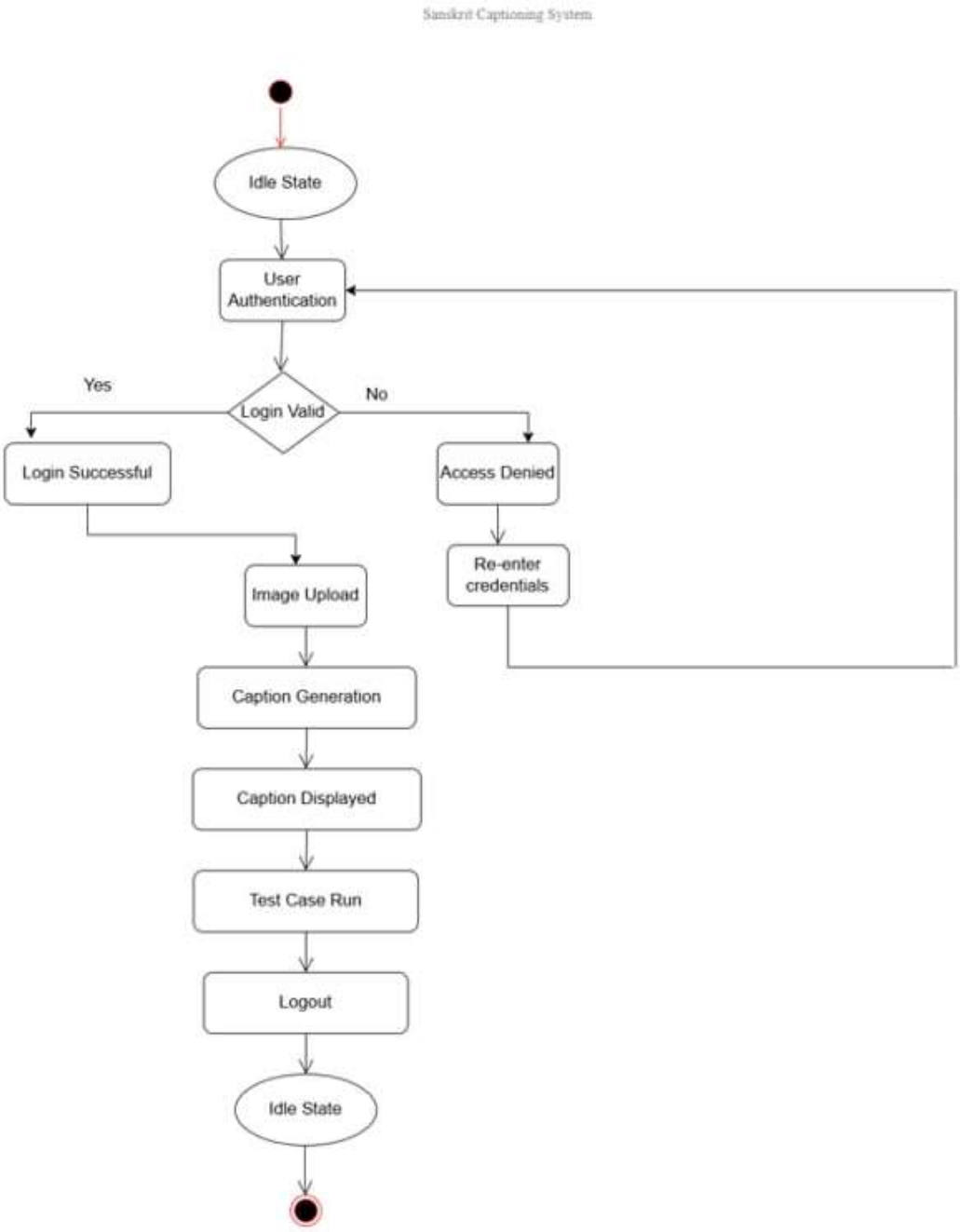


Figure 6: State Diagram of Image Captioning System

Sequence Diagram

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are show as arrows.

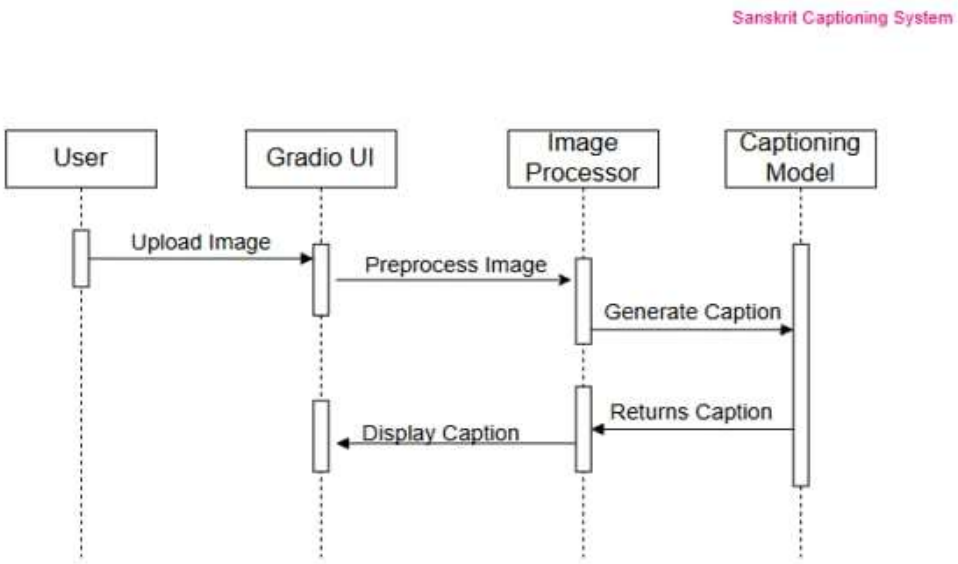


Figure 7: Sequence Diagram of Image Captioning System

The sequence diagram illustrates the flow of a Sanskrit Captioning System deployed on a Gradio UI. The flow starts with the user uploading an image through the Gradio UI, which sends the image to the image processor for pre-processing. Pre-processed, the image is sent to the captioning model, which generates a caption. The model sends the caption back to the image processor, which forwards it to the Gradio UI to be displayed. The user reads the generated caption, and image captioning is complete.

3.5.3 Process modeling using Activity Diagrams

Activity Diagram

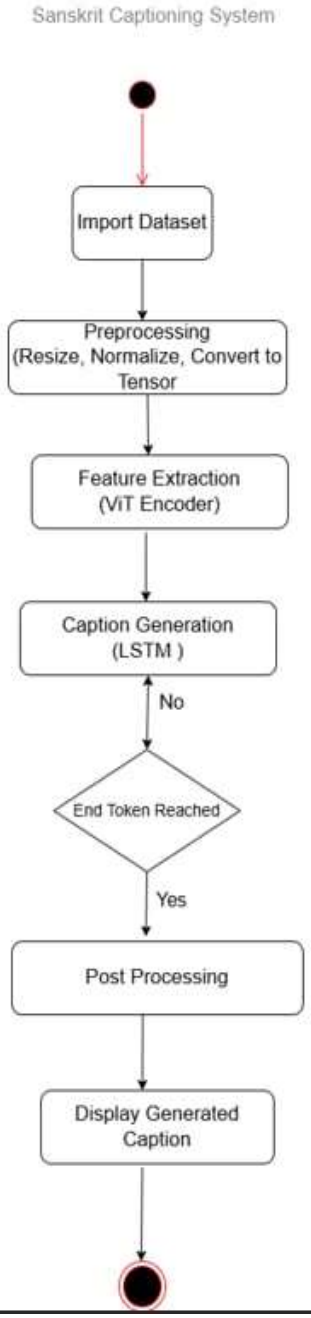


Figure 8: Activity Diagram of Image Captioning System

An Activity Diagram for an Image Captioning System represents the flow of operations involved in processing an image and generating a meaningful caption. It provides a step-by-step visualization of how an image moves through different stages, from user input to the final caption output.

The activity diagram illustrates the process of a Sanskrit Captioning System. The process starts with the importing of the dataset, then preprocessing, where resizing, normalization, and tensor conversion are applied to the images. Then, a Vision Transformer (ViT) encoder is used to extract features from images. The extracted features are fed to an LSTM-based model to generate captions. The process is executed iteratively up to the generation of the end token, which signifies the end of the caption. After being generated, the caption is sent for post-processing to be refined and then presented as the final output. This systematic process guarantees effective image-to-text conversion in Sanskrit.

CHAPTER 4: SYSTEM DESIGN

4.1 System Architecture

System design refers to the process of defining the architecture, components, and interfaces involved in a system. It is the application of system theory to develop a structured product. In the system design of an image captioning system, the process begins with image data collection, followed by preprocessing and feature extraction. The image is passed through a Vision Transformer (ViT) for feature extraction, and a linear projection layer adjusts the feature vector dimensions. The feature vector is then fed into the LSTM-based decoder, which generates a caption by sequentially predicting words. Finally, the system displays the generated caption to the user. The UML diagrams are refined to provide a detailed description of the system components, making it easier to understand the overall functionality. The refined diagrams include the class diagram, sequence diagram, and activity diagram of different system modules.

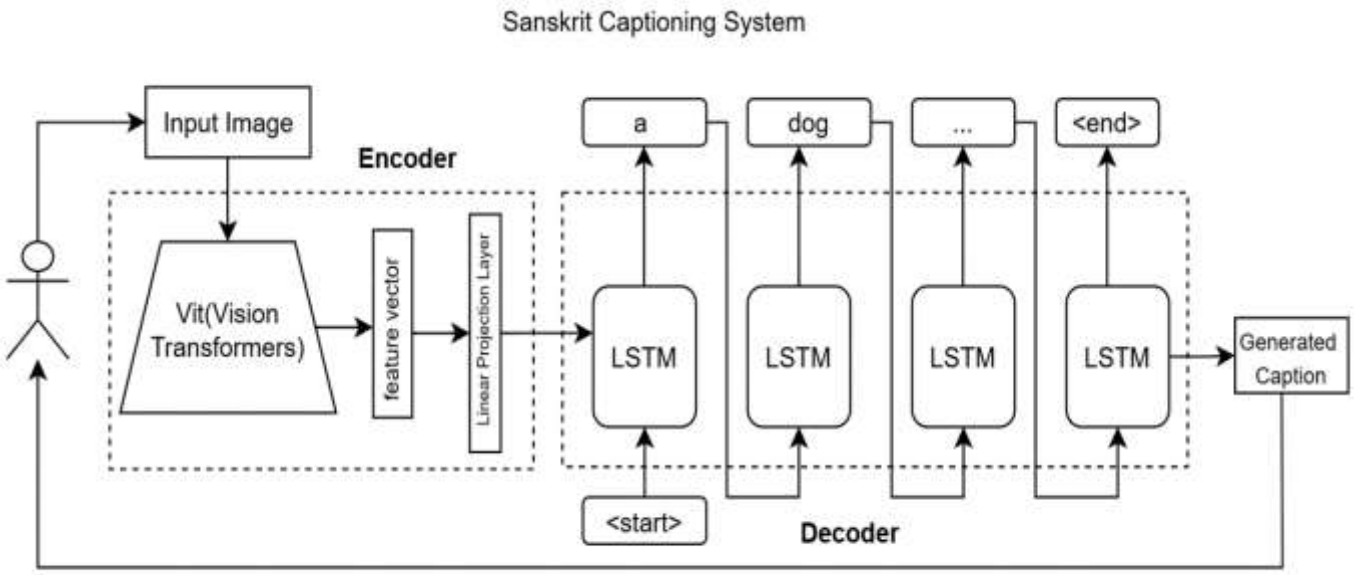


Figure 9: System Architecture of Image Captioning System

4.2 Refinement of Class Diagram

This refined class diagram represents the interaction of various components. The User class has authentication support (login/logout) and offers methods for uploading an image and generating a caption. The ImageProcessor class performs image preprocessing and feature vector extraction, as needed for generating a caption. The CaptionModel class loads the pre-trained model and vocabulary for creating meaningful captions from extracted features. Also, the Admin class is tasked with training and test data management, analysis, and application of machine learning models to improve captioning accuracy.

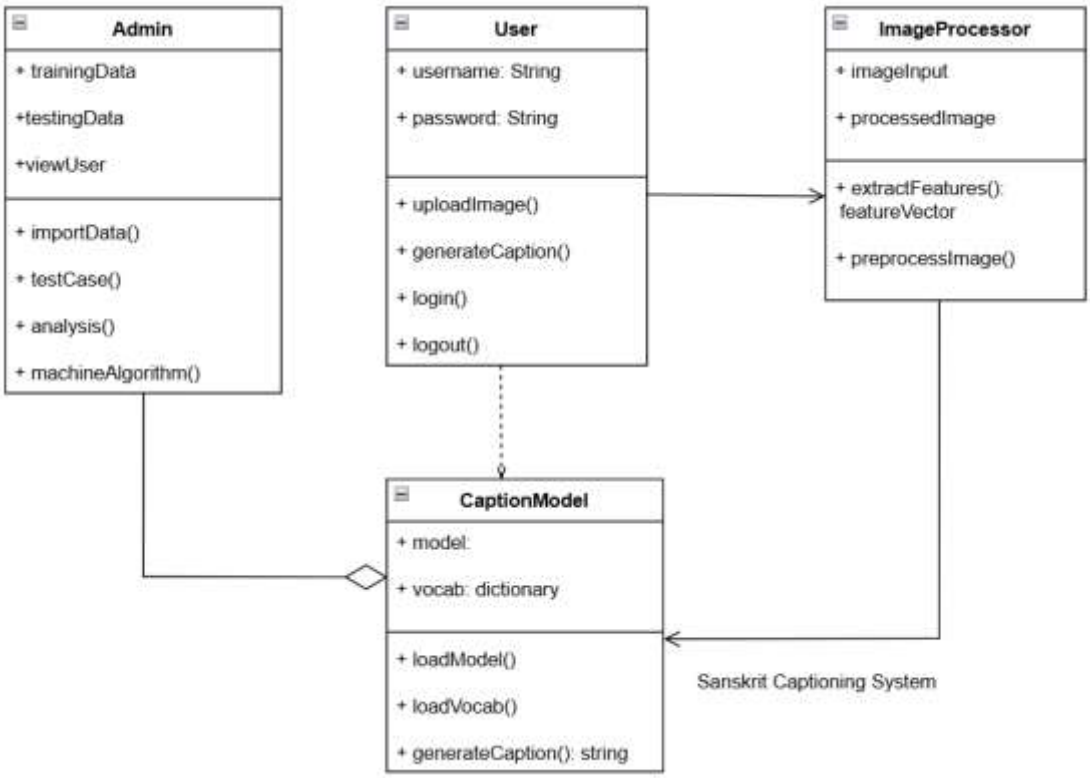


Figure 10: Refined Class Diagram of Image Captioning System

4.3 Refinement of Sequence Diagram

This is a step-by-step refined sequence diagram that shows the image caption generation process in the system and the invalid image format handling process. The user uploads an image to the System, which accepts the image, processes it, and then forwards the processed image to the Model for caption generation. The Model responds with a generated caption or raises an error if the image format is invalid. If the caption is successfully generated, the system shows it to the user; otherwise, it shows an error message. The alt (alternative) block in this improved sequence diagram makes sure to show an error if the type of image uploaded is invalid.

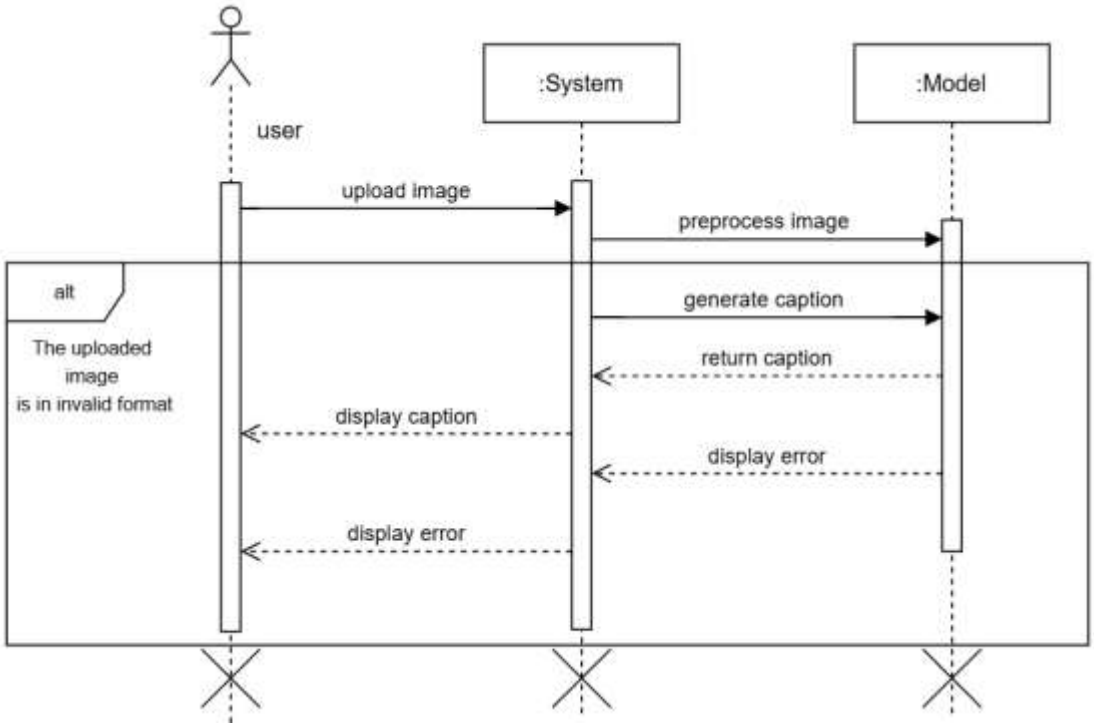


Figure 11: Refinement of Sequence Diagram of Image Captioning System

4.4 Component Diagram

The Sanskrit Captioning System component diagram illustrates the coordination of different system components to come up with Sanskrit captions for uploaded images. The User Interface is the portal through which users upload images and send login requests. When it receives a login request, the Image Processor interacts with the Security Module for authentication of user credentials from the Database, providing secure access. After authentication, the uploaded image is processed and forwarded to the Caption Generator that utilizes state-of-the-art models for Sanskrit caption generation. The caption generated is sent to the user via the User Interface. The system enables secure handling of data using access control and authentication mechanisms through the Security Module.

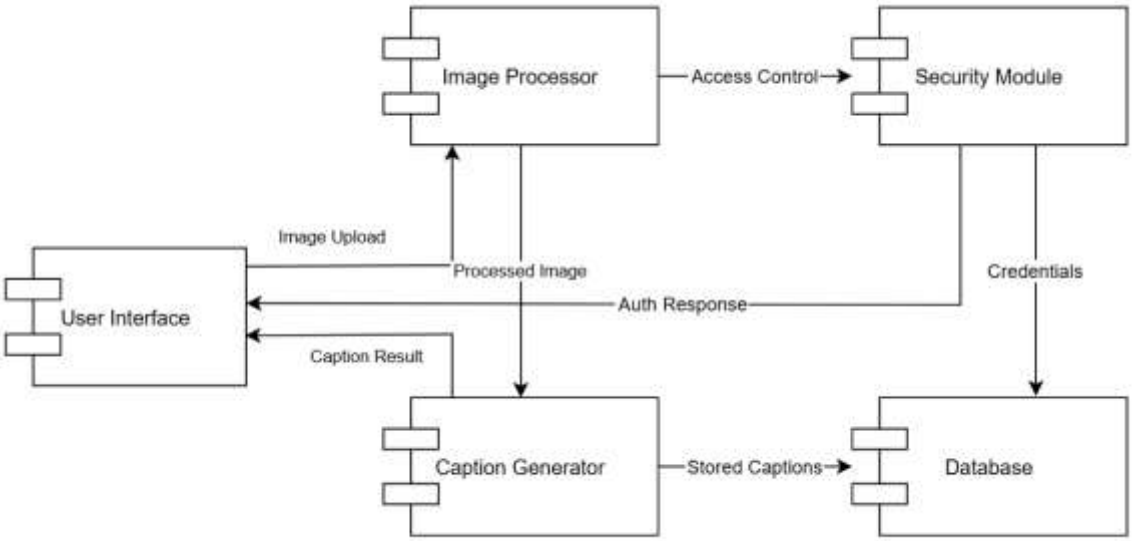


Figure 12: Component Diagram of Image Captioning System

4.5 Deployment Diagram

A deployment diagram depicts the physical architecture of the image captioning system. In other words, this diagram shows the way software components are distributed among hardware resources, including servers, devices, and cloud infrastructure. Nodes such as a server or a local machine may host the model, web interface, and database. Artifacts include the image captioning model, user interface files, and datasets. It explains communication paths between the model, UI, and how other components interact.

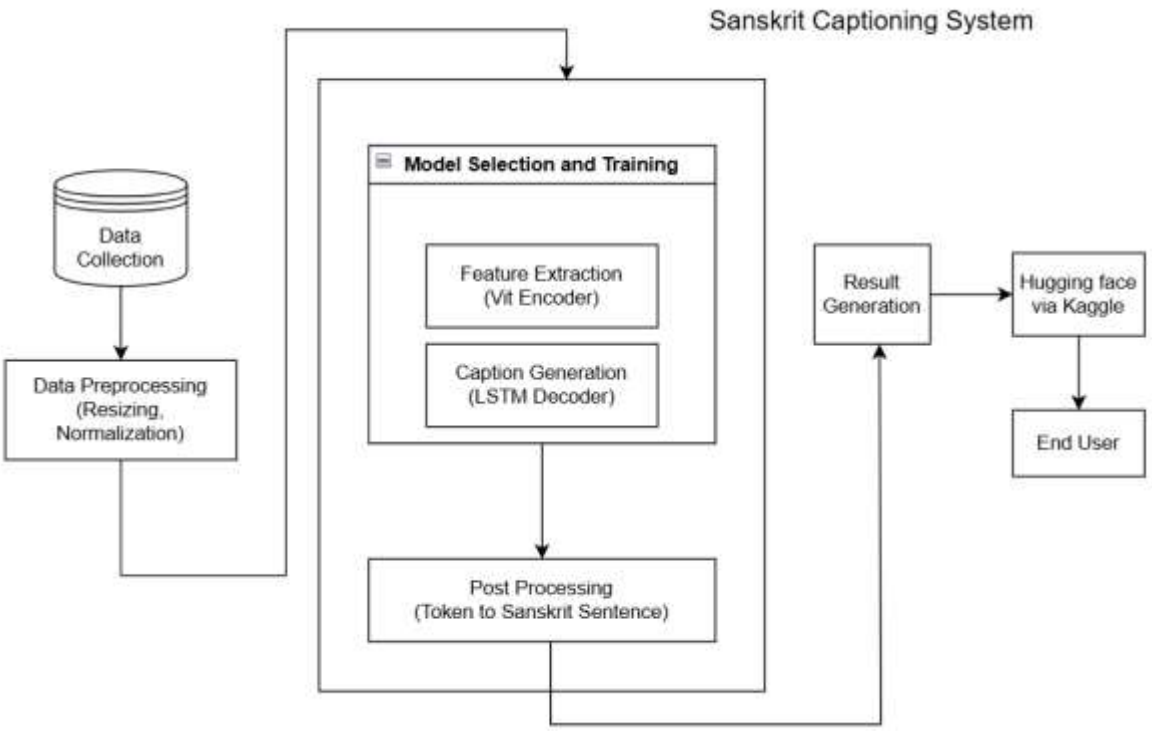


Figure 13: Deployment Diagram of Image Captioning System

4.6 Implementation Details of Algorithms

The proposed model for image captioning is implemented by combining a pre-trained Vision Transformer, ViT, as the feature extractor, and a Long Short-Term Memory Network, LSTM, as the generator of the captions. This section covers some critical stages of implementation: data pre-processing, feature extraction, training and caption generation and prediction.

1. Preprocessing Steps

The dataset for this project includes images with corresponding captions in the Sanskrit language. The preprocessing pipeline for such a dataset will involve the following:

Image Processing: Resizing and normalizing the images before feeding into the Vision Transformer.

Text Tokenization: Tokenize the captions by converting the words into numerical representations using a vocabulary file - vocab.json. Add padding and special tokens (<start>, <end>) to standardize the sequence lengths.

Label Encoding: The captions are first encoded as sequences of indices by using a word-to-index mapping.

START

1. Load dataset (images and their corresponding captions)

2. Preprocess images:

   a. Resize images to required size (e.g., 224x224)

   b. Normalize pixel values

3. Tokenize and preprocess text:

   a. Convert words into numerical tokens using a vocabulary

   b. Pad or truncate captions to a fixed length

4. Split dataset into training, validation, and test sets

END

2. Feature Extraction using ViT

A pre-trained ViT model is used to extract high-level feature representations from input images. The input image fed into the ViT model, which returns a feature vector representing the visual content. Extracted features are projected to the required dimensionality by the decoder.

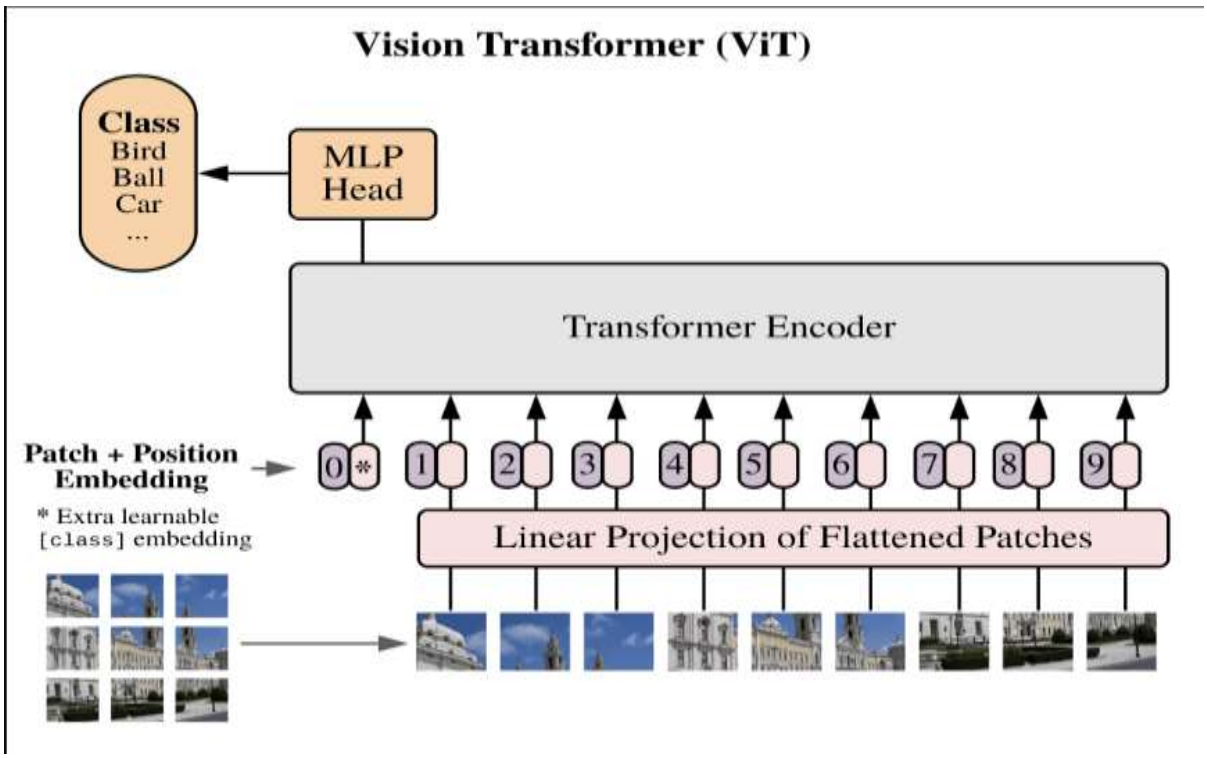


Figure 14: Vision Transformer

START

1. Load pre-trained Vision Transformer (ViT) model

2. Remove the classification head to extract feature embeddings

3. Convert images into patches and pass them through the Transformer encoder:

a. Divide the image into small patches (e.g., 16x16)

b. Flatten patches and apply linear projection

c. Add positional embeddings

d. Pass through multiple self-attention layers

4. Extract final feature representation from ViT output

END

3. Caption Generation using LSTM

The LSTM network-based decoder forms the captions in a word-synchronous manner upon having the feature vectors of images from the encoder. The decoder takes as input the image feature vector and the previously generated word. A fully connected layer projects LSTM outputs onto vocabulary probabilities.

START

1. Initialize LSTM decoder with embedding and output layers

2. For each training sample:

a. Take the image features from ViT as input

b. Initialize LSTM's hidden state with the image features

c. Iterate over caption tokens:

i. Pass previous word’s embedding to LSTM

ii. Generate hidden state and predict next word

iii. Append predicted word to output sequence

3. Compute loss (Cross-Entropy) and update model parameters using backpropagation

4. Repeat training for multiple epochs until convergence

END

4. Prediction (Generating Captions)

The following operational steps are followed to perform the inference in generating captions for new images by the trained model: the image is passed through the ViT encoder to get feature embeddings; the decoder generates words sequentially, taking as input the previously predicted word at each step; beam search is one of the strategies to enhance generation by considering multiple possible sequences and selecting the most probable one. This caption is obtained by converting the predicted sequence of indices back into words.

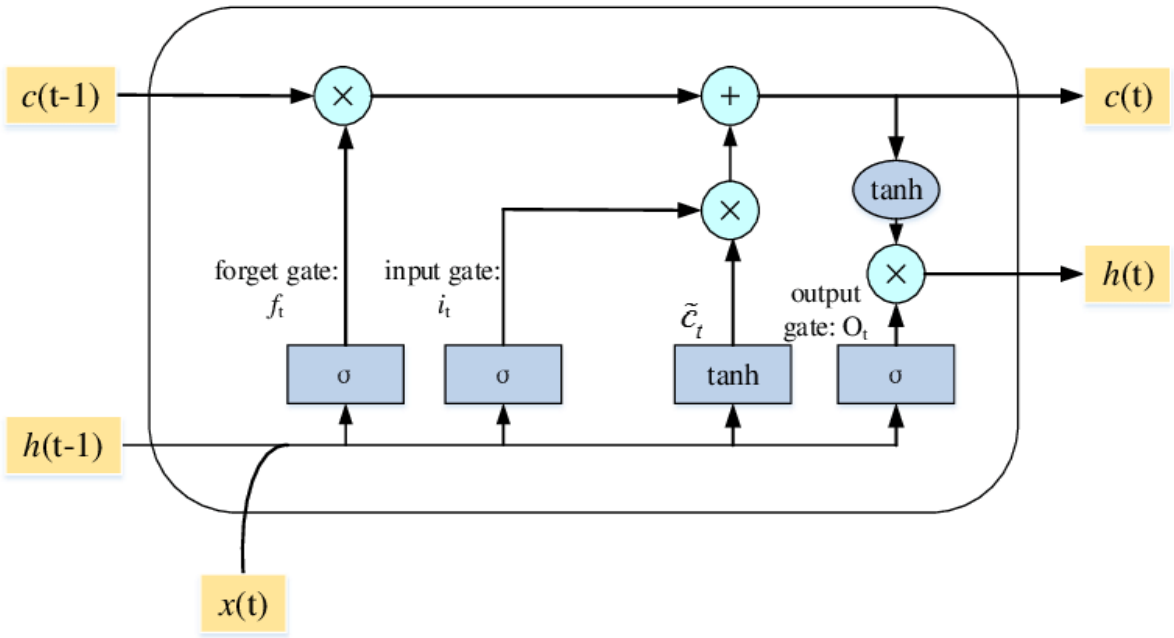


Figure 15: Long Short Term Memory (LSTM)

START

1. Given a new image:

   a. Extract features using ViT

   b. Initialize LSTM's hidden state with extracted features

2. Start caption generation:

a. Begin with the `<start>` token

b. Predict next word using LSTM

c. Append predicted word to output caption

d. Repeat until `<end>` token or max caption length is reached

3. Return the generated caption

END

Flow of Algorithm and its equation

1. Input Image Patch Embedding

The input image I of size $H \times W \times C$ is divided into fixed-size patches of size $P \times P$. The number of patches is:

$$N = (H \times W) / P^2$$

Each patch is flattened into a vector x_i, and a linear transformation is applied:

$$z_i = W_e x_i + b_e$$

where:

- W_e is the embedding matrix,
- b_e is the bias term.

A learnable class token x_cls is prepended, and positional embeddings E_pos are added:

$$Z_0 = [x\_cls, z_1, z_2, \ldots, z\_N] + E\_pos$$

2. Multi-Head Self-Attention (MHSA)

Self-attention is computed using Query (Q), Key (K), and Value (V) matrices:

$$Q = ZW\_Q, K = ZW\_K, V = ZW\_V$$

The Scaled Dot-Product Attention is:

$$Attention(Q, K, V) = softmax((QK^T) / \sqrt{d_k}) V$$

where:

- d_k is the dimension of keys/queries,
- W_Q, W_K, W_V are learnable weight matrices.

For multi-head attention, multiple attention heads are concatenated:

$$MHSA(Q, K, V) = Concat(head_1, \ldots, head_h) W_o$$

where W_o is the output projection matrix.

3. Feed-Forward Network (FFN)

Each transformer block contains a Feed-Forward Network (FFN):

$$FFN(X) = ReLU(X W_1 + b_1) W_2 + b_2$$

where:

- W_1, W_2 are weight matrices,
- b_1, b_2 are biases.

This is followed by Layer Normalization and Residual Connections:

$$Z' = LayerNorm(Z + MHSA(Z))$$

$$Z\_out = LayerNorm(Z' + FFN(Z'))$$

4. Caption Generation Using LSTM

After feature extraction from ViT, an LSTM-based decoder generates the caption sequentially:

$$h_t, c_t = LSTM(y_{t-1}, h_{t-1}, c_{t-1})$$

$$P(y_t) = softmax(W\_y\, h_t + b\_y)$$

where:

- h_t and c_t are the hidden and cell states,
- y_t is the predicted word at time t,
- W_y and b_y are learnable parameters.

5. Final Objective Function

The model is trained using Cross-Entropy Loss:

$$L = - \Sigma \text{ (from } t=1 \text{ to } T) \log P(y_t \mid y_1, \ldots, y_{t-1})$$

where T is the length of the caption.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Implementation

Following the study and design of the system, implementation is the next step. The code is required to produce the desired result. The verification and validation of the coding exist to determine whether all of the generated systems meet the user requirements. The various tools aid in building modules, which provide a starting point for coding. In addition, testing must end before the system launch.

Tools Used: Software developers use a variety of tools for the development, training, and deployment of the system. The coding and model training of the system are carried out using Kaggle and Jupyter notebook. As a web-based system, it includes several case tools and programming languages for scripting with their necessary database server and framework.

5.2 Analysis and Design Tools

1. Draw.io is a versatile tool that allows developers to create detailed and visually appealing diagrams for system design. It provides a wide range of shapes, symbols, and templates tailored for various types of diagrams, including flowcharts, UML diagrams, and network diagrams. With its intuitive drag-and-drop interface, users can easily design and customize diagrams to accurately represent the system's architecture, data flows, and processes. Draw.io also supports collaboration features, enabling team members to work together in real-time on diagram creation and editing.

2. ClickUp is a widely used application that offers powerful features for creating Gantt charts, a type of bar chart that visually represents the schedule of tasks and activities in a project. Using ClickUp developers can easily organize and allocate tasks, assign resources, and set dependencies to create a comprehensive project schedule.

Gantt charts generated in ClickUp provide a clear overview of project timelines, milestones, and progress, making it easier for developers to manage and track project activities effectively.

3. MS Word (2013) is a word processing application commonly used for project documentation and report writing. It offers a range of formatting options, templates, and features specifically designed for creating professional-looking documents. Developers can use MS Word to document project requirements, specifications, design decisions, and other important information related to the system development process. With its user- friendly interface and collaboration capabilities, MS Word facilitates the creation, editing, and sharing of project documentation among team members, ensuring clarity and consistency in communication.

5.3 Tools and Technologies

1. Programming Languages & Frameworks

- **Python**: Core language for development.
- **PyTorch**: Used for building the ViT encoder and LSTM decoder models.
- **TorchVision**: For image pre-processing (resizing, normalization).
- **Pandas**: For managing image-caption pairs and dataset preprocessing.

2. Hardware & Cloud Resources

- **CUDA**: For GPU acceleration during training.
- **Kaggle**: For free GPU access for training.

3. Model Management

- **Model File Format**: The trained model is saved as a **.**pth file.

4. Optional Deployment

- **Gradio**: For building a simple UI to deploy the model as a web application.

5.5 Module Description

5.5.1 Image Preprocessing Module

```
def preprocess_img(img):
    transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ])
    img = transform(img)
    return img
```

Figure 16: Image Preprocessing Function

Functionality:

This module prepares input images for feature extraction. It ensures that images are in an appropriate format for processing by the Vision Transformer (ViT) model.

Technical Specifications:

- Resizing: Images are resized to a fixed dimension (e.g., 224x224 pixels) to conform to the input size required by the ViT model.
- Normalization: Pixel values are normalized to ensure consistent intensity distribution, which improves model performance.
- Tokenization: Unique tokens such as <start> and <end> are appended to mark the beginning and end of the caption.
- Libraries Used: PIL (Python Imaging Library) , NumPy , TorchVision -

5.5.2 Feature Extraction Module

```
class Encoder(nn.Module):
    def __init__(self, embed_dim, num_unfrozen_layers=8):
        super(Encoder, self).__init__()

        self.vit = ViTModel.from_pretrained('google/vit-base-patch16-224-in21k')

        for param in self.vit.parameters():
            param.requires_grad = False

        if num_unfrozen_layers > 0:
            for layer in self.vit.encoder.layer[-num_unfrozen_layers:]:
                for param in layer.parameters():
                    param.requires_grad = True

        self.fc = nn.Linear(self.vit.config.hidden_size, embed_dim)

    def forward(self, x):
        with torch.no_grad():
            outputs = self.vit(x)
            x = outputs.last_hidden_state[:, 0]

        x = self.fc(x)
        return x
```

Figure 17: Encoder Function to extract features

Functionality:

This module leverages a pre-trained Vision Transformer (ViT) encoder to extract high-level visual features from the input images. It converts images into high-dimensional vector representations that capture crucial visual information.

Technical Specifications:

- Patch Creation: The input image is divided into fixed-size patches (e.g., 16x16 pixels per patch).
- Patch Embedding: Each patch is linearly transformed into an embedding vector.
- Self-Attention Layers: The transformed patches pass through several self-attention layers within the ViT model.

- Feature Vector: The final output is a feature vector that represents the content of the image.
- Libraries Used: TorchVision, Hugging Face Transformers, PyTorch

5.5.3 Caption Generation Module

```
class Decoder(nn.Module):
    def __init__(self, hidden_dim, vocab_size, num_layers):
        super(Decoder, self).__init__()

        self.embedding = nn.Embedding(vocab_size, hidden_dim)
        self.rnn = nn.LSTM(hidden_dim, hidden_dim, num_layers=num_layers, dropout=0.5, batch_first=True)
        self.linear = nn.Linear(hidden_dim, hidden_dim)
        self.projection = nn.Linear(hidden_dim, vocab_size)
        self.non_lin = nn.ReLU()
        self.dropout = nn.Dropout(p=0.5)

    def forward(self, input_batch, hidden_state=None):
        embeddings = self.embedding(input_batch)

        if hidden_state is None:
            output, hidden_state = self.rnn(embeddings)
        else:
            output, hidden_state = self.rnn(embeddings, hidden_state)

        output = self.dropout(self.linear(self.non_lin(output)))
        projection = self.projection(output)
        return projection, hidden_state
```

Figure 18: Caption Generation using LSTM

Functionality:

This module generates Sanskrit captions using a Long Short-Term Memory (LSTM) model based on the extracted image features. It ensures that the generated captions are grammatically accurate and contextually relevant.

Technical Specifications:

- Input to LSTM: The ViT feature vector is used as input to the LSTM decoder.

- Caption Generation: The decoder generates captions one word at a time in a sequential fashion.
- Word Prediction: A softmax layer predicts the most likely next word in the sequence.
- Loss Calculation: Uses cross-entropy loss for training the model.
- Libraries Used: PyTorch, TensorFlow, NumPy

5.5.4 User Interface (UI) Module

Functionality:

This module provides an interactive interface where users can upload images and view the generated captions in Sanskrit.

Technical Specifications:

- Interactive UI: Built using Gradio, allowing users to upload images and get real-time captions.
- Backend Integration: Uses a Python backend to process images and dynamically generate captions.
- Optional Deployment: Can be deployed using Flask for seamless integration into web applications.
- Libraries Used: Gradio

Workflow and Integration

1. Image Upload: The user uploads an image through the UI.
2. Image Preprocessing: The Image Preprocessing Module prepares the uploaded image.
3. Feature Extraction: The Feature Extraction Module (ViT) encodes the image into a feature vector.
4. Caption Generation: The Caption Generation Module (LSTM) utilizes the extracted features to generate a Sanskrit caption.
5. Caption Display: The UI Module displays the uploaded image along with the generated caption.

5.6 Testing

Testing is essential in software development because it detects hazards and reduces maintenance expenses. It generates bug-free code by evaluating predicted and actual outcomes. After each unit ends, testing may begin. Unit and system testing served as tools for the Sanskrit Image Captioning System. Manual testing generates, performs, analyzes, and provides findings to uncover any existing problems. Testing reduces maintenance costs while proving the product's quality, usefulness, and performance by finding risks early on.

5.6.1 Unit Testing

After each component's job completes, unit testing occurs in isolation from the rest of the system. Unit testing determines quality and dependability. It contributes to system verification by answering the question, "Are we building the right model?". Better unit testing aids in another level of testing in which testing applies to a specific function, method, or class, with a set of inputs and the output compared to an intended result.

Test Case for Login

Table 1: Test case for unsuccessful registration

Test Case ID	Test Case Description	Expected Result	Actual Result	Pass/ Fail	Test Data
001	Verify that all the required fields are present and filled.	Display an error message prompting the user to fill out all required fields.	Please fill out all the missing field error.	Pass	Name: Smriti KC Email:smriti@gmail.com Username: Smriti KC Password: smriti123 Repeat Password: smriti

Table 2: Test case for successful registration

Test Case ID	Test Case Description	Expected Result	Actual Result	Pass/ Fail	Test Data
002	Verify that all the required fields are present and filled.	Registration successful! You can now log in.	Registration successful! You can now log in.	Pass	Name: Smriti KC Email:smriti@gmail.com Username: Smriti KC Password: smriti123 Repeat Password: smriti123

Table 3: Test case for unsuccessful login

Test Case ID	Test Case Description	Expected Result	Actual Result	Pass/ Fail	Test Data
001	Verify that the system displays an error for incorrect username.	"Invalid Credentials. Try Again."	“Invalid Credentials. Try Again.”	Pass	Username: Iliya Password: testpass
002	Verify that the system displays an error for incorrect password.	"Invalid Credentials. Try Again."	“Invalid Credentials. Try Again.”	Pass	Username: Smriti KC Password: wrongpass

Table 4: Test Case for successful login

Test Case ID	Test Case Description	Expected Result	Actual Result	Pass/Fail	Test Data
003	Verify that the system allows login with correct username and password.	Successful login and access to the main interface.	Login successful, access granted to the main interface.	Pass	Username: Smriti KC Password: password124

5.6.2 System Testing

After the project completes, it is vital to verify whether the user specification was satisfied. System testing guarantees that the system meets all of the requirements. The practicality of unit testing influences its usefulness and performance. It answers the question, "Are we building the model correctly?". Here the system testing was done manually, and the test results were operational and indicated the intended conclusion.

Table 5: Test case for System Testing

SN	Test Case	Steps	Test Data	Expected Result	Pass/ Fail
001	Registration Page	Go to registration page	Name: Smriti KC Email:smriti@gmail.com Username: Smriti KC Password: smriti123	Registration successful! You can now log in	Pass
002	Verify Login	1. Go to the login page. 2. Enter correct username and password. 3. Click Login	Username: Smriti KC Password: smriti123	Login Successful	Pass
003	Verify that the user can upload an image and generate a caption.	1. Go to the "Upload Image" tab. 2. Upload a valid image. 3. Click "Generate Caption".	Image: 1303335399_b3facd47ab.jpg	एकः बालकः स्लैड् इत्यस्य अधः पतति।'	Pass

004	Verify that the user can select a test case and generate the corresponding caption.	1. Go to the "Test Cases" tab. 2. Select a test case from the dropdown. 3. Click "Generate Caption for Test Case".	Test Case: "एकः कृष्ण-श्वेत-कुक्कुरः तृणस्य माध्यमेन धावति।"	"Generated Caption: एकः कृष्ण-श्वेत-कुक्कुरः तृणस्य माध्यमेन धावति।"	Pass

5.7 Result Analysis

The image captioning model improved consistently over the course of the five training epochs, as per both the loss measures and the evaluation scores. Trends in the model's performance are discussed here and what they mean.

5.7.1 Evaluating Accuracy

1. Training and Validation Loss

The training loss fell steadily from 5.1953 in Epoch 1 to 3.3340 in Epoch 5, a notable 35.8% drop. The trend downward signifies learning from the training data and optimization of the model parameters.

Validation loss took a different trend, from 4.5177 at Epoch 1 to 4.1762 at Epoch 4 and then slightly rising again to 4.2540 at Epoch 5. It indicates that the model is still improving on the training set but is possibly at the initial phases of over fitting after the fourth epoch, as training loss was significantly higher in comparison to validation loss in the last epoch.

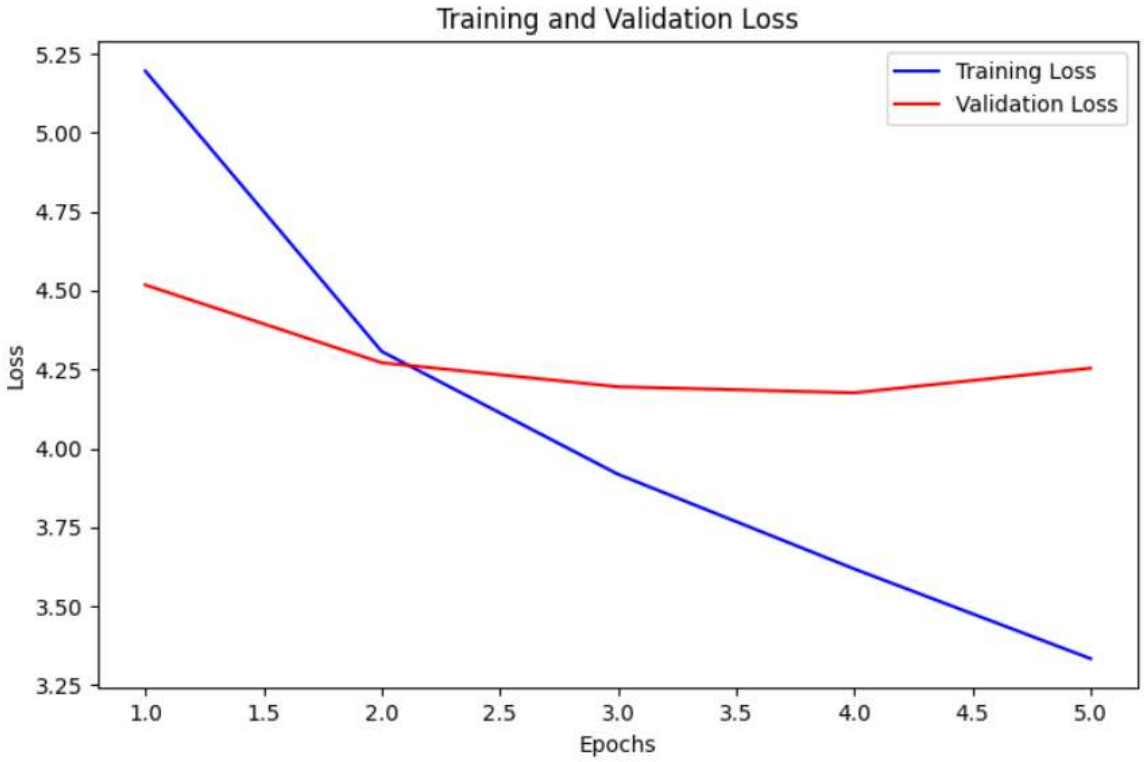


Figure 19: Training and Validation Loss over 5 epochs

2. BLEU Score Analysis

The BLEU score, which measures the precision of n-gram matches between generated captions and reference captions, showed consistent improvement across all epochs:

- Starting at 0.0342 in Epoch 1
- Reaching 0.0622 in Epoch 5

This represents an 81.9% relative improvement, indicating that the generated captions increasingly contained exact phrases found in the reference captions. While the absolute BLEU scores remain relatively low (below 0.1), this is not unusual for image captioning tasks, where there can be multiple valid ways to describe the same image, resulting in lower literal overlap with reference captions.

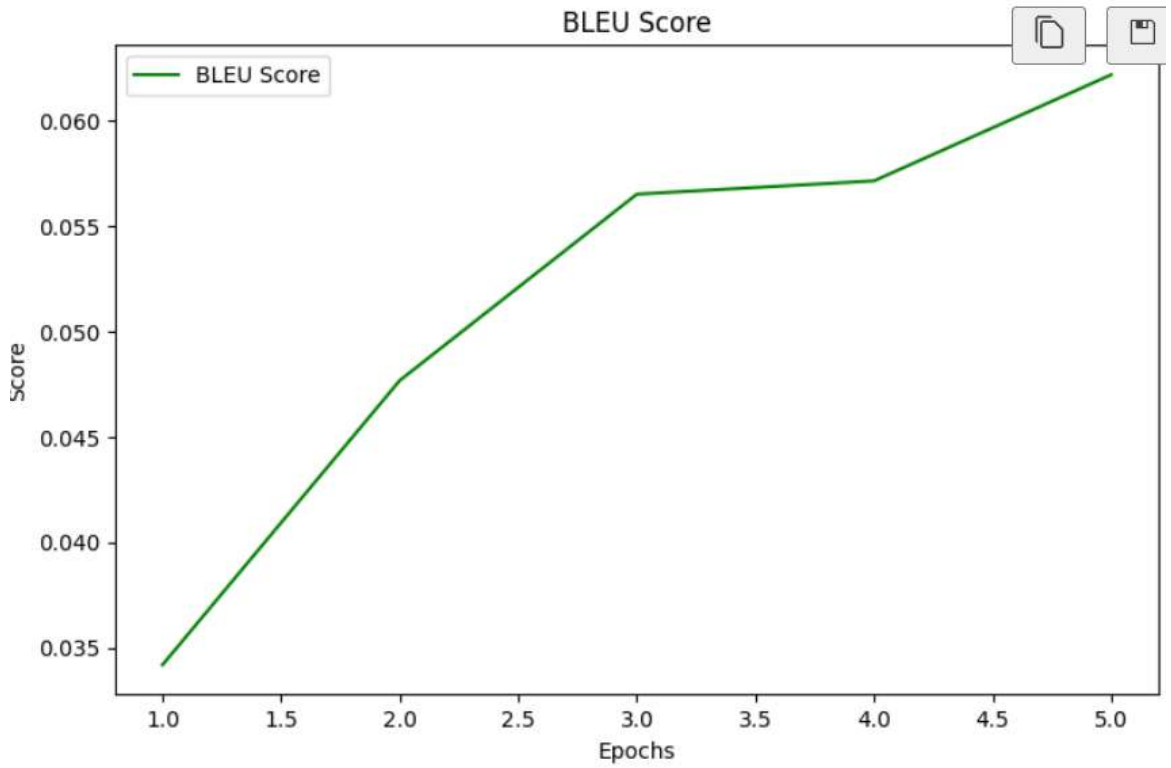


Figure 20: BLEU Score Progression across Training Epochs

3. ROUGE Score Analysis

The ROUGE metrics, which evaluate the overlap between generated and reference captions, also showed steady improvement:

```
Epoch 5 Train Loss: 3.3340
Validation Loss: 4.2540
BLEU Score: 0.0622
ROUGE-1 Score: 0.3454
ROUGE-2 Score: 0.0979
ROUGE-L Score: 0.3411
```

Figure 21: Model Performance Metrics at Epoch 5

ROUGE-1 (Unigram Overlap):

- Increased from 0.2981 in Epoch 1 to 0.3454 in Epoch 5 (15.9% improvement)

ROUGE-2 (Bigram Overlap):

- Increased from 0.0639 in Epoch 1 to 0.0979 in Epoch 5 (53.2% improvement)

ROUGE-L (Longest Common Subsequence):

- Increased from 0.2945 in Epoch 1 to 0.3411 in Epoch 5 (15.8% improvement)

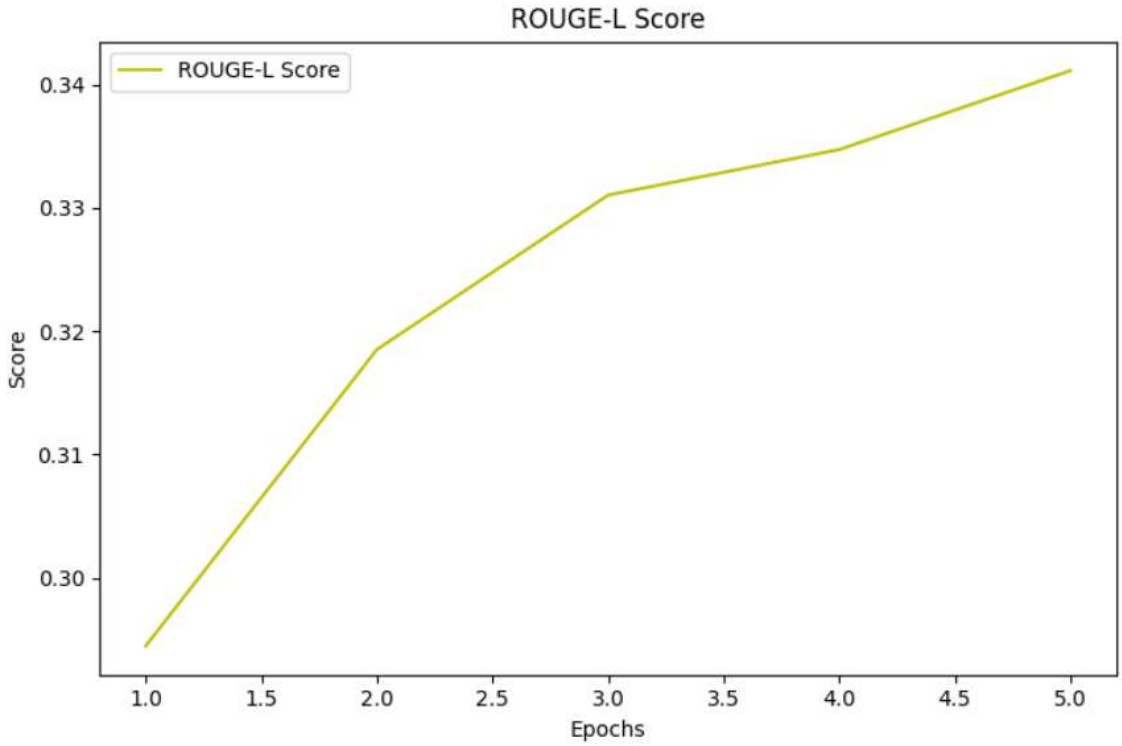


Figure 22: ROUGE-L Score Progression across Training Epochs

The higher ROUGE-1 scores compared to ROUGE-2 indicate that the model is more successful at capturing individual words from the reference captions than consecutive word pairs. This suggests the model is learning the vocabulary necessary to describe image content but still developing in its ability to construct human-like phrases and sentences.

The ROUGE-L scores closely track ROUGE-1, suggesting that the model is capturing meaningful sequences of words that appear in the reference captions.

4. Overall Performance Assessment

The image captioning model demonstrates promising capabilities, as evidenced by the consistent improvement in all evaluation metrics across epochs. The training dynamics reveal several key insights:

1. **Learning Efficiency**: The model shows a strong capacity to learn from the training data, with substantial reductions in training loss and corresponding improvements in evaluation metrics.

2. **Potential Overfitting Concerns**: The slight uptick in validation loss at Epoch 5, coupled with the continuing decrease in training loss, suggests careful monitoring is needed in extended training.

3. **Content Capture**: The relatively higher ROUGE-1 scores compared to ROUGE-2 indicate the model is better at identifying key objects and attributes in images than constructing natural-sounding phrases to describe their relationships.

4. **Improvement Trajectory**: All evaluation metrics show a pattern of diminishing returns in later epochs, with the most substantial gains occurring in the earlier epochs.

While these results demonstrate the model's effectiveness at capturing image content, there remains significant room for improvement in generating fluent and natural-sounding captions. Future work could focus on architectural enhancements to better capture the relationships between objects in images, fine-tuning language modeling components, or exploring attention mechanisms to improve the model's ability to focus on relevant image regions when generating specific parts of captions.

CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS

6.1 Conclusion

It has played a vital role in showing the capability of deep learning in the generation of image descriptions based on Sanskrit, employing an ViT-LSTM image captioning system. It will train the model to relate visual features to meaningful textual representations and in turn will give further scope for research in multimodal AI applications in low-resource languages. While the system provides the functional approach to automatic caption generation certain limitations remain.The dataset matters a lot for the accuracy and quality of the captions, sometimes the model fails on complex or less frequent words. Also, not least, computational constraints and training efficiency affect the overall performance. In spite of all these challenges, the results show that transformer-based encoders combined with sequence-generating decoders can effectively bridge the gap between vision and language in Sanskrit NLP applications.

6.2 Recommendation

Further improvement of the performance and applicability of the model is possible with a number of extensions. First, increasing the size of the dataset, including more diverse images and captions, would lead to better vocabulary coverage and sentence variation, hence less generic predictions. Data augmentation techniques can also be applied to enrich the training samples and enhance model generalization. Second, architectural modifications may replace LSTM with transformer-based decoders for better sentence generation and contextual understanding. Accuracy in output can be further improved by fine-tuning a larger ViT model or incorporating attention mechanisms within the decoder. Further improvements can also be made by tuning hyperparameters related to learning rate, dropout rates, and batch size. Deployment of the model as a user-friendly application can be done using web-based or mobile applications for real-world usage. Future research could explore real-time caption generation, multilingual support, and integration with other applications to improve accessibility and practical adoption of Sanskrit image captioning technology.

REFERENCES

[1] O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and Tell: A Neural Image Caption Generator," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[2] R. N. Shaw, M. Paprzycki and A. Ghosh, Advanced Communication and Intelligent Systems, Springer Nature, 2023.

[3] Y. Gong, R. Li, B. Fu, Y. Liu, J. Wang and R. Li, "ViT-Cap: A Novel Vision Transformer-Based Capsule Network Model for Finger Vein Recognition," *Applied Sciences*, no. 20, p. 10364, 2022.

[4] Y. Lyu, Y. Liu and Q. Zhao, "Performance and Cost Balancing in Vision Transformer-Based Image Captioning," in *ASSE '23: Proceedings of the 2023 4th Asia Service Sciences and Software Engineering Conference*, 2024.

[5] R. Castro, I. Pineda, W. Lim and M. E. Morocho-Cayamcela, "Deep Learning Approaches Based on Transformer Architectures for Image Captioning Tasks," *IEEE Access*, vol. 10, pp. 33679-33694, 2022.

[6] Y. Lyu, Y. Liu and Q. Zhao, "Performance and Cost Balancing in Vision Transformer-Based Image Captioning," in *ASSE 2023: 2023 4th Asia Service Sciences and Software Engineering Conference*, 2024.

[7] E. C. Reddy, G. B. Siva, T. Reddy, Y. Anudeep and R. Jansi, "Synergizing VGG16 Convolutional Features with LSTM Architectures for Image Captioning Mastery," in *Innovations in Cybersecurity and Data Science(ICICDS 2024)*, 2024.

[8] P. Dhamanskar, W. Pereira and S. Khot, "Image Captioning: Analyzing CNN-LSTM and Vision-GPT Models".

[9] I. Nandhini, L. L. Prasanth, T. Nagalakshmi and a. D. Manjula, "Integrating Convolutional and Recurrent Networks for Image Caption Generation: A Unified Approach," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024.

[10] P. Young, A. Lai, M. Hodosh and J. Hockenmaier, 2014. [Online]. Available: https://www.kaggle.com/datasets/adityajn105/flickr8k..

[11] S. Jaiswal, H. Pallthadka, R. P., Chinchewadi and T. Jaiswal, "Optimized Image Captioning: Hybrid Transformers Vision Transformers and Convolutional Neural Networks: Enhanced with Beam Search," *International Journal of Intelligent Systems and Applications (IJISA)*, vol. 16, no. 2, pp. 53-61, 2024.

[12] A. Dosovitskiy, L. Beyer, J. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. H. Aschinger and R. H. Müller, "An Image Is Worth 16×16 Words: Transformers for Image Recognition at Scale," in *9th International Conference on Learning Representations (ICLR)*, 2021.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, 2017.

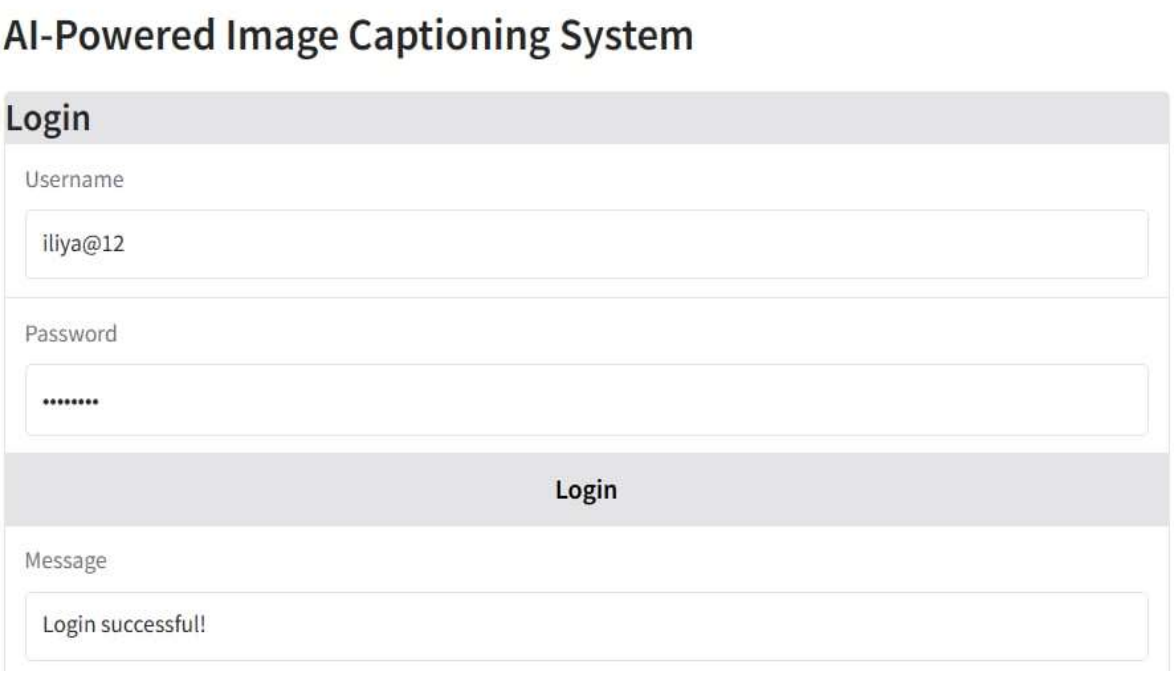
[14] K. A. Sai, K. M. H. S. M. Krishna, Y. Samudrala and T. Anjali, "Image Captioning: Analyzing CNN-LSTM and Vision-GPT Models," in *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, 2024.

[15] W. Liu, S. Chen, L. Guo, X. Zhu and J. Liu, "CPTR: Full Transformer Network for Image Captioning," 2021.

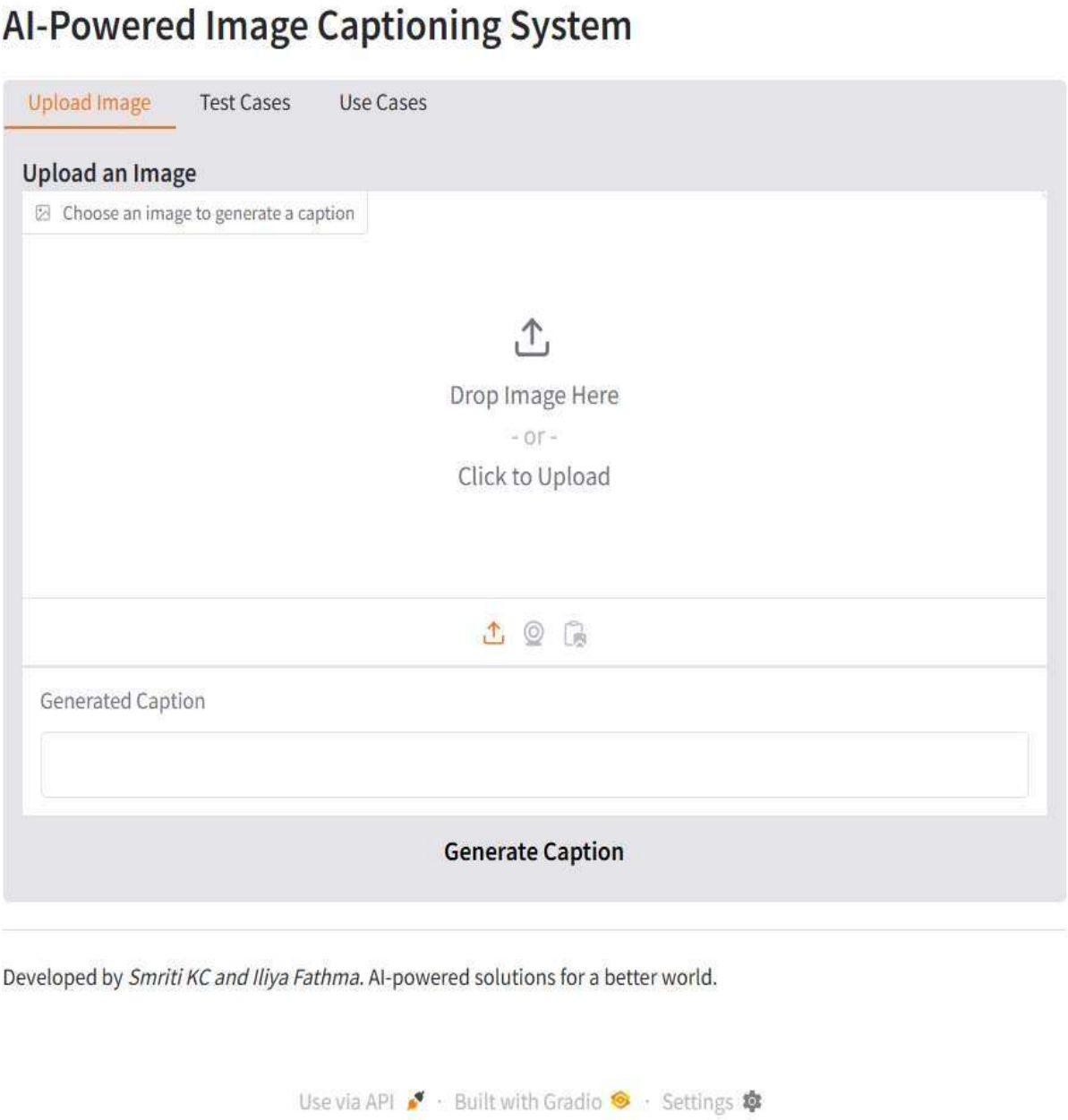
APPENDICES

Registration Page:

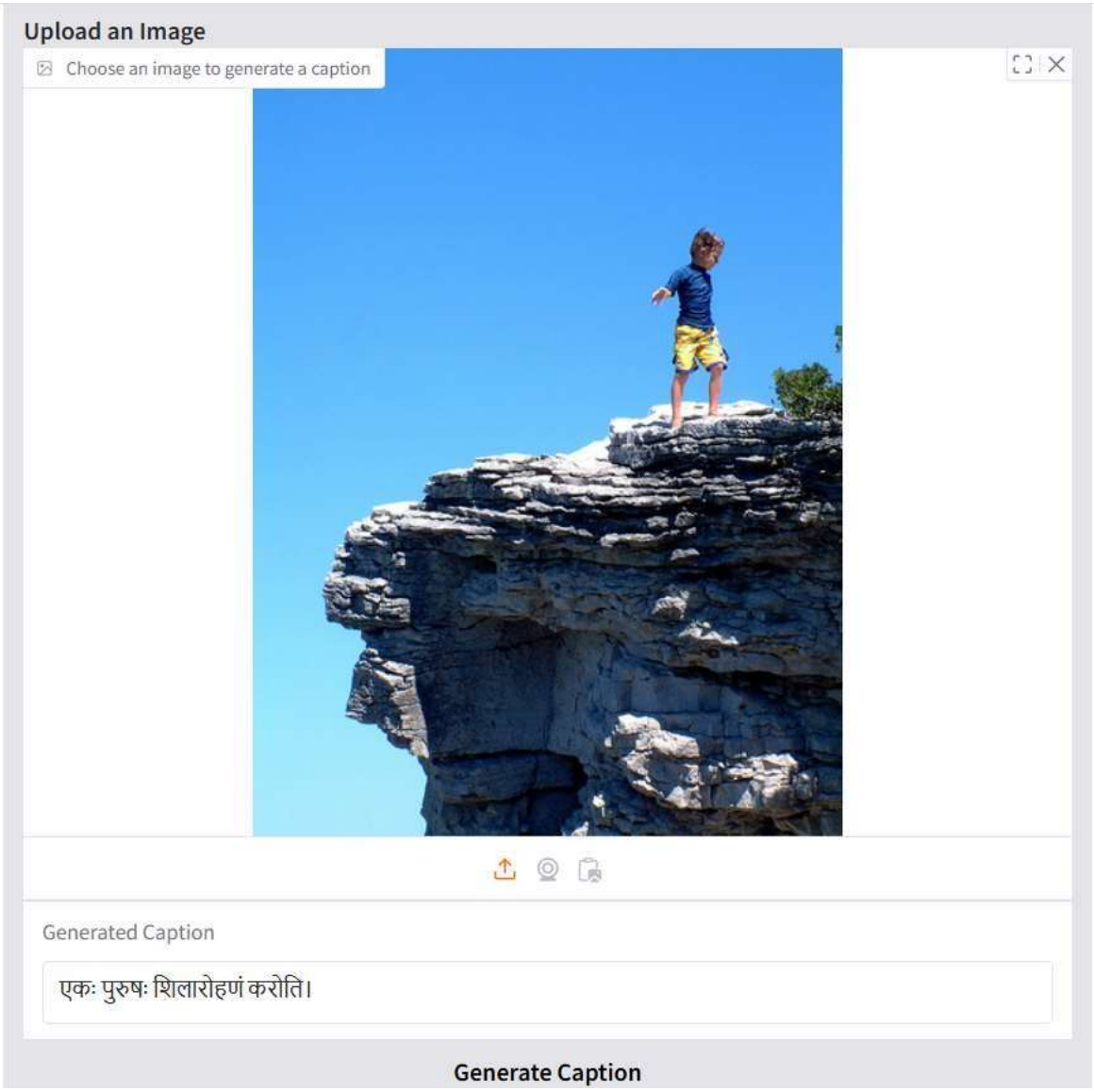
Login Page:


AI-Powered Image Captioning System
Login
Username
iliya@12
Password
Login
Message
Login successful!

User Interface after Login:



Uploading image and generating caption:



AI-Powered Image Captioning System

Upload Image | Test Cases | Use Cases

Upload an Image

Generated Caption

एकः पुरुषः पर्वतस्य उपरि द्विचक्रिकायां चालयति।

Generate Caption

Developed by *Smriti KC and Iliya Fathma*. AI-powered solutions for a better world.

AI-Powered Image Captioning System

Upload Image | Test Cases | Use Cases

Upload an Image

Generated Caption

एकः बालकः साकर्-कन्दुकं किक् कर्तुं प्रयतते।

Generate Caption

Test Case:


AI-Powered Image Captioning System
Upload Image
Test Cases
Use Cases
Test Cases
Select a Test Case
एकः कुक्कुरः तृणस्य माध्यमेन धावति।
✓ एकः कुक्कुरः तृणस्य माध्यमेन धावति।
एकः कुक्कुरः हिमस्य माध्यमेन धावति।
एकः बालकः वर्णरञ्जित-नलिकायुक्ते पूल्-मध्ये पतितः पतति।
एकः बालकः शिलायाः उपरि वायुवे उच्चे स्थितः अस्ति।
स्की-क्रीडकाः वनस्य माध्यमेन पर्वतस्य उपरि गच्छन्ति।
एकः बालकः साकर्-कन्दुकं किक् कर्तुं प्रयतते।

Select a Test Case

एकः कुक्कुरः तृणस्य माध्यमेन धावति।

Image

Generated Caption

Generated Caption: एकः कुक्कुरः तृणस्य माध्यमेन धावति।

Expected Caption

Expected Caption: एकः कुक्कुरः तृणस्य माध्यमेन धावति।

Result

Correct

Image

Generated Caption

Generated Caption: एकः कुक्कुरः हिमस्य माध्यमेन धावति।

Expected Caption

Expected Caption: एकः कुक्कुरः हिमस्य माध्यमेन धावति।

Result

Correct

AI-Powered Image Captioning System

Upload Image | Test Cases | Use Cases

Use Cases

1. Educational Applications:Helps student learn Sanskrit through image based captions.
2. AI and NLP research in low resource languages:Advances AI models for Sanskrit grammar and translation.
3. Content Creation:Helps creators quickly add Sanskrit captions to social media posts.
4. Preservation of ancient language:Helps sustain and promote Sanskrit language by generating captions.

Developed by *Smriti KC and Iliya Fathma*. AI-powered solutions for a better world.